用户手册

Foxit® PDF SDK(ActiveX)

编程指南

适用于Windows

Microsoft® Partner
Gold Independent Software Vendor (ISV)

目录

概述

Foxit PDF SDK ActiveX 是一款可视编程组件，可以让您的应用程序实现高质高效的 PDF 文档显示效果，其体积小巧，系统资源消耗少，可迅速集成于其它应用程序中。

Foxit PDF SDK ActiveX 使用与福昕阅读器相同的核心技术，能够快速、准确、高质量的渲染 PDF 文件。

Foxit PDF SDK ActiveX 使用更加灵活，并且它还内嵌了丰富的界面和编程特性。程序设计人员可通过简单的拖放组件，迅速为应用程序添加 PDF 显示功能。ActiveX 还允许用户对 PDF 文档进行各种操作，如页面导航、改变缩放比例、页面旋转、页面滚动及打印，等等。

Foxit PDF SDK ActiveX 5.0 添加了签名模块，允许开发人员使用 Windows 系统证书中心存储的第三方数字证书签署 PDF 文档，同时，也支持数字签名验证（用于验证含数字签名文档的真实性和完整性）、签名时间戳、签名打印控制等。此外，Foxit PDF SDK ActiveX 5.0 中还添加了许多新功能和新事件，例如异步下载等，允许开发人员更好地控制组件、更好地使用 PDF 文档。

Foxit PDF SDK ActiveX 5.0 提供两个不同的版本，即标准版和专业版。这两个版本提供不同的 GUID，您可以在同一电脑上同时注册和使用。与标准版比较，专业版包含更多的特性：创建/编辑批注、导入/导出表单数据、执行 JS 脚本、将 PDF 导出为文本格式、数字签名等。您可以根据您的应用程序来选择 ActiveX 的版本。您可以在线购买专业版或标准版 ActiveX，如果您想要版本 5.0 中的表单模块、批注模块和电子签章模块，请联系我们的销售团队（发送邮件至 sales@foxitsoftware.cn）了解更多授权信息。

在本开发手册中，

所有标记(*)的属性和功能只属于专业版本，
所有标记(#)的属性和方法只属于表单模块，
所有标记(^)的属性和方法只属于注释模块，
所有标记(&)的属性和方法只属于电子签章模块。
Foxit PDF SDK ActiveX 可运行在 Windows 系统上，是一个独立的控件，无需再安装任何的 PDF 应用程序。需要注意的是，在 Windows 系统下，只有 Admin 权限的用户才能成功的注册 ActiveX 控件。

福昕已经为开发人员编写 VC++示例代码，通过示例代码即可快速的掌握如何使用 ActiveX 提供的属性和方法。请访问 www.foxitsoftware.cn 下载相关的示例代码。

Foxit PDF SDK ActiveX 5.0 专业版增加签名模块，需要另行授权许可。如需使用签名模块，请通过邮件（sales@foxitsoftware.cn）与我们联系。

解锁码: 如果您已经购买了 Foxit PDF SDK ActiveX 并且已经拿到正式的 ActiveX 控件和解锁码，在调用任何一个函数前，请确保在您的程序中至少调用过一次 UnLockActiveX or UnLockActiveXEx 函数。该函数将在后续的章节中有详细描述。如果您只是想评估 ActiveX，则无需调用该函数。

标准版 GUID：0F6C092B-6E4C-4976-B386-27A9FD9E96A1

专业版 GUID：F53B7748-643C-4A78-8DBC-01A4855D1A10

入门教程

福昕 ActiveX 控件提供的是一个独立的 OCX 文件。通过执行命令"regsvr32 OCX name"即可成功注册。需要注意的是，必须指定 OCX 的完整的路径。

ActiveX 控件不仅提供人机的交互界面，同时提供一系列属性和方法以便应用程序能够方便的操作 ActiveX 控件。

在本章列出的示例代码中，我们设定创建了一个 ActiveX 控件的实例，该实例的名称是 FoxitReaderSDK。

标准版&专业版

1) 打开一个 PDF 文件

例如，打开一个名为“testdoc.pdf”的 PDF 文档。

```
// 提示: 如果您是在试用 ActiveX，则无需进行解锁。
// 试用的水印将在所有页面上显示。
// 对于已经付费的用户，则首先需要进行解锁。
FoxitReaderSDK.UnLockActiveX(“license_id”,”unlock_code”);
FoxitReaderSDK.OpenFile(“testdoc.pdf”,””);
```

2) 跳转到指定页面

例如，我们计划跳转到当前 PDF 文档的第三页。

```
FoxitReaderSDK.GoToPage(2). //页面索引值从 0 开始
```

3) 缩放页面

如果您想显示 PDF 页面的原始尺寸，可在 VC 中添加如下代码:

```
FoxitReaderSDK.SetZoomLevel(0);
```

或者

```
FoxitReaderSDK. SetZoomLevel(100);
```

如果您想设置缩放比率 Zoomlevel 为 200%，可使用如下代码:

```
FoxitReaderSDK. SetZoomLevel(200);
```

4) 旋转页面视图

一个 PDF 页面可以在四个不同方向上显示: 垂直正向、旋转 90 度、旋转 180 度或旋转 270 度。可以调用 SetRotate 函数或者设置 Rotate 属性来改变页面显示的方向。

将页面顺时针旋转 90 度，可在 VC 中使用如下代码:

```
FoxitReaderSDK. SetRotate(1);
```

5) 打印 PDF 文件

- 调用 PrintWithDialog 方法，会弹出一个打印设置窗口，完成设置后即可将 PDF 文件打印出来。
- 如果想实现静默打印，即不弹出打印设置窗口，则需要通过 IPDFPrinter 提供的接口来完成打印操作。

6) 隐藏或显示用户界面元素

- 调用 ShowToolBar 函数即可显示或隐藏工具栏，调用 ShowStatusBar 可以显示或隐藏状态栏等。
- 如果要创建自己的工具栏，只需隐藏 ActiveX 控件内置工具栏，然后在控件外部创建属于自己的工具栏。

7) 遍历整个大纲目录树

调用 GetOutlineFirstChild 和 GetOutlineNextSibling 即可遍历整个大纲目录树，通过 IPDFOutline 提供的接口即可查看各节点的信息。

8) 检索 PDF 文件

调用 FindFirst 函数，可定位到指定文本在整个 PDF 文档中的第一个位置。如果没有找到与给定文本相匹配的内容，则函数返回值为 0。如果找到与指定文本相匹配的内容，则函数返回非零值，同时高亮对应的文本内容。重复调用 FindNext 函数，可定位到下一个与给定文本相匹配的内容。
如果想要在不打开和不显示的情况对其他 PDF 文件进行检索，只需调用 FindFileFirst 函数和 FindFileNext 函数。

签名模块

Foxit PDF SDK ActiveX 5.0 添加了签名模块。

开发开人员可以使用 Windows 系统证书中心存储的第三方数字证书签署 PDF 文档。签名模块支持数字签名验证（用于验证含数字签名文档的真实性和完整性）、签名时间戳、签名打印控制等。

注释模块

通过使用不同的注释工具，最终用户可以在 PDF 文件中绘制直线、圆形以及其他各种图形。要实现这样的功能，只需在应用程序中改变 CurrentTool 属性的值。例如，将 CurrentTool 属性的值设置为 “Line Tool”。

Foxit PDF SDK ActiveX 4.0 添加了注释模块。通过注释模块提供的接口，可以操作 PDF 注释对象，创建属于自己的注释。

表单模块

从版本 3.0 开始，Foxit PDF SDK ActiveX 提供了 PDF 表单的操作接口。通过调用 GetCurrentForm 函数，可获取当前 PDF 文件的 IForm 接口指针，然后使用 IPDFForm 和 IPDFFormField 提供的属性和方法来执行针对 PDF 表单的操作。

以下代码展示了如何在一个 PDF 页面中添加一个按钮:

```
CPDFformField button1=form1.AddField("button1","button",m_nCurPage,0,0,55,30);
button1.SetButtonCaption("N","Normal");
button1.SetButtonCaption("R","Rollover");
button1.SetButtonCaption("D","Down");
button1.SetBehavior("push"); // push;Invert; NULL; Outline;
button1.SetTooltip("reset all form");
button1.SetTextFont("Courier");
button1.SetTextSize(15);
button1.SetJavaScriptAction("down","app.alert(\"Mouse Down!\")");
```

IFoxitPDFSDK

本章节主要讲解的是 ActiveX 提供的所有的属性和方法。需要注意的是，所有的描述均基于 C 语法。如果您使用的编程语言不同于 C/C++，请遵守对应编程语言的语法。

备注:

标记(*)的函数属于专业版；

标记 (#)的函数属于表单模块；

标记（^)的函数属于批注模块；

标记(&)的函数属于电子签章模块。

属性

1) FilePath

类型：

BSTR，只读

描述：

PDF 文件的完整路径。 如果当前没有打开的 PDF 文件，或者 PDF 文件是通过缓存或者流中打开的，则该属性的值为空。

2) Password

类型：

BSTR，只读

描述：

PDF 文件的打开密码。

3) PageCount

类型：

long，只读

描述：

PDF 文件的总页数。

4) CurPage

类型：

long，只读

描述：

当前显示的 PDF 页面的索引值，值从 0 开始。

5) Rotate

类型：

short，读/写

描述：

PDF 文件显示时的旋转情况，其值如下所列：

0 = 垂直正向；

1 = 顺时针旋转 90 度；

2 = 旋转 180 度；

3 = 逆时针旋转 90 度。

6) Zoomlevel

类型：

long，读/写

描述：

PDF 文档的缩放比率。

一般情况下，缩放比率在 10~1600 之间，也可使用如下特定的值:

0 = 显示页面的原始尺寸，相当于 100%。

1 = 以适当的缩放比率显示页面，确保客户端显示窗口能够显示整个 PDF 页面。

2 = 以适当的缩放比率显示页面，确保在水平方向上页面充满整个客户端显示窗口。

7) CurrentTool

类型：

BSTR，读/写

描述：

ActiveX 当前支持的工具。有效值如下：

“Hand Tool”

“ZoomOut Tool”

“ZoomIn Tool”

“Select Text Tool”

“Find Text Tool”

“Snapshot Tool”

*“Loupe Tool”

*“Magnifier”

*“Annot Tool”

*“Rectangle Link Tool”

*“Quadrilateral Link Tool”

*“Arrow Tool”

*“Line Tool”

*“Dimension Tool”

*“Square Tool”
*“Rectangle Tool”
*“Circle Tool”
*“Ellipse Tool”
*“Polygon Tool”
*“Cloudy Tool”
*“Polyline Tool”
*“Pencil Tool”
*“Rubber Tool”
*“Highlight Tool”
*“Underline Tool”
*“Strikeout Tool”
*“Squiggly Tool”
*“Replace Tool”
*“Note Tool”
*“Push Button Tool”
*“Check Box Tool”
*“Radio Button Tool”
*“Combo Box Tool”
*“List Box Tool”
*“Text Field Tool”
*“Distance Tool”
*“Perimeter Tool”
*“Area Tool”
*“Typewriter”
*“CallOut”
*”Textbox”
*“Image Tool”
*“Sound Tool”
*“Movie Tool”
*“FileAttachment Tool”
*“Attach a file”
*“ESignature Tool”

备注：

通过调用 CountTools 函数可以获取当前 ActiveX 版本所支持的工具的数量，然后调用 GetToolByIndex 函数来获取每一个工具的名称。

8) Printer

类型：

IPDFPrinter，只读

描述：

该打印机属性返回的是一个 IPDFPrinter 接口，通过接口可实现打印机的管理和打印操作。

9) DocumentInfo

类型：

IPDFDocumentInfo*，只读

描述：

该属性返回的一个 IPDFDocumentInfo 接口，通过接口可获取 PDF 文件的属性信息，包括：

Author(作者)

Producer(创建者)

CreatedDate(创建时间)

Keywords(关键词)

ModifiedDate(修改时间)

Creator(创建工具)

Subject(主题)

Title(标题)

10) ActiveXVersion

类型：

BSTR，只读

描述：

ActiveX 的版本信息。

11) *bHasFormFields

类型：

BOOL，只读

描述：

该属性用于判断 PDF 文件中是否包含表单域：如果当前文件中含有 PDF 表单域，则该属性的值为 True，否则为 False。

12) *bHighlightFormFields

类型：

BOOL，读/写

描述：

该属性用于表示是否高亮 PDF 文件中的表单域：设置该属性值为 True，将会高亮所有的 PDF 表单域，突出 PDF 表单域，给出提示的作用。

13) *FormFieldsHighlightAlpha

类型：

short, 读/写

描述：

该属性用于表示 PDF 表单域在被高亮时的透明度。0 代表完全透明，255 代表不透明。

14) *FormFieldsHighlightColor

类型：

OLE_COLOR, 读/写

描述：

该属性值用于表示 PDF 表单域的高亮颜色。

方法

1) 控件解锁

- **UnLockActiveX**

使用（从福昕公司获取的）正式授权来解锁 ActiveX。

函数原型：

Void UnLockActiveX(BSTR lisence_id, BSTR unlock_code)

参数：

license_id - 授权文件中包含的 SN。

unlock_code - 授权文件中包含的 Unlock。

返回值：

无

备注：

购买 ActiveX 后，获取正式授权的 OCX 文件和对应的授权文件，请首先调用该函数解锁以去除试用水印，再进行其他函数的调用。

- **UnLockActiveXEx**

使用（从福昕公司获取的）正式授权来解锁 ActiveX。

函数原型：

Void UnLockActiveXEx(BSTR strLicense)

参数：

strLicense - 福昕公司提供的 ActiveX 授权码。

返回值：

无

备注：

该函数的功能与 UnlockActiveX 相同。

- *** IsUnLocked**

查看当前 ActiveX 控件版本是否已解锁。

函数原型：

boolean IsUnLocked()

参数：

无

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- ***RemoveEvaluationMark**

在正式授权的情况下，移除试用水印。

函数原型：

BOOL RemoveEvaluationMark()

参数：

无

返回值：

成功则返回 TRUE，反之则返回 FALSE。

2) 全局设置

- **SetCurrentLanguage**

通过指定语言识别号来切换 ActiveX 用户界面的语言。要求存在对应的 XML 文件，根据语言识别号切换到相应的语言。

函数原型：

void SetCurrentLanguage(short LanguageID)

参数：

LanguageID - 语言识别号。有效值为 1-34，每个数字代表一种语言。

1 - 阿拉伯语
2 - 保加利亚语
3 - 匈牙利语
4 - 加泰罗尼亚语
5 - 捷克语
6 - 简体中文
7 - 繁体中文
8 - 丹麦语
9 - 荷兰语
10 - 英语

11 - 爱沙尼亚语
12 - 芬兰语
13 - 法语
14 - 加利西亚语
15 - 德语
16 - 希腊语
17 - 意大利语
18 - 韩语
19 - 拉托维亚语
20 - 立陶宛语
21 - 挪威语
22 - 波兰语
23 - 葡萄牙语
24 - 巴西葡萄牙语
25 - 罗马语
26 - 俄罗斯语
27 - 斯罗维尼亚语
28 - 西班牙语
29 - 瑞典语
30 - 土耳其语
31 - 希伯来语
32 - 日语
33 - 泰语
34 - 瓦伦西亚语

返回值：

无

备注：

ActiveX 支持对用户界面进行动态的语言切换，支持的语言多达 34 种。

要求加载额外的语言包(xml 文件)。如需指定的语言包，请发邮件至 sales@foxitsoftware.cn 与我们联系。

- **SetCurrentLanguageByString**

通过指定语言包（XML 文件）来切换 ActiveX 用户界面的语言。

函数原型：

void SetCurrentLanguageByString(BSTR FileName)

参数：

FileName - XML 语言包的路径。以下是 34 种语言及其对应的 XML 语言包名称。

阿拉伯语 - lang_ar_ae.xml

保加利亚语	-	lang_bg_bg.xml
匈牙利语	-	lang_hu_hu.xml
加泰罗尼亚语	-	lang_ca_es.xml
捷克语	-	lang_cz_cz.xml
简体中文	-	lang_zh_cn.xml
繁体中文	-	lang_tw_cn.xml
丹麦语	-	lang_da_dk.xml
荷兰语	-	lang_nl_nl.xml
英语	-	lang_en_us.xml
爱沙尼亚语	-	lang_et_ee.xml
芬兰语	-	lang_fi_fi.xm
法语	-	lang_fr_fr.xml
加利西亚语	-	lang_gl_es.xml
德语	-	lang_de_de.xml
希腊语	-	lang_el_gr.xml
意大利语	-	lang_it_it.xml
韩语	-	lang_ko_kr.xml
拉托维亚语	-	lang_lv_lv.xml
立陶宛语	-	lang_lt_lt.xml
挪威语	-	lang_nb_no.xml
波兰语	-	lang_pl_pl.xml
葡萄牙语	-	lang_pt_pt.xml
巴西葡萄牙语	-	lang_pt_br.xml
罗马语	-	lang_ro_ro.xml
俄罗斯语	-	lang_ru_ru.xml
斯罗维尼亚语	-	lang_sl_si.xml
西班牙语	-	lang_es_es.xml
瑞典语	-	lang_sv_se.xml
土耳其语	-	lang_tr_tr.xml
希伯来语	-	lang_he_il.xml
日语	-	lang_jp_jp.xml
泰语	-	lang_th_th.xml
瓦伦西亚语	-	lang_va_es.xml

返回值：

无

备注：

ActiveX 支持对用户界面进行动态的语言切换 ,支持的语言多达 34 种。只需加载额外的语言包(xml 文件)。如需指定的语言包，请发邮件至 sales@foxitsoftware.cn 与我们联系。

- **SetModulePath**

指定 fpdfcjk.bin 组件的路径，用以显示 PDF 文件中的中文、日文和韩文文本。

函数原型：

void SetModulePath(LPCTSTR lpFolderName)

参数：

lpFolderName - fpdfcjk.bin 组件的完整路径。

返回值：

无

备注：

fpdfcjk.bin 组件用以显示 PDF 文档中的中文、日文和韩文文本。

- **SetLogFile**

在应用程序中调用该函数可设置一个日志文件，该日志文件将记录程序中调用的每一个接口。

函数原型：

BOOL SetLogFile(BSTR filepath)

参数：

filepath - 日志文件的路径。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- **AboutBox**

弹出”关于”对话框。

函数原型：

Void AboutBox()

参数：

无

返回值：

无

3) 打开和关闭 PDF

- **OpenFile**

从本地磁盘中或从 HTTP 网络服务器上打开一个 PDF 文件。

函数原型：

BOOL OpenFile (BSTR FilePath, BSTR Password)

参数：

FilePath - 本地 PDF 文件路径或者 HTTP 服务器上的 URL 地址。

Password - PDF 文件的打开密码。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

备注：

通过该接口打开的 PDF 文件，不会被程序独占；也就是，同一时间，其他应用程序还可打开这一个 PDF 文件。

- **OpenMemFile**

打开内存中 PDF 文件。

函数原型：

BOOL OpenMemFile(long pBuffer, long Size, BSTR Password)

参数：

pBuffer - 内存中指向 PDF 数据的指针。

Size - PDF 数据的大小。

Password - PDF 文件的打开密码。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- **OpenBuffer**

从缓存中打开一个 PDF 文件。

函数原型：

BOOL OpenBuffer(VARIANT Buffer, long size, BSTR password)

参数：

Buffer - 包含 PDF 数据的字节数组。

Size - PDF 数据的大小。

Password - PDF 文件的打开密码。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- **OpenStream**

通过从 IStream 类提供的接口中获取 PDF 数据的方式打开一个 PDF 文件。

函数原型：

BOOL OpenStream (IStream* Stream, BSTR Password)

参数：

Stream - 一个 IStream 的指针，指向 PDF 数据。

Password - PDF 文件的打开密码。如果无密码，则值为空。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- **SetFileStreamOption**

为一个 PDF 文件设置文件流标记。

函数原型：

Void SetFileStreamOption(BOOL bFileStream)

参数：

bFileStream - 文件流标记。

返回值：

无

备注：

如果一个 PDF 文件被频繁的访问，可以通过该方法将内容流加载到内存中以提高执行效率；当然这样将会消耗更多的内存。

- **OpenCustomFile**

通过用户自定义方式打开一个 PDF 文件。

函数原型：

BOOL OpenCustomFile(BSTR Password)

参数：

Password - PDF 文档的打开密码。如果无密码，则值为空。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

备注：

当应用程序调用该方法时，ActiveX 控件将触发 CustomFileGetSize 和 CustomFileGetBlock 事件。在这两个事件中，应用程序将按照用户自定义的格式打开一个 PDF 文件，并返回文件的大小和数据。

- **OpenFtpFile**

打开一个 FTP 服务器上的 PDF 文件。

函数原型：

BOOL OpenFtpFile(BSTR ftpName, BSTR username, BSTR userPassword, long port, BSTR filePath, BSTR filePassword, boolean Passive)

参数：

ftpName - FTP 服务器的名称。

Username - FTP 服务器的登录用户名。

userPassword - FTP 服务器的登录密码。

port - FTP 服务器端口。

filePath - FTP 文件的路径。

filePassword - PDF 文件的密码。

Passive - 该属性值表示是否主动连接服务器。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- **CloseFile**

关闭当前打开的 PDF 文件。

函数原型：

Void CloseFile()

参数：

无

返回值：

无

4) PDF 保存

- **SaveAs**

将当前打开的文档另存为一个新的文件。

函数原型：

Void SaveAs (BSTR FileName)

参数：

FileName - 将另存为的完整本地文件路径。

返回值:

无

- **Save**

保存当前打开的文档。

函数原型：

Void Save ()

参数：

无

返回值：

无

备注：

该函数只能保存本地磁盘中的 PDF 文件，无法保存从 URL 中打开的 PDF 文件。

- **SaveToStream**

将当前打开的文档保存到内存流中。

函数原型：

IStream* SaveToStream()

参数：

无

返回值：

返回一个 IStream 指针。IStream 提供的接口支持将 PDF 文件数据从 IStream 对象中读取，也支

持将 PDF 数据写入 IStream 对象中。

- ***SetShowSavePrompt**

关闭 PDF 文档时，设置弹出保存对话框。

函数原型：

void SetShowSavePrompt(boolean bShow, short Result)

参数：

bShow - 该属性值表示是否弹出保存对话框。

Result - bShow 为 TRUE 的时候，弹出保存提示框，Result 无效。bShow 为 FALSE 的时候不弹出保存提示框，Result 为 1 或 6 的时候，是会后台保存的，其他值无效。

返回值：

无。

5) 页面导航

- **ExistForwardStack**

查询下一个视图是否存在。

函数原型：

BOOL ExistForwardStack()

参数:

无

返回值：

如果存在下一个视图，则返回 TRUE，反之则返回 FALSE。

备注：

视图定义：阅读到 PDF 文档的某一个位置或者 PDF 文档显示的某一个状态称作视图。用户的每一个操作都会产生一个新的视图。例如，如果用户跳转到一个新的页面，然后进行了缩放页面，那么这两个操作将产生两个新的视图。程序通过调用这一系列的方法，可以很方便的在不同的视图间进行切换。

- **GoForwardStack**

跳转到下一个视图。

函数原型：

Void GoForwardStack()

参数：

无

返回值：

无

- **ExistBackwardStack**

查询是否存在上一个视图。

函数原型：

BOOL ExistBackwardStack()

参数：

无

返回值：

如果存在上一个视图，则返回 TRUE，反之则返回 FALSE。

- **GetVisibleLeftTopPage**

获取显示区域中左上角的页面页码。

函数原型：

Long GetVisibleLeftTopPage()

参数：

无

返回值：

返回显示区域中左上角的 PDF 页面索引值。

- **GoBackwardStack**

跳转到上一个视图。

函数原型：

Void GoBackwardStack ()

参数：

无

返回值：

无

- **GoToNextPage**

跳转到当前显示的 PDF 文件页面的下一页。

函数原型：

Void GoToNextPage()

参数：

无

返回值：

无

- **GoToPrevPage**

跳转到当前显示的 PDF 文件页面的上一页。

函数原型：

Void GoToPrevPage()

参数：

无

返回值：

无

- **GoToPage**

跳转到当前的 PDF 文件的指定页面。

函数原型：

void GoToPage(long page_index)

参数：

page_index - PDF 文件页面的索引值。

返回值：

无

- **GoToPagePos**

跳转到当前 PDF 文档中的一个指定位置。

函数原型：

Void GoToPagePos (long nPageIndex, float PageX, float PageY)

参数：

nPageIndex - PDF 页面的索引值。

PageX - PDF 页面内的 x 坐标。

PageY - PDF 页面内的 y 坐标。

返回值：

无

- **GotoPageDest**

跳转到 PDF 文档中一个指定的位置。

函数原型：

Void GotoPageDest (ILink_Dest * link_dest)

参数：

link_dest - 通过 OnHyperLink 事件获取的一个 ILink_Dest 实例。

返回值：

无

6) 查看

- **ShowDocumentInfoDialog**

弹出文档的属性对话框。

函数原型：

void ShowDocumentInfoDialog()

参数：

无

返回值：

无

- **SetPDFMeasureUnit**

设置 PDF 文档的测量单位。

函数原型：

BOOL SetPDFMeasureUnit(short nType)

参数：

nType - 指定测量的单位，有效值为：

0 = 点

1 = 英寸

2 = 厘米

3 = 像素

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- **GetPageWidth**

获取指定的 PDF 页面的宽度。

函数原型：

float GetPageWidth(short nPageIndex)

参数：

nPageIndex - PDF 页面的索引值。

返回值：

返回可显示区域的页面宽度（以点为单位），1 点= 1/72 英寸 (约等于 0.3528 mm)。

- **GetPageHeight**

获取指定 PDF 页面的高度。

函数原型：

float GetPageHeight(short nPageIndex)

参数：

nPageIndex - PDF 页面的索引值。

返回值：

返回可显示区域的页面高度（以点为单位），1 点= 1/72 英寸 (约等于 0.3528 mm)。

- **CountTools**

获取当前 ActiveX 版本中可使用工具的数量。

函数原型：

short CountTools()

参数：

无

返回值：

返回当前 ActiveX 版本可使用工具的数量。

- **GetToolByIndex**

指定索引值，获取对应 ActiveX 工具的名称。

函数原型：

BSTR GetToolByIndex (short nIndex)

参数：

nIndex - 当前 ActiveX 版本工具的索引值。

返回值：

返回指定的 ActiveX 工具的名称。

- **ScrollView**

将当前视图在水平方向和垂直方向上进行 dx 和 dy 距离的滚动，采用的是设备坐标。

函数原型：

Void ScrollView(long dx, long dy)

参数：

dx - 水平方向的滚动的距离。

dy - 垂直方向的滚动的距离。

返回值：

无

- **GetScrollLocation**

获取显示区域的滚动条当前的位置。

函数原型：

void GetScrollLocation (long *dx, long *dy)

参数：

dx - [返回值]水平滚动条的位置。

dy - [返回值]垂直滚动条的位置。

返回值：

无

- **GetScrollLocationEx**

GetScrollLocation 方法的扩展。

函数原型：

void GetScrollLocationEx(VARIANT * HPos, VARIANT * VPos)

参数：

HPos - [返回值]水平滚动条的位置。

VPos - [返回值]垂直滚动条的位置。

返回值：

无

- **SetViewRect**

显示 PDF 页面的指定区域内容。

函数原型：

Void SetViewRect (float Left, float Top, float Width, float Height)

参数：

Left - 左上角 x 坐标，PDF 坐标系。

Top - 左上角 y 坐标，PDF 坐标系。

Width - 指定区域的宽度。

Height - 指定区域的高度。

返回值：

无

备注：

这里的坐标指的是 PDF 坐标，而非设备坐标；长度单位是 PDF 的点。该函数不会改变 ActiveX 控件窗口的位置和尺寸大小，而是通过改变当前 PDF 页面的显示位置和缩放比率，以保证设定的区域内的 PDF 内容能够充满整个 ActiveX 控件窗口。

一个典型的应用场景是: 最终用户通过鼠标点击和拖拽划定一个区域后，松开鼠标，应用程序首先调用 ConvertClientCoodToPageCood 接口将鼠标坐标转换为 PDF 坐标 然后调用 SetViewRect 接口将之前选中的区域内的 PDF 页面内容充满整个显示窗口。

- **ConvertClientCoordToPageCoord**

将 ActiveX 控件的窗口坐标转换为 PDF 页面坐标。

函数原型：

BOOL ConvertClientCoordToPageCoord (long nClientX, long nClientY, long* pnPageIndex, float* pPageX, float* pPageY)

参数：

nClientX - ActiveX 控件的窗口坐标，x 坐标，单位是像素。

nClientY - ActiveX 控件的窗口坐标，y 坐标，单位是像素。

pnPageIndex - [返回值]保存指定的坐标点转换成 PDF 页面坐标后， 所在的 PDF 页面索引值。

pPageX - [返回值]保存 PDF 页面坐标的 x 坐标。

pPageY - [返回值]保存 PDF 页面坐标的 y 坐标。

返回值：

转换成功则返回 TRUE，反之返回 FALSE。

给定的区域不仅包括当前正在显示的 PDF 页面，也可能会包括灰色的页面背景区。如果坐标点落在灰色的页面背景区，则转换失败。

- **ConvertClientCoordToPageCoordEx**

ConvertClientCoordToPageCoord 函数的扩展。

函数原型：

BOOL ConvertClientCoordToPageCoordEx (long nClientX, long nClientY, VARIANT* pnPageIndex, VARIANT* pPageX, VARIANT* pPageY)

参数：

nClientX - ActiveX 控件的窗口坐标，x 坐标，单位是像素。

nClientY - ActiveX 控件的窗口坐标，y 坐标，单位是像素。

pnPageIndex - [返回值]保存坐标点转换成 PDF 页面坐标后所在的 PDF 页面索引值。

pPageX - [返回值]保存 PDF 页面坐标的 x 坐标。

pPageY - [返回值]保存 PDF 页面坐标的 y 坐标。

返回值：

成功则返回 TRUE，反之返回 FALSE。

备注：

给定的区域不仅包括当前正在显示的 PDF 页面，也可能会包括灰色的页面背景区。如果坐标点落在灰色的页面背景区，则转换失败。

- **ConvertPageCoordToClientCoord**

将 PDF 页面坐标转换为 ActiveX 控件的窗口坐标。

函数原型：

BOOL ConvertPageCoordToClientCoord(long nPageIndex, float dPageX, float dPageY, long* pnClientX, long* pnClientY)

参数：

nPageIndex - PDF 页面索引值。

dPageX - PDF 页面坐标的 x 坐标。

dPageY - PDF 页面坐标的 y 坐标。

pnClientX - [返回值]保存 ActiveX 控件的窗口坐标的 x 坐标。如果该坐标超出了控件的窗口区域，则被视为无效的转换结果。

pnClientY - [返回值]保存 ActiveX 控件的窗口坐标的 y 坐标。如果该坐标超出了控件的窗口区域，则被视为无效的转换结果。

返回值：

转换成功则返回 TRUE，反之返回 FALSE。

如果当前没有打开文档或者页码错误，则返回值为 FALSE。

- **SetBackgroudColor**

设置 PDF 文档显示时的背景颜色。

函数原型：

boolean SetBackgroudColor(OLE_COLOR color)

参数：

color - PDF 文档显示的背景颜色。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

7) 超链接

- **EnableHyperLink**

启用超链接注释对象。

函数原型：

Void EnableHyperLink(BOOL bEnable)

参数：

bEnable - 值为 TRUE 则启用超链接，值为 FALSE 则禁用超链接。

返回值：

无

- **GetHyperLinkInfo**

获取指定超链接注释对象的信息。

函数原型：

BOOL GetHyperLinkInfo(short nPageIndex, short nIndex, BSTR* linktype,BSTR* linkdata, LPDISPATCH* linkdest)

参数：

nPageIndex - PDF 页面索引值。

nLinkIndex - 超链接的索的引值。

linktype - [返回值]超链接的类型。

linkdata - [返回值]超链接的数据。

linkdest - [返回值]超链接重定向的目的地。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

备注：

目前，仅支持获取 GoTo、GoToR、Lanuch 和 URI 类型链接的信息。

- ***CountHyperLinks**

统计指定 PDF 页面所包含的超链接的数量。

函数原型：

short CountHyperLinks(short nPageIndex)

参数：

nPageIndex - PDF 页面索引值。

返回值：

成功则返回超链接的数量。

返回值为 0 表示指定页面不包含任何超链接；

返回值为-1 表示函数执行失败。

- ***HighlightHyperLink**

高亮显示指定的一个超链接对象。

函数原型：

void HighlightHyperLink(short nPageIndex, short nLinkIndex)

参数：

nPageIndex - PDF 页面的索引值。

nLinkIndex - 超链接对象的索引值。

返回值：

无

- ***GetHyperLinkRect**

获取指定超链接的位置。

函数原型：

BOOL GetHyperLinkRect(short nPageIndex, short nIndex, float* top, float* left, float* bottom, float* right)

参数：

nPageIndex - PDF 页面的索引值。

nLinkIndex - 超链接的索引值。

top - 顶部坐标返回的指针。

left - 左边坐标返回的指针。

bottom - 底部坐标返回的指针。

right - 右边坐标返回的指针。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

8) 文本操作

- **GetPageText**

从当前加载的 PDF 文件中提取指定页的文本内容。

函数原型：

BSTR GetPageText (long nPageIndex)

参数：

nPageIndex - PDF 页面的索引值。

返回值：

返回提取的 PDF 文本。

- ***GetPageTextW**

从当前加载的 PDF 文件中提取指定页的文本内容。

函数原型：

long GetPageTextW(long nPageIndex, long FAR* pBuffer, long FAR* nBuflen)

参数：

nPageIndex - PDF 页面的索引值。

pBuffer - 接收内容的缓存。

nBuflen - 提取的页面内容的长度。

返回值：:

成功则返回 TRUE，反之则返回 FALSE。

- **GetSelectedRectsCount**

获取文本选中区域的数量。

函数原型：

long GetSelectedRectsCount()

参数：

无

返回值：

返回文本选中区域的总数量。

- **GetSelectedRectByIndex**

获取指定索引的选择区域。

函数原型：

long GetSelectedRectByIndex(long nIndex, long* nPageIndex, float* left, float* bottom, float* right, float* top)

参数：

nIndex - 当前选定区域的索引值，值从 0 开始。

nPageIndex - 选定区域所在的 PDF 页面的索引值。

Left - 选定区域左边的坐标。

Bottom - 选定区域底端的坐标。

Right - 选定区域右边的坐标。

Top - 选定区域顶端的坐标。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- **GetSelectedText**

获取当前选中的文本。

函数原型:

BSTR GetSelectedText()

参数:

无

返回值:

返回当前选中的文本内容。

- **GetSelectedTextEx**

GetSelectedText 函数的扩展。

函数原型:

long GetSelectedTextEx(long* pBuffer, long nBuflen)

参数:

pBuffer - 接收选定文本内容的缓存。

nBufLen - 选定文本内容的长度。

返回值:

返回选定文本内容的长度。

- ***LoadPDFTextDoc**

从 PDF 文件中提取一个文本文档。

函数原型：

IPDFTextDoc* LoadPDFTextDoc(BSTR filePath, BSTR filePass)

参数：

filePath - PDF 文件的完整路径；该路径可以是一个 URL 地址。

filePass - PDF 文件的打开密码。

返回值：

返回一个 IPDFTextDoc 对象。

- ***ExtractTextFromPDF**

将一个 PDF 文件转换为 txt 文件。

函数原型：

BOOL ExtractTextFromPDF(BSTR sourcePath, BSTR sourcePass, BSTR destPath)

参数：

sourcePath - PDF 文件的完整路径。

sourcePass - PDF 文件的打开密码。

destPath - 要保存的 txt 文件路径。如果该文件不存在，将在指定的位置创建一个新

的文件。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

9) 自定义用户界面

- **ShowTitleBar**

显示或隐藏标题栏。

函数原型：

Void ShowTitleBar(BOOL bShow)

参数：

bShow - 该值表示是否显示标题栏。

值为 TRUE，则显示标题栏；

值为 FALSE，则隐藏标题栏。

返回值：

无

- **ShowStatusBar**

显示或隐藏状态栏。

函数原型：

Void ShowStatusBar(BOOL bShow)

参数：

bShow - 该值表示是否显示状态栏。

值为 TRUE，则显示状态栏；

值为 FALSE，则隐藏状态栏。

返回值：

无

- **ShowToolBar**

显示或隐藏工具栏。

函数原型：

Void ShowToolBar(BOOL bShow)

参数：

bShow - 该值表示是否显示工具栏。

值为 TRUE，则显示工具栏；

值为 FALSE，则隐藏工具栏。

返回值：

无

- **ShowToolbarButton**

显示或隐藏工具栏上的按钮。

函数原型：

Void ShowToolBarButton(short nIndex, BOOL bShow)

参数：

nIndex - 按钮的索引值。

bShow - 该值表示是否显示工具栏上的按钮。

值为 TRUE，则显示该按钮；

值为 FALSE，则隐藏该按钮。

返回值：

无

- **EnableToolTip**

显示或隐藏工具的提示信息。

函数原型：

BOOL EnableToolTip (BOOL bEnable)

参数：

bEnable - 该值表示是否显示工具的提示消息。

值为 TRUE，则显示提示信息；

值为 FALSE，则隐藏提示信息。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- **GetPanelStatus**

获取整个导航栏的显示状态。

函数原型：

BOOL GetPanelStatus()

参数：

无

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- **ShowNavigationPanels**

显示或隐藏导航栏面板。

函数原型：

BOOL ShowNavigationPanels(BOOL bShow)

参数：

bShow - 该值表示是否显示导航栏面板。

值为 TRUE，则显示导航栏；

值为 FALSE，则隐藏导航栏。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- **ShowNavPanelByString**

显示指定名称的导航栏面板。

函数原型：

BOOL ShowNavPanelByString(LPCTSTR lpszPanelName)

参数：

lpszPanelName - 导航栏面板，有效值为：

"Bookmarks"，

"Pages"，

"Layer"，

"Attachments ”

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- **GetLayoutShowMode**

获取页面显示的布局模式。

函数原型：

Void GetLayoutShowMode(short* pnShowMode, short* pnFacingCount)

参数：

pnShowMode - [返回值]当前的模式。

pnFacingCount - [返回值]当前水平方向上显示页面的数量。

返回值：

无

- **SetLayoutShowMode**

设置页面显示的布局模式。

函数原型：

Void SetLayoutShowMode(BrowseMode nShowMode, short nFacingCount)

参数：

nShowMode - 可以设置如下的值:

0 = MODE_SINGLE

1 = MODE_CONTINUOUS

nFacingCount - 每行显示的列数。

返回值：

无

备注：

- 在显示区域，可按照 N*M 的方式显示 PDF 文档。
- 当显示布局的模式设定成 MODE_SINGLE 时 ,在 ActiveX 控件的显示区域每行将只显示一页，nFacingCount 的值将不起作用。
- 当显示布局的模式设定成 MODE_CONTINUOUS 时， 在 ActiveX 控件的显示区域每行将可以显示多页。

- **SetFacingCoverLeft**

在双页显示模式下，调用该函数将决定 PDF 文档的封面是否显示在左侧。

函数原型：

Void SetFacingCoverLeft (BOOL bLeft)

参数：

bLeft - 该值表示是否将封面显示在左侧。

返回值：

无

- **SetDisplayBackgroundColor**

设置 ActiveX 控件的背景颜色。

函数原型：

void SetDisplayBackgroundColor(long clrArgb)

参数：

clrArgb： ActiveX 控件的背景颜色。要求输入十进制的 ARGB 值。

返回值：

无。

- **SetContextMenuString**

设置右键菜单项名称。

函数原型：

void SetContextMenuString(BSTR string)

参数：

string - 右键菜单项名称。

返回值：

无

- ***ShowFormFieldsMessageBar**

显示或隐藏表单域信息提示栏。

函数原型：

Void ShowFormFieldsMessageBar(BOOL bShow)

参数：

bShow - 该值表示是否显示表单信息提示栏。

值为 TRUE，则显示信息提示栏；

值为 FALSE，则隐藏信息提示栏。

返回值：

无

- ***ShowContextMenu**

显示右键菜单。

函数原型：

void ShowContextMenu(BOOL bShow)

参数：

bShow - 设置是否显示右键菜单。

值为 TRUE，则显示右键菜单；

值为 FALSE，则隐藏右键菜单。

返回值：

无

- ***SetSigContextMenuString**

设置签名域的右键弹出菜单。

函数原型：

Void SetSigContextMenuString(BSTR string);

参数：

string - 右键菜单项。如果是多个菜单项，则通过逗号进行分隔。

返回值：

无。

- ***EnableBookmarkEdit**

设置书签是否可以编辑。

函数原型：

void EnableBookmarkEdit(boolean bEdit)

参数：

bEdit - 该值表示书签是否可以编辑。

返回值：

无

- **ShowStdBookmarkContextMenu**

设置是否显示书签的默认菜单项。

函数原型：

void ShowStdBookmarkContextMenu(short nIndex, boolean bShow)

参数：

nIndex - 书签的默认菜单项的索引。索引值从 1 开始。

bShow - 该值表示是否显示菜单项。

返回值：

无

- **SetBookmarkContextMenuString**

设置书签的右键菜单。

函数原型：

void SetBookmarkContextMenuString(BSTR string)

参数：

string - 书签的右键菜单，多个菜单项使用逗号分隔。

返回值：

无

- **ShowStdPagesContextMenu**

设置是否显示页面缩略图的默认菜单项。

函数原型：

void ShowStdPagesContextMenu(short nIndex, boolean bShow)

参数：

nIndex - 页面缩略图的默认菜单项的索引。索引值从 1 开始。

bShow - 是否显示菜单项。

返回值：

无

- **SetPagesContextMenuString**

设置页面缩略图的右键菜单。

函数原型：

void SetPagesContextMenuString(BSTR string)

参数：

string - 页面缩略图的右键菜单，多个菜单项使用逗号分隔。

返回值：

无

10) PDF 打印

- **PrintWithDialog**

弹出打印对话框，进行 PDF 文件的打印。

函数原型：

Void PrintWithDialog()

参数：

无

返回值：

无

- **OpenMemFileForPrinter**

直接打印内存中的 PDF 文件，而无需显示该 PDF 文件。

函数原型：

IPDFPrinter* OpenMemFileForPrinter(long buffer, long size)

参数：

Buffer - 内存中 PDF 文件的数据。

Size - 缓存中数据的长度。

返回值：

函数返回一个 IPDFPrinter 类的实例，以便对打印机进行控制。

- **OpenFileForPrinter**

直接打开一个 PDF 文件并进行打印，但不显示该 PDF 文件。

函数原型：

IPDFPrinter*OpenFileForPrinter(BSTR file_path)

参数：

file_path - 将打印的 PDF 文件的完整路径。

返回值：

函数返回一个 IPDFPrinter 类的实例，以便对打印机进行控制。

- ***PrintPopupAnnot**

设置是否打印弹出式注释对象的内容。

函数原型：

void PrintPopupAnnot(boolean bPrint)

参数：

bPrint - 该值表示是否打印弹出式注释的内容。

返回值：

无

- ***AddPrintMark**

添加文本水印到 PDF 文档中进行打印。

函数原型：

boolean AddPrintMark(BSTR string, float center_x, float center_y, BSTR fontname, short lfCharSet, short fontsize, OLE_COLOR fontcolor, short textmode, short alpha, short rotate)

参数：

string： - 要添加的文本水印的内容。

center_x - PDF 坐标系的横坐标。

center_y - PDF 坐标系的纵坐标。

FontName - 文本水印的字体。

lfCharSet - 文本水印的字符集。可以使用以下预定义的值：

ANSI_CHARSET、

BALTIC_CHARSET、

CHINESEBIG5_CHARSET、

DEFAULT_CHARSET（表示字符集基于本地操作系统，例如，系统位置是 English (United States),字符集将设置为 ANSI_CHARSET。）

EASTEUROPE_CHARSET、

GB2312_CHARSET、

GREEK_CHARSET、

HANGUL_CHARSET、

MAC_CHARSET、

OEM_CHARSET（表示字符集依赖本地操作系统）

RUSSIAN_CHARSET、

SHIFTJIS_CHARSET、

SYMBOL_CHARSET、

TURKISH_CHARSET。

FontSize - 文本水印的字号。

FontColor - 文本水印的字体颜色。

textmode - 文本水印的填充模式。

0 = 字体填充

1 = 描边

2 = 填充和描边

alpha - 文本水印的透明度，值为 0 到 255。

Rotate - 旋转角度。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

11) 文本搜索

- **FindFirst**

在当前 PDF 文件中进行文本搜索，获取第一个搜索结果。

函数原型：

BOOL FindFirst (BSTR search_string, BOOL bMatchCase, BOOL bMatchWholeWord)

参数：

SearchString - 要搜索的文本字符串。

BMatchCase - 设置搜索是否区分大小写。

BMatchWholeWord - 设置搜索是否全词匹配。

返回值：

成功找到搜索结果，则返回 TRUE，反之则返回 FALSE。

备注：

如果 PDF 文件中存在搜索结果，则跳转到那个页面，并自动更新 CurPage 属性值，高亮页面上的搜索结果。

- **FindFirstEx**

在当前 PDF 文件中进行文本搜索，获取第一个搜索结果。FindFirst 方法的扩展。

函数原型：

BOOL FindFirstEx(const VARIANT FAR& search_string, BOOL bMatchCase, BOOL bMatchWholeWord)

参数：

Search_string - 要搜索的文本字符串。

bMatchCase - 设置搜索是否区分大小写。

bMatchWholeWord - 设置搜索是否全词匹配。

返回值：

成功找到搜索结果，则返回 TRUE，反之则返回 FALSE。

- **FindNext**

在 PDF 文件中获取下一个搜索结果，文本搜索由 FindFirst 指定。

函数原型：

BOOL FindNext (BOOL bSearchDown)

参数：

bSearchDown - 设置文本搜索的方向。设置为 TRUE，向文档尾方向进行搜索；设置为 FALSE，向文档头方向进行搜索。

返回值：

成功找到搜索结果，则返回 TRUE，反之则返回 FALSE。

备注：

如果找到下一个，则直接跳转到那个页面，自动更新 CurPage 属性值，并高亮页面中匹配的内容。

- **FindPageFirst**

在当前 PDF 页面上进行文本搜索，获取第一个搜索结果。

函数原型：

IFindResult*FindPageFirst(long nPageIndex, BSTR search_string,BOOL bMatchCase, BOOL bMatchWholeWord)

参数：

nPageIndex	-	PDF 页面索引值，起始页为 0。
Search_string	-	要搜索的文本字符串。
bMatchCase	-	设置搜索是否区分大小写。
bMatchWholeWord	-	设置搜索是否全词匹配。

返回值：

成功获取搜索结果，则返回一个 IFindResult 接口类实例，否则返回 NULL。

- **FindPageNext**

在当前 PDF 页面上获取下一个搜索结果，文本搜索由 FindPageFirst 指定。

函数原型：

IFindResult* FindPageNext()

参数：

无

返回值：

成功获取搜索结果，则返回一个 IFindResult 接口类实例，否则返回 NULL。

备注：

首次搜索时调用 FindPageFirst 函数，然后调用 FindPageNext 获取下一个搜索结果。

- **FindFileFirst**

在指定的 PDF 文件中进行文本搜索，获取第一个搜索结果。

函数原型：

IFindResult* FindFileFirst(BSTR file_path, BSTR search_string,BOOL bMatchCase, BOOL bMatchWholeWord)

参数：

file_path	-	进行搜索的 PDF 文件的完整路径。
search_string	-	要搜索的文本字符串。
bmatchCase	-	设置搜索是否区分大小写。
BMatchWholeWord	-	设置搜索是否全词匹配。

返回值：

成功获取搜索结果，则返回一个 IFindResult 接口类实例，否则返回 NULL。

备注：

该方法允许在不打开文件的情况下直接进行搜索。

例如，想要在一个目录中对所有的 PDF 文件进行关键字搜索，只需循环遍历目录中的每一个 PDF 文件，对每一个文件的内容进行关键字搜索即可。

如果在一个 PDF 文件中找到与之相匹配的内容 ,函数的返回值是一个包含相关细节的 IFindResult 接口类的示例。然后，调用 GoToSearchResult 打开文件，自动跳转到相应的页面并高亮页面中匹配的内容。

- **FindFileFirstEx**

在指定的 PDF 文件中进行文本搜索，获取第一个搜索结果。对 FindFileFirst 方法的扩展。

函数原型：

IFindResult * FindFileFirstEx(BSTR file_path, BSTR password, VARIANT search_string, boolean bMatchCase, boolean bMatchWholeWord)

参数：

file_path	-	PDF 文件的完整路径
Password	-	PDF 文件的打开密码。
search_string	-	要搜索的文本字符串。
bMatchCase	-	设置搜索是否区分大小写。
BMatchWholeWord	-	设置搜索是否全词匹配。

返回值：

成功获取搜索结果，则返回一个 IFindResult 接口类实例，否则返回 NULL。

- **FindFileNext**

获取下一个搜索结果，文本搜索由 FindFileFirst 指定。

函数原型：

IFindResult * FindFileNext()

参数：

无

返回值：

成功获取搜索结果，则返回一个 IFindResult 接口类实例，否则返回 NULL。

- **GoToSearchResult**

显示并高亮搜索结果。

函数原型：

Void GoToSearchResult (IFindResult* findresult)

参数：

Findresult - 由 FindFileFirst 或 FindFileNext 返回的 IFindResult 接口类的实例。

返回值：

无

- **FindMemFileFirst**

在内存当中查找文件。

函数原型：

IFindResult*FindMemFileFirst(VARIANT buffer, long fileSize, BSTR password, BSTR search_string, BOOL bMatchCase, BOOL bMatchWholeWord)

参数：

Buffer - 指向包含 PDF 文件数据的缓冲区。
fileSize - PDF 的文件大小。
Password - PDF 的文件打开密码。
Search_string - 要搜索的文本字符串。
bMatchCase - 设置搜索是否区分大小写。
bMatchWholeWord - 设置搜索是否全词匹配。

返回值：

成功获取搜索结果，则返回一个 IFindResult 接口类实例，否则返回 NULL。

- **FindClose**

释放 FindMemFileFirst 当中分配的内存。

函数原型：

Void FindClose()

参数：

无

返回值：

无

- ***SearchAndHighlightAllTextOnPage**

在指定的 PDF 页面上搜索文本，并高亮所有符合条件的结果。

函数原型：

Void SearchAndHighlightAllTextOnPage(BSTR searchstring, BOOL bMatchCase, BOOL bMatchWholeWord,long PageNo)

参数：

PageNo - 要搜索的 PDF 页面索引值。

返回值：

无

- ***SetSearchHighlightFillColor**

设置 PDF 文本搜索的高亮颜色。

函数原型：

void SetSearchHighlightFillColor(OLE_COLOR FillColor, short Alpha)

参数：

FillColor - 文本搜索结果的高亮颜色。
Alpha - 文本搜索结果的高亮透明度，值为 0 到 255。

返回值：

无

12) 文档安全

- **GetDocPermissions**

  获取 PDF 文件的权限。

函数原型：

  long GetDocPermissions()

参数：

  无

返回值：

  返回一个 32 位的整型数表示 PDF 文件的权限信息，详情请参照 PDF 规范。

  返回值为 0xffffffff，则表示 PDF 文件不存在权限控制。

- ***SetUserPermission**

  设置当前 PDF 文件的权限。

函数原型：

  BOOL SetUserPermission(long dwPermission)

参数：

  dwPermission - 权限控制标识。

返回值：

  成功则返回 TRUE，反之则返回 FALSE。

- ***** **CheckOwnerPassword**

  判断文档的所有者密码（文档修改密码）是否正确。

函数原型：

  BOOL CheckOwnerPassword(BSTR lpszPermPsw)

参数：

  lpszPermPsw - PDF 文件的所有者密码（文档修改密码）。

返回值：

  成功则返回 TRUE，反之则返回 FALSE。

- ***SetOwnerPassword**

  设置当前的 PDF 文件的所有者密码（文档修改密码）。

函数原型：

  BOOL SetOwnerPassword(LPCTSTR lpszNewValue)

参数：

  lpszNewValue - PDF 文件的所有者密码（文档修改密码）字符串。

返回值：

  成功则返回 TRUE，反之则返回 FALSE。

- ***SetUserPassword**

设置当前的 PDF 文件的打开密码。

函数原型：

BOOL SetUserPassword(LPCTSTR lpszNewValue)

参数：

lpszNewValue - PDF 文件的打开密码字符串。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

13) JS 脚本

- *** RunJScript**

运行 JavaScritp 脚本。

函数原型：

void RunJScript(LPCTSTR csJS)

参数：

scJS - JavaScript 脚本。

返回值：

无

- ***ShowDocJsDialog**

弹出文档 Javascript 对话框。

函数原型：

Void ShowDocJsDialog()

参数：

无

返回值：

无

- ***ShowJsConsoleDialog**

弹出 Javascript 控制台对话框。

函数原型：

Void ShowJsConsoleDialog()

参数：

无

返回值：

无

14) 多实例

- **SetCurrentWnd**

当 ActiveX 在多实例情况下，设置当前的实例。

函数原型：

Void SetCurrentWnd(long hWnd)

参数：

hWnd - OCX 实例的窗口句柄。

返回值：

无

- **GetCurrentWnd**

获取当前窗口的句柄。

函数原型：

Long GetCurrentWnd()

参数：

无

返回值：

成功则返回当前窗口句柄，反之则返回 NULL。

- **GetCtrlInstance**

获取控件实例句柄。

函数原型：

Long GetCtrlInstance()

参数：

无

返回值：

成功则返回控件实例句柄，反之则返回 NULL。

15) 页面管理与编辑

- **GetPageRotation**

获取指定的当前 PDF 页面的旋转度数。

函数原型:

long GetPageRotation(long page_index)

参数:

page_index - PDF 页面的索引值。

返回值:

返回值分别代表了 4 个旋转度数：0, 90, 180 和 270。

0 - 0

1 - 90

2 - 180

3 - 270

- *** SetPageRotate**

旋转当前 PDF 文件中的指定页面。

函数原型：

Boolean SetPageRotate(long pageIndex, long rotate)

参数：

pageIndex - 指定旋转的 PDF 页的页码

rotate - 设置顺时针旋转的角度

返回值：

成功则返回 True，反之则返回 False。

- ***InsertNewPage**

在当前 PDF 文件的指定位置插入空白页。

函数原型：

Boolean InsertNewPage(long nPageIndex, long nPageWidth, long nPageHeight)

参数：

nPageIndex - 待插入的新空白 PDF 页的页码。

nPageWidth - 待插入的新空白 PDF 页的页面宽度。

nPageHeight - 待插入的新空白 PDF 页的页面高度。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- ***InsertPage**

在当前 PDF 文件的指定页面前插入 PDF 页面。

函数原型：

boolean InsertPage(long nInsertAt, BSTR lpszPDFFileName, BSTR lpszPssword, BSTR lpszPageRangeString)

参数：

nInsertAt - 待插入的第一个 PDF 页面的索引值。

lpszPDFFileName - 待插入页面所在的源 PDF 文件的路径。

lpszPssword - 源 PDF 文件的密码。

lpszPageRangeString - 待插入页面在源 PDF 文件中的页码范围。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- ***DeletePage**

删除当前 PDF 文件中的指定页面。

函数原型：

boolean DeletePage(long pageIndex, long count)

参数：

pageIndex - 待删除页面的起始页的索引值。

count - 待删除页面的总页数

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- ***SwapPage**

交换当前 PDF 文件的两个页面的位置。

函数原型：

boolean SwapPage(long pageIndex1, long pageIndex2)

参数：

pageIndex1 - 进行交换的页面 1 的索引值。

pageIndex2 - 进行交换的页面 2 的索引值。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- ***SetPageIndex**

通过设置新的页面索引值，改变当前 PDF 文件中指定页面的位置。

函数原型：

Boolean SetPageIndex (long pageOldIndex, long pageNewIndex)

参数：

pageOldIndex - 原来的 PDF 页面索引值。

pageNewIndex - 新的 PDF 页面索引值。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- ***SetPageCropBox**

设置剪切区域，裁剪当前 PDF 文件中指定的页面。

函数原型：

boolean SetPageCropBox(long pageIndex, float left, float top, float right, float bottom)

参数：

pageIndex - 进行裁剪的 PDF 页的索引值。

left - 页面左上角的水平坐标。

top - 页面左上角的垂直坐标。

right - 页面右下角的水平坐标。

bottom - 页面右下角的垂直坐标。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- ***FlattenPage**

将当前 PDF 文件的指定页面扁平化。

函数原型：

boolean FlattenPage(long pageIndex)

参数：

pageIndex - PDF 页面的索引值。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- ***ExportPagesToPDF**

将当前 PDF 文档中的指定页面导出为一个新的 PDF 文件。

函数原型：

BOOL ExportPagesToPDF (BSTR lpszPDFFileName, BSTR lpszPageRangeString)

参数：

lpszPDFFileName - 新的 PDF 文件的完整路径。

lpszPageRangeString - 待导出的 PDF 页面。格式如 "0, 2, 3-5"，不能指定为逆序，例如 "5-2"。也就是说，破折号前面的值不能大于破折号后面的值。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- ***ImportImageToPdf**

在指定的 PDF 文件中导入图片。

函数原型：

BOOL ImportImageToPdf(LPCTSTR pdfFilePath, long index, LPCTSTR imageFilePath)

参数：

pdfFilePath - PDF 文件的路径。

index - 待插入图片的 PDF 页面的索引值。

imageFilePath - 图片文件的路径。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- ***AddImageObject**

将图片插入到文档中。

函数原型：

BOOL AddImageObject (long nPageIndex, float left, float bottom, float width, float height, BSTR BmpFileName, short alpha, short rotate)

参数：

nPageIndex - 待插入图片的 PDF 页面索引值。
left - 待插入图片的水平坐标，坐标原点是左下角。
bottom - 待插入图片的垂直坐标，坐标原点是左下角。
width - 图片在 PDF 页面中的的显示宽度。
height - 图片在 PDF 页面中的显示高度。
BmpFileName - 待插入的源图片的路径。
alpha - 一个 0 到 255 之间的数字，表示图片显示的透明度。
rotate - 图片的旋转度数。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- ***ConvertPDFPageToImage**

将指定 PDF 页面转换成 JPG 图片。

函数原型：

BOOL ConvertPDFPageToImage(LPCTSTR pdfFile, long lnPageIndex, LPCTSTR imageFilePath, long lnImageWidth, long lnImageHeight)

参数：

pdfFile - 源 PDF 文档的路径。
lnPageIndex - 进行转换的 PDF 页面的索引值。
imageFilePath - 生成的图像文件的保存路径。
lnImageWidth - 转换生成的图片宽度。
lnImageHeight - 转换生成的图片高度。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

16) 异步下载

- **OpenFileAsync**

通过异步方式打开 PDF 文件。主要用于打开线性化 PDF 文件，可在 B/S 构架中快速下载并渲染 PDF 数据。

函数原型：

BOOL OpenFileAsync(LPCTSTR strURL, LPCTSTR strPDFPassword, LPCTSTR strUserName, LPCTSTR strUserPassword)

参数：

strURL - PDF 文档的 URL 下载地址；支持 FTP 和 HTTP 类型。
strPDFPassword - PDF 文档的打开密码；如果无密码，则值为空。
strUserName - FTP 服务器的登陆用户名；如果不需要，则值为空。
strUserPassword - FTP 服务器的登陆密码；如果不需要，则值为空。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- ***SetAsyncFileLen**

设置通过异步方式加载的 HTTP 服务器上的 PDF 文件长度。

函数原型：

Void SetAsyncFileLen (long lnSize)

参数：

lnSize - PDF 文件长度。

返回值：

无。

备注：

*SetAsyncFileLen 必须和*OpenAsyncFile，*SetAsyncFileData 以及*OnFetchAsyncFileData 配合调用。

- ***OpenAsyncFile**

开始通过异步方式打开 HTTP 服务器上的文件。

函数原型：

Void OpenAsyncFile(LPCTSTR strPDFPassword)

参数：

strPDFPassword - PDF 文件的打开密码。

返回值：

无。

备注：

*OpenAsyncFile 必须和*SetAsyncFileLen，*SetAsyncFileData 以及*OnFetchAsyncFileData 配合调用。

- ***SetAsyncFileData**

设置通过异步方式打开的 PDF 文档数据。

函数原型：

Void SetAsyncFileData(long lnFileBuffer, long offset, long size)

参数：

lnFileBuffer - PDF 文件数据的内存地址。

offset - 数据的偏移量。

Size - PDF 数据的大小。

返回值：

无。

备注：

*SetAsyncFileData 必须和*OpenAsyncFile，*SetAsyncFileLen 以及*OnFetchAsyncFileData

配合调用。

17) 书签

- **GetOutlineFirstChild**

获取大纲目录树中指定节点的第一个子节点。

函数原型：

IPDFOutline*GetOutlineFirstChild(IPDFOutline* Outline)

参数：

Outline - 大纲目录树中的一个节点。如果想获取大纲目录树的根节点，则值为 NULL。

返回值：

如果该节点存在子节点，则返回 IPDFOutline，反之则返回 NULL。

- **GetOutlineNextSibling**

获取大纲目录树中指定节点的下一个兄弟节点。

函数原型：

IPDFOutline* GetOutlineNextSibling (IPDFOutline* Outline)

参数：

Outline - 大纲目录树中的一个节点。

返回值：

如果该节点存在下一个兄弟节点，则返回 IPDFOutline，反之则返回 NULL。

18) 表单

- ***ExportFormToFDFFile**

将 PDF 表单数据从当前打开的文档中导出，并保存为一个独立的 FDF 文件。

函数原型：

BOOL ExportFormToFDFFile (BSTR FDFFileName)

参数：

FDFFileName - 导出的 PDF 表单数据形成的 FDF 文件的路径。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- ***ImportFormFromFDFFile**

将 PDF 表单数据从 FDF 文件导入到当前打开的文档中。

函数原型：

BOOL ImportFormFromFDFFile(BSTR FDFFileName)

参数：

FDFFileName - 要导入到当前 PDF 文档的源表单数据 FDF 文件路径。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- ***FindFormFieldsTextFirst**

在表单域中进行文本搜索，获取第一个搜索结果。

函数原型：

BOOL FindFormFieldsTextFirst (BSTR searchstring, BOOL bMatchCase)

参数：

Searchstring - 要搜索的文本字符串。

bMatchCase - 设置搜索是否区分大小写。

返回值：

成功找到搜索结果则返回 TRUE，反之则返回 FALSE。

- ***FindFormFieldsTextNext**

在表单域中继续获取搜索结果，文本搜索由 FindFormFieldsTextFirst 指定。

函数原型：

BOOL FindFormFieldsTextNext()

参数：

无

返回值：

成功找到搜索结果则返回 TRUE，反之则返回 FALSE。

- *** SubmitForm**

将 PDF 表单提交到指定的位置。

函数原型：

BOOL SubmitForm(BSTR csDestination)

参数：

csDestination - 指定的 URL 地址。

返回值：

提交成功则返回 TRUE，反之返回 FALSE。

- **#GetCurrentForm**

获取当前选中的表单对象。

函数原型：

IPDFForm* GetCurrentForm()

参数：

无

返回值：

获取成功则返回 IPDFForm，反之返回 NULL。

19) 注释

- **ForceRefresh**

完成高亮添加后，刷新页面。

函数原型：

Void ForceRefresh()

参数：

无

返回值：

无

备注：

该函数用于提高高亮显示的效率。在旧版本中，用户每添加一次高亮文本，页面就会刷新一次，若用户多次添加高亮文本，则页面将会多次刷新，这就降低了显示效率。从 5.0 版本开始，我们将高亮功能的添加和刷新动作进行分离 ,即在用户完成高亮添加后再调用 ForceRefresh 函数刷新页面。

- ***Highlight**

在当前 PDF 文件的指定页面，高亮选定的矩形区域。

函数原型：

void Highlight(long nPageIndex, float left, float top, float right, float bottom)

参数：

nPageIndex	-	待高亮区域所在的 PDF 页面的索引值。
left	-	指定矩形区域的左上角的水平坐标。
top	-	指定矩形区域的左上角的垂直坐标。
right	-	指定矩形区域的右下角的水平坐标。
bottom	-	指定矩形区域的右下角的垂直坐标。

返回值：

无

- ***RemoveAllHighlight**

移除当前打开的 PDF 文档中所有的高亮（注释对象）。

函数原型：

void RemoveAllHighlight()

参数：

无

返回值：

无

- ***ImportAnnotsFromFDFFile**

将注释数据从 FDF 文件导入到当前的 PDF 文件中。

函数原型：

BOOL ImportAnnotsFromFDFFile(BSTR FDFFileName)

参数：

FDFFileName - 进行批注数据导入的源 FDF 文件路径。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- ***ExportAnnotsToFDFFile**

将注释从当前打开的文档中导出，并保存为一个独立的 FDF 文件。

函数原型：

BOOL ExportAnnotsToFDFFile(BSTR FDFFileName)

参数：

FDFFileName - 导出的注释数据 FDF 文件的保存路径。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

- ***ImportAnnotsFromFDFFileEx**

将注释数据从 FDF 文件中导入到当前打开的一个 PDF 文档页中。

函数原型:

boolean ExportAnnotsToFDFFileEx(long PageIndex, BSTR FDFFileName)

参数:

PageIndex - PDF 文件指定页面的索引值，FDF 数据将会导入到此 PDF 页中。

FDFFileName - 注释数据源 FDF 的文件路径。

返回值:

成功则返回 TRUE，反之则返回 FALSE。

- *** ExportAnnotsToFDFFileEx**

将注释数据从当前打开的 PDF 文档的指定页面导出，并另存为一个独立的 FDF 文件。

函数原型:

boolean ExportAnnotsToFDFFileEx(long PageIndex, BSTR FDFFileName)

参数:

PageIndex - PDF 文件指定页面的索引值，将到处此页的注释数据。

FDFFileName - 导出的注释数据生成的 FDF 文件的完整路径。

返回值:

成功则返回 TRUE，反之则返回 FALSE。

- ***SetBDrawAnnot**

显示 PDF 页面上的注释。

函数原型：

Void SetBDrawAnnot(BOOL bDrawAnnot)

参数：

bDrawAnnot - 值为 TRUE，则显示注释。值为 FALSE，则隐藏注释。

返回值：

无

- ***ShowAllPopup**

显示所有注释的弹出框。

函数原型：

Void ShowAllPopup (BOOL bShow)

参数：

bShow - 值为 TRUE，则显示注释的弹出框。值为 FALSE，则隐藏注释的弹出框。

返回值：

无

- ***ReleaseAnnot**

释放 GetAnnot 返回的对象。

函数原型：

void ReleaseAnnot(IPDFAnnot* Annot)

参数：

pageAnnots - 注释对象。

返回值：

无

- **^GetPageAnnots**

获取当前 PDF 文件中的指定页面上的注释信息。

函数原型：

IPDFPageAnnots GetPageAnnots(long pageIndex)

参数：

pageIndex - PDF 页面的索引值。

返回值：

成功则返回 IPDFPageAnnots，该对象中包含页面所有的注释信息；反之则返回 NULL。

- **^GetFormatTool**

获取 FormatTool 的句柄。

函数原型：

IPDFormatTool GetFormatTool ()

参数：

无

返回值：

成功则返回 FormatTool 的句柄，失败则返回 NULL。

20) 电子签章

- **&GetPDFSignatureMgr**

获取 PDF 文档中的签章。

函数原型：

IPDFSignatureMgr* GetPDFSignatureMgr()

参数：

无

返回值：

返回指向 PDFSignatureMgr 对象的指针。

- **&GetLastSigModuleError**

获取签章接口最后返回的错误码。

函数原型：

long GetLastSigModuleError()

参数：

无

返回值：

签章接口最后返回的错误码。

0 = 成功
1 = 参数错误
2 = 状态错误
3 = 执行错误
4 = 证书不存在

- **&GetLastSigModuleErrMsg**

获取签章接口最后返回的错误提示。

函数原型：

BSTR GetLastSigModuleErrMsg()

参数：

无

返回值：

返回签章接口最后返回的错误提示。

21) 其他

- ***UploadCurFileToFTP**

将当前的 PDF 文件上传到一个 FTP 服务器。

函数原型：

BOOL UploadCurFileToFTP(BSTR ftpName, BSTR userName, BSTR userPassword, long port, BSTR FilePath)

参数：

ftpName	-	FTP 服务器名称
userName	-	FTP 服务器登录用户名
userPassword	-	FTP 服务器登录密码
port	-	FTP 服务器端口
filePath	-	FTP 文件路径

返回值：

成功则返回一个非零值，反之返回值为零。

- ***IsDualPage**

判断 PDF 文档的页面是否是双层的。

函数原型：

BOOL IsDualPage(short pageIndex);

参数：

pageIndex - PDF 页面索引值。

返回值：

返回值指明该 PDF 页面是否是双层的。

备注：

双层的意思是 PDF 页面包含一个图片和与图片对应的隐藏的文本。

- ***AddWaterMark**

将一个文本型水印插入到文档中。

函数原型：

BOOL AddWaterMark (short page, BSTR string, float left, float bottom, short fontsize, OLE_COLOR fontcolor, short textmode, short alpha, short rotate)

参数：

Page	-	要插入水印的页码
String	-	水印的内容
Left	-	插入水印的水平坐标
Bottom	-	插入水印的垂直坐标
Fontsize	-	水印文本的字体大小
Fontcolor	-	水印文本的字体颜色
Textmode	-	文本的填充模式，有效值为 0、1、2

0 = 表示填充文本

1 = 表示绘制文本边框

2 = 表示同时绘制边框和填充文本

alpha - 透明度，有效值为 0~255。

rotate - 水印旋转的角度。

返回值：

成功则返回 TRUE，否则返回 FALSE。

- ***GetBitmap**

将 PDF 的页面渲染为一张图片。

函数原型：

long GetBitmap(short nPageIndex, long pixelWidth, long pixelHeight, float rectLeft, float rectTop, loat rectRight, float rectBottom, long PixelFormat)

参数：

nPageIndex - 指定进行渲染的页面页码。

pixelWidth - 形成的图片的宽度。

pixelHeight - 形成的图片的高度。

rectLeft - 设备坐标系中显示区域左边的像素位置。

rectTop - 设备坐标系中显示区域上方的像素位置。

rectRight - 设备坐标系中显示区域右边的像素位置。

rectBottom - 设备坐标系中显示区域下方的像素位置。

PixelFormat - 形成的图片的像素格式。

返回值：

返回图片的句柄。

- *** GetBarcodeBitmap**

将字符串转换为条形码图片。

函数原型：

Long GetBarcodeBitmap(BSTR contents, short format, long moduleHSize, long moduleVSize, short ecLevel)

参数：

Contents - 将被转换的内容。

Format - 编码格式，有效值如下：

UNSPECIFY = -1

CODE_39 = 1

CODE_128 = 3

EAN_8 = 6

UPC_A = 7

EAN_13 = 8

ITF = 9

如果选择的是 ITF，则字符串的长度必须是 8、10、12、16、20、24、44 中的一个。

QR_CODE = 15

moduleHSize - 图片的宽度。

moduleVSize - 图片的高度。

ecLevel - 交错级别。该参数只对 QR 码有效。

ECLEVEL_L = 0

ECLEVEL_M = 1

ECLEVEL_Q = 2

ECLEVEL_H = 3

返回值：

成功则返回一个二位的图片，否则返回 NULL。

事件

1) BeforeDraw

在开始绘制显示内容之前触发的事件。

函数原型：

Void BeforeDraw (long dc)

参数：

dc - 设备上下文相关的句柄。

返回值：

无

2) AfterDraw

在完成显示内容绘制之后触发的事件。

函数原型：

Void AfterDraw (long dc)

参数：

dc - 设备上下文相关的句柄。

返回值：

无

3) OnZoomChange

当 Zoomlevel 属性发生变化的时候触发该事件。

函数原型：

Void OnZoomChange()

参数：

无

返回值：

无

4) OnPageChange

当显示的 PDF 页面发生变化时触发该事件。

函数原型：

Void OnPageChange()

参数：

无

返回值：

无

备注：

翻页、页面跳转等操作都会触发该事件。

5) OnOpenDocument

当打开一个 PDF 文件时触发该事件。

函数原型:

Void OnOpenDocument (BSTR filepath)

参数:

Filepath - PDF 文件的路径。

6) OnOpenPassword

当尝试打开一个包含密码的 PDF 文件时触发该事件。

函数原型：

Void OnOpenPassword (BSTR* password, BOOL* cancel)

参数：

Password - PDF 文件的打开密码。

Cancel - 该参数用于判断当密码错误时，是否取消文件打开操作。值为 FALSE，该事件将一直被触发，直到密码输入正确为止。

7) OnOpenFile

当打开一个 PDF 文件失败时触发该事件。

函数原型：

void OnOpenFile(short Error)

参数：

Error - 返回的错误代码。

8) OnFilePathInvalidate

当打开文件路径错误时触发该事件。

函数原型：

void OnFilePathInvalidate(BSTR WarnString)

参数：

WarnString - 返回的错误提示信息。

9) OnSearchProgress

当搜索 PDF 文件时触发该事件。

函数原型：

Void OnSearchProgress (long pageNumber, long pageCount)

参数：

pageNumber - 开始搜索的 PDF 页码。

pageCount - 进行搜索的 PDF 总页码数。

10) CustomFileGetSize

当以 OpenCustomFile 方式打开 PDF 文档时触发该事件。

函数原型：

Void CustomFileGetSize (long* size)

参数：

size - [out] 该参数指向一个数值，用于设置 PDF 文件的长度。

11) CustomFileGetBlock

当以 OpenCustomFile 方式打开 PDF 文档时触发该事件。

函数原型：

Void CustomFileGetBlock(long pos, long pBuf, long size)

参数：

pos - 相对于文件起始位置的偏移位置，单位为字节。

pBuf - 指向一个用于接收 PDF 文件数据的缓冲。

Size - 缓冲区的大小。

备注：

从指定的位置开始获取数据。以字节为单位，从文件起始位置开始进行偏移，偏移量和缓冲区的大小不能超出 PDF 文件的长度。

12) OnDocumentChange

当 PDF 文档内容发生改变时触发该事件。

函数原型：

Void OnDocumentChange()

参数：

无

13) OnCloseDocument

当关闭一个 PDF 文件时触发该事件。

函数原型：

Void OnCloseDocument (BSTR filepath)

参数：

Filepath - 将关闭的 PDF 文件的路径。

14) OnShowSavePrompt

当关闭一个已经被修改过的 PDF 文件时触发该事件。

函数原型：

void OnShowSavePrompt(BOOL* bShow, short * nResult)

参数：

bShow - 该参数指明是否在 ActiveX 控件中显示默认的提示框。

nResult - 该参数指明是要保存被修改过的 PDF 文件，默认不保存。

15) OnClick

当鼠标左键点击时触发该事件。

函数原型：

Void OnClick (long hWnd, long ClientX, long ClientY)

参数:

hWnd - 鼠标左键点击时的窗口句柄。

ClientX - 鼠标左键在 ActiveX 控件窗口客户区点击时水平坐标。

ClientY - 鼠标左键在 ActiveX 控件窗口客户区点击时垂直坐标。

16) OnDbClick

当鼠标左键双击时触发该事件。

函数原型：

Void OnDbClick (long hWnd, long ClientX, long ClientY)

参数：

hWnd - 鼠标左键双击时的窗口句柄。

ClientX - 鼠标左键在 ActiveX 控件窗口客户区双击时水平坐标。

ClientY - 鼠标左键在 ActiveX 控件窗口客户区双击时垂直坐标。

17) OnRButtonClick

当鼠标右键双击时触发该事件。

函数原型：

void OnRButtonClick(long hWnd, long ClientX, long Client)

参数：

hWnd - 鼠标右键双击时的窗口句柄

ClientX - 鼠标右键在 ActiveX 控件窗口客户区双击时水平坐标。

ClientY - 鼠标右键在 ActiveX 控件窗口客户区双击时垂直坐标。

18) OnDownLoadFinish

当从网络下载 PDF 文件结束时触发该事件。

函数原型:

void OnDownLoadFinish()

参数:

无

19) OnUploadFinish

当调用 UploadCurFileToFTP 时触发该事件。

函数原型：

void OnUploadFinish(short nRetCode)

参数：

nRetCode - 返回的操作结果。

返回值：

无

20) OnErrorOccurred

当调用 SDK 提供的接口出现异常时触发该事件。

函数原型：

void OnErrorOccurred(BSTR lpszErrorMsg)

参数：

lpszErrorMsg - 异常时提示的字符串。

备注：

目前仅支持 GetToolByIndex 和 ShowToolbarButton 两个接口。

21) OnTextHyperLink

点击文本链接时触发该事件。

函数原型：

void OnTextHyperLink(BSTR csUrl, boolean* cancel)

参数：

csUrl - 文本链接的 URL 地址。

cancel - 该参数用于判断是否响应文本链接。

返回值：

无

22) *OnHyperLink

点击超链接注释对象时触发该事件。

函数原型：

Void OnHyperLink(BSTR linktype, BSTR linkdata, Link_Dest* dest, BOOL* cancel)

参数：

Linktype - 超链接类型的字符串。

超链接类型的字符串包括：

GoTo ： 跳转到当前 PDF 文档的某一页，linkdata 为 None，dest 包括超链接重定向的目的地位置信息。

GoToR： 跳转到本地磁盘中的另一个本地 PDF 文件。若要在新的窗口中显示文件，则 linkdata 中在文件名后加 1，反之 linkdata 中在文件名后加 0。dest 包括超链接重定向的目的地位置信息。

Launch： 执行外部程序，若要在新的窗口中打开文档，则 linkdata 中在文件名后加 1，反之则在文件名后加 0。

linkData - 超链接附加信息的字符串。

dest - 超链接重定向的目的地。

Cancel - 值为 TRUE，则不打开超链接；值为 FALSE，则打开超链接。

23) *OnDoGoToRAction

当执行 GoToR（超链接类型）操作时触发该事件。

函数原型：

Void OnDoGoToRAction(BSTR sFilePath, Link_Dest* dest)

参数：

sFilePath - 目标文件的路径。

dest - 文件中的目标对象。

返回值：

无

24) OnExcuteMenuItem

当通过 OnAddmenuItemAction 事件添加的用户自定义菜单项被执行时触发该事件。

函数原型：

void OnExcuteMenuItem(BSTR sMenuItem, boolean* bResult)

参数：

sMenuItem - 用户自定义的菜单项

bResult - 操作产生的结果（TRUE 或者 FALSE）

返回值：

无

25) *OnContextMenuIndex

在鼠标右键弹出的菜单中点击某一项时触发该事件。与 SetContextMenuString 函数配合使用。

函数原型：

void OnContextMenuIndex(short nIndex)

参数：

nIndex - 菜单的索引值。

26) *OnAddMenuItemAction

当执行”添加一个菜单项”操作时触发该事件。

函数原型：

void OnAddMenuItemAction(BSTR*pMenuItem)

参数：

pMenuItem - 菜单项的内容。

返回值：

无

27) *OnFetchAsyncFileData

当打开一个异步加载的文件时，通过触发该事件来获取所需的文件数据。

函数原型：

void OnFetchAsyncFileData(long offset, long size);

参数：

offset： 所需数据在文件中的偏移量。

size： 所需数据的长度。

返回值：

无。

28) *OnCurPageIndexChanged

当应用程序改变了 CurPage 值时，即当前页面改变时触发该事件。

函数原型：

Void OnCurPageIndexChanged (long curPageIdx, long newPageIdx)

参数：

curPageIdx - 改变前的 CurPage 值。

newPageIdx - 改变后的 CurPage 值。

返回值：

无。

29) *OnSigContextMenuIndex

当点击签名域的右键菜单项时触发该事件。

函数原型：

void OnSigContextMenuIndex(short nIndex, IDispatch* SignatureField);

参数：

nIndex - 菜单项的索引值。

SignatureField - 鼠标右键点击的签名域。

返回值：

无。

30) OnPagesContextMenuIndex

当点击页面缩略图的右键菜单项时触发该事件。

函数原型：

void OnPagesContextMenuIndex(short nIndex,VARIANT* pageArray)

参数：

nIndex - 页面缩略图的右键菜单项的索引值。

pageArray - 页面数组。

返回值：

无。

31) OnBookmarkContextMenuIndex

当点击书签的右键菜单项时触发该事件。

函数原型：

void OnBookmarkContextMenuIndex(short nIndex)

参数：

nIndex - 书签右键菜单项的索引值。

返回值：

无。

32) *OnFormFieldClick

当点击表单域时触发该事件。

函数原型：

void OnFormFieldClick(PDFFormField* pClickedField)

参数：

nIndex - 表单域的索引值。

返回值：

无。

33) *OnFormFieldKeyDown

选中一个表单域，键盘按键按下时触发该事件。

函数原型：

void OnFormFieldKeyDown(PDFFormField* pFormField, long* nKey)

参数：

pFormField - 表单域。

nKey - 键盘的虚拟键码。

返回值：

无。

34) *OnFormFieldKeyUp

选中一个表单域，键盘按键弹起时触发该事件。

函数原型：

void OnFormFieldKeyUp(PDFFormField* pFormField, long* nKey)

参数：

pFormField - 表单域。

nKey - 键盘的虚拟键码。

返回值：

无。

35) *OnSetFocus

当在控件窗口里设置焦点时触发该事件。

函数原型：

void OnSetFocus(long hwnd)

参数：

hwnd - 设置焦点的控件的窗口句柄。

返回值：

无。

36) *OnKillFocus

当从控件窗口释放焦点时触发该事件。

函数原型：

void OnKillFocus(long hwnd)

参数：

hwnd - 释放焦点的控件窗口句柄。

返回值：

无。

37) *OnDbClickEx

当鼠标左键双击时触发该事件。

函数原型：

void OnDbClickEx(long hWnd, long ClientX, long ClientY, boolean* bRet)

参数：

hwnd - 鼠标左键双击时的窗口句柄。

ClientX - ActiveX 控件的窗口坐标，x 坐标点。

ClientY - ActiveX 控件的窗口坐标，y 坐标点。

bRet - 设置控件接收到双击事件后是继续走控件内部的流程，还是走客户指定的回调里的流程。TRUE 时走客户在回调内所做的事情，FALSE 时继续往下走控件的流程。

返回值：

无。

38) *OnRButtonClickEx

当鼠标右键双击时触发该事件。

函数原型：

void OnRButtonClickEx(long hWnd, long ClientX, long ClientY, boolean* bRet)

参数：

hwnd - 鼠标右键双击时的窗口句柄。

ClientX - ActiveX 控件的窗口坐标，x 坐标点。

ClientY - ActiveX 控件的窗口坐标，y 坐标点。

bRet - 设置控件接收到双击事件后是继续走控件内部的流程，还是走客户指定的回调里的流程。TRUE 时走客户在回调内所做的事情，FALSE 时继续往下走控件的流程。

返回值：

无。

39) #FormFieldError

在设置 PDF 表单域时出现异常会触发该事件。

函数原型：

Void FormFieldError(long nErrorCode)

参数:

nErrorCode - 返回的异常错误码。

IPDFPrinter

通过 IPDFPrinter 所提供的接口可以进行打印设置和执行 PDF 打印操作。

属性

1) PrinterName

类型：

BSTR

描述：

打印机名称。

2) PrinterRangeMode

类型：

PrinterRangeMode

描述：

PDF 文件打印的范围选择，可设置如下选项：

PRINT_RANGE_ALL = 0,
PRINT_RANGE_CURRENT_VIEW = 1,
PRINT_RANGE_CURRENT_PAGE = 2,
PRINT_RANGE_SELECTED = 3,

3) PrinterRangeFrom

类型：

short

描述：

PDF 文件打印的起始页码。

备注：

该属性仅在 PrinterRangeMode = PRINT_RANGE_SELECTED 时有效。

4) PrinterRangeTo

类型：

short

描述：

PDF 文件打印的结束页码。

备注：

该属性仅在 PrinterRangeMode = PRINT_RANGE_SELECTED 时有效。

5) NumOfCopies

类型：

short

描述：

PDF 文件打印的份数。

6) Scaling

类型：

short

描述：

打印对话框中的打印缩放比例。

7) AutoRotate

类型：

boolean

描述：

该属性值表示打印对话框上是否选中“自动旋转“选项。

1 = 选中“自动旋转”选项

0 = 不选“自动旋转”选项

8) AutoCenter

类型：

boolean

描述：

该属性值表示打印对话框上是否选中“自动居中” 选项。

1 = 选中“自动居中”选项

0 = 不选“自动居中”选项

9) Collate

类型：

boolean

描述：

该属性值表示打印对话框上是否选中“自动分页”选项。

1 = 选中“自动分页”选项

0 = 不选“自动分页”选项

10) Rotation

类型：

short

描述：

PDF 文件打印时的旋转度数。

11) RangeSubset

类型：

short

描述：

该属性值表示 PDF 文件打印时包含的子集情况。

12) ReversePage

类型：

boolean

描述：

该属性值表示打印对话框上是否选中“逆序打印”选项。

1 = 选中“逆序打印”选项

0 = 不选中“逆序打印”选项

13) PageBorder

类型：

boolean

描述：

设置是否打印页面边框。

14) PrintWhat

类型：

Short

描述：

PDF 文件的打印内容，有效值为：

0 = 只打印正文；

1 = 打印正文和批注等其他内容；

2 = 只打印批注等其他内容。

方法

1) PrintWithDialog

进行 PDF 文件打印时，弹出打印对话框。

函数原型：

void PrintWithDialog()

参数：

无

返回值：

无

2) PrintQuiet

进行 PDF 文件打印时，选择静默打印。

函数原型：

void PrintQuiet()

参数：

无

返回值：

无

3) SetPaperSize

选择打印纸，具体的参数值请参照 Windows SDK 说明文档。

函数原型：

void SetPaperSize (long paperSize)

参数：

paperSize - 打印纸的尺寸，Windows SDK 说明文档规定的值。

返回值：

无

4) GetSystemPrinterCount

获取系统打印机的数量。

函数原型：

Long GetSystemPrinterCount()

参数：

无

返回值：

返回系统打印机的总数。

5) GetSystemPrinterNameByIndex

通过指定索引值，获取系统打印机的名称。

函数原型：

BSTR GetSystemPrinterNameByIndex(long index)

参数：

无

返回值：

返回对应索引值的系统打印机名称。

6) SetPaperSizeByPage

静默打印和正常打印过程，选中“根据 PDF 页面大小选择打印纸张”选项。

函数原型：

void SetPaperSizeByPage(BOOL bPage)

参数：

bPage - 设置是否选中“根据 PDF 页面大小选择打印纸张”选项。

TRUE = 选中“根据 PDF 页面大小选择打印纸张”选项

FALSE = 不选“根据 PDF 页面大小选择打印纸张”选项

返回值：

无

IPDFDocumentInfo

通过 IPDFDocumentInfo 所提供的接口可以获取 PDF 文档的属性信息。

属性

1) Author

类型：

BSTR

描述：

PDF 文件的作者。

2) Subject

类型：

BSTR

描述：

PDF 文件的主题。

3) CreatedDate

类型：

BSTR

描述：

PDF 文件的创建时间。

4) ModifiedDate

类型：

BSTR

描述：

PDF 文件的最近修改时间。

5) Keywords

类型：

BSTR

描述：

PDF 文件的关键词。

6) Creator

类型：

BSTR

描述：

PDF 文件的创建者。

7) Producer

类型：

BSTR

描述：

创建 PDF 文件的工具。

8) Title

类型：

BSTR

描述：

PDF 文件的标题。

IFindResult

通过 IFindResult 所提供的接口可以获取搜索结果的信息。

方法

1) GetFindRectsCount

获取搜索结果的矩形框对象个数。

函数原型：

long GetFindRectsCount()

返回值：

成功则返回 TRUE，反之则返回 FALSE。

2) GetFindRectByIndex

获取指定的矩形框。

函数原型：

Long GetFindRectByIndex(longnIndex,float* left, float* bottom, float* right, float* top)

参数：

nIndex	-	矩形框的索引值。
left	-	矩形框区域的左边坐标（PDF 坐标系统）。
bottom	-	矩形框区域的底部坐标（PDF 坐标系统）。
right	-	矩形框区域的右边坐标（PDF 坐标系统）。
top	-	矩形框区域的顶端坐标（PDF 坐标系统）。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

3) GetFindPageNum

获取搜索结果所在的 PDF 页面索引值。

函数原型：

long GetFindPageNum()

参数：

无

返回值：

返回页面索引值。

4) GetFindFileName

获取执行搜索操作的文件名。

函数原型：

BSTR GetFindFileName()

参数：

无

返回值：

返回搜索结果所在的文件名。

5) GetFindString

获取搜索结果内容。

函数原型：

BSTR GetFindString()

参数：

无

返回值：

返回搜索结果内容。

IPDFOutline

通过 IPDFOutline 所提供的接口可以获取 PDF 文件的书签的相关信息。

方法

1) GetOutlineDest

获取书签的链接目标。

函数原型：

ILink_Dest* GetOutlineDest ()

参数：

无

返回值：

返回书签的跳转目标。

2) GetOutlineAction

获取书签所关联的动作。

函数原型：

IPDF_Action* GetOutlineAction ()

参数：

无

返回值：

返回书签的跳转动作。

3) GetOutLineColor

获取书签文本的颜色（RGB）。

函数原型：

DWORD GetOutlineColor ()

参数：

无

返回值：

返回书签文本的颜色。

4) NavigateOutline

遍历指定的大纲结点所包含的子节点的数据。

函数原型：

void NavigateOutline ()

参数：

无

返回值：

无

5) GetOutlineTitle

获取大纲结点的标题。

函数原型：

BSTR GetOutlineTitle()

参数：

无

返回值：

返回当前大纲结点的标题。

6) GetOutLineTitle2

获取大纲节点的标题，作为一个变量。

函数原型：

VARIANT GetOutLineTitle2 ()

参数：

无

返回值：

返回大纲节点的标题，作为一个变量。

7) GetOutlineExpandValue

获取大纲子节点的数量。

函数原型：

Long GetOutlineExpandValue()

返回值：

返回子节点的数量。

ILink_Dest

通过 ILink_Dest 所提供的接口可以获取链接所关联的 PDF 页面的相关信息。

方法

1) GetPageIndex

获取 PDF 页面的索引值。

函数原型：

long GetPageIndex()

参数：

无

返回值：

返回 PDF 页面的索引值。

2) GetZoomMode

获取页面显示模式。

函数原型：

long GetZoomMode()

参数：

无

返回值：

返回表示页面显示模式的值：

2 - 适合页面

3 - 适合宽度

4 - 适合宽度

3) GetZoomParamCount

获取缩放参数。

函数原型：

long GetZoomParamCount()

参数：

无

返回值：

返回缩放参数值。

4) GetZoomParam

获取指定的缩放参数的内容。

函数原型：

double GetZoomParam(long nIndex)

参数：

无

返回值：

返回指定的缩放参数的内容，该内容描述的是缩放的信息。

5) GetDestName

获取目标对象名称。

函数原型：

BSTR GetDestName()

参数：

无

返回值:

返回目标对象名称。

IPDFAction

通过 IPDFAction 所提供的接口可以获取 Action 的信息。

方法

1) GetURIPath

获取 Action 所关联的 URI 路径。

函数原型：

BSTR GetURIPath()

参数：

无

返回值：

返回 URI 路径。

2) GetFilePath

获取 Action 所关联的外部文件的路径。

函数原型:

BSTR GetFilePath()

参数：

无

返回值:

返回外部文件的路径。

3) GetType

获取 Action 的类别。

函数原型:

int GetType()

参数：

无

返回值:

返回表示 Action 类型的值：

1 - Go to Action;

2 - Remote Go-To Actions;

4 - Launch Actions;

6 - URI Actions;

4) GetDest

获取 Action 所关联的 dest

函数原型：

Ilink_Dest* GetDest()

参数：

无

返回值：

返回跳转定义。

#IPDFForm

方法

1) #ImportFromFDF

从 FDF 文件导入表单内容。

函数原型:

Void ImportFromFDF(LPCTSTR bstrFullPath)

参数:

bstrFullPath - 待导入的 FDF 文件路径。

返回值:

无

2) #ExportToFDF

导出特定表单元素并保存为 FDF 文件。

函数原型：

Void ExportToFDF (LPCTSTR bstrFullPath, BOOL bEmptyFields, const VARIANT FAR& arrFields)

参数：

bstrFullPath - 导出后生成的 FDF 文件的路径

bEmptyFields - 该参数用于判断

值为 True 则仅导出 arrField 指定的表单元素；

值为 False 则导出除 arrFields 函数指定的元素外的所有表单元素。

arrFields - 待导出的表单元素数组。

返回值：

无

3) #AddField

添加一个表单域。

函数原型：

LPDISPATCH AddField(LPCTSTR bstrFieldName, LPCTSTR bstrFieldType, long pageIndex, float left, float top, float right, float bottom)

参数：

bstrFieldName - 表单全名。

bstrFieldType - 表单域类型。

表单域类型包括：

文本域

按钮
组合框
列表框
复选框
单选按钮

pageIndex - 添加表单域的 PDF 页面的索引值（索引值从 0 开始计算）。
Left - 表单域左边的坐标。
Top - 表单域顶部的坐标。
right - 表单域右边的坐标。
Bottom - 表单域底部的坐标。

返回值：

返回新创建的表单对象。

备注：

表单域的坐标是在旋转后的页面中测量的，也就是说，无论页面怎么旋转，表单左下角的坐标都是[0,0]。

4) # GetSelectedField

选中表单域。

函数原型：

IPDFFormField* GetSelectedField()

参数：

无

返回值：

表单域对象。

5) #RemoveField

删掉指定表单域。

函数原型:

Void RemoveField(IPDFFormField pFormField)

参数:

pFormField - 待删除的表单域。

返回值:

无

6) #RemoveFieldsByName

删除一个表单域

函数原型：

Void RemoveFieldsByName (LPCTSTR bstrFieldName)

参数：

bstrFieldName - 待删除表单的全名，若表单中含有多个子注释，则所有子注释都将会被删除；若系统中含有多个同名的表单，则所有同名的表单都将会被删除。

返回值：

无

7) #GetFieldsCount

获取表单域的总数量。

函数原型：

long GetFieldsCount()

参数：

无

返回值：

成功则返回表单域的总数量，反之则返回-1。

8) #GetFieldByIndex

获取特定的表单域。

函数原型：

LPDISPATCH GetFieldByIndex(long index)

参数：

index - 表单域的索引值。

返回值：

成功则返回表单域对象的指针，反之则返回 NULL。

#IPDFFormField

属性

1) #Alignment

类型：

String

描述：

文本对齐方式。

备注：

仅用于文本域。

2) #BorderStyle

类型：

String

描述：

表单域边框的类型。

备注：

可用于所有表单类型。

3) #BorderWidth

类型：

short

描述：

表单域边框的宽度。

备注：

可用于所有表单类型。

4) #ButtonLayout

类型：

short

描述：

按钮的外观设置。有效值包括：

0 – 仅为文本；按钮有文字说明，但没有图标。

1 – 仅为图标；按钮有图标，但没有文字说明。

2 – 在文本上显示图标；按钮图标显示在文字说明的顶部。

3 – 在图标上显示文本；按钮的文本显示在图标的顶部。

4 – 先显示图标，然后显示文本；按钮的图标显示在文字说明的左边。

5 – 先显示文本，然后显示图标；按钮的图标显示在文字说明的右边。

6 – 在图标下显示文本；按钮的文本显示在图标顶部的下方。

备注：

仅用于按钮类型。

5) #CalcOrderIndex

类型：

short

描述：

优化列阵中当前表单域的索引。

备注：

可用于所有的表单类型。

6) #CharLimit

类型：

short

描述：

文本域的（文本）字符数限制。

备注：

仅用于文本域类型。

7) #DefaultValue

类型：

String

描述：

表单域的默认值。

备注：

可用于所有的表单类型。

8) #Behavior

类型：

String

备注：

有效值包含：None, Invert, Outline, Push

N (None) - 无高亮显示

I (Invert) - 转化注释内容

O (Outline) - 转化注释边框

P (Push) - 在页面文字的下方显示注释

9) #IsEditable

类型：

Boolean

描述：

该属性用于判断 Combo Box 是否可编辑。

备注：

仅用于 Combo Box。

10) #IsHidden

类型：

Boolean

描述：

该属性用于判断是否隐藏表单域。

备注：

可用于所有的表单类型。

11) #IsMultiline

类型：

Boolean

描述：

该属性用于判断文本域是单行浏览或是多行浏览。

备注：

仅用于文本域。

12) #IsPassword

类型：

Boolean

描述：

该属性用于判断输入密码时以明文或者密文方式显示。

备注：

仅用于文本域。

13) #IsReadOnly

类型：

Boolean

描述：

该属性用于判断表单域是否只读。

备注：

可用于所有的表单类型。

14) #IsRequired

类型：

Boolean

描述：

该属性用于判断是否将表单域设置为非空。

备注：

可用于组合框、单选按钮和文本框。

15) #NoViewFlag

类型：

Boolean

描述：

该属性用于判断是否显示表单元素。

“1”表示隐藏表单元素；

“0”表示显示表单元素。

备注：

可用于所有表单类型。

16) #PrintFlag

类型：

Boolean

描述：

该属性用于判断打印内容是否包含表单元素。

“1”表示包含表单元素；

“0”表示不包含表单元素。

备注：

可用于所有表单类型

17) #Name

类型：

String

描述：

当前选中的表单域的名称。

备注：

只读，可用于所有表单类型。

18) #Style

类型：

CString

描述：

设置复选框和单选按钮的形状：

1) 格子形
2) 十字形
3) 菱形
4) 圆形
5) 星形
6) 方形

备注：

可用于复选框和单选按钮。

19) #TextFont

类型：

String

描述：

字体（Reference 1.7 416）

字体可设置为：

Courier
Courier-Bold
Courier-Oblique
Courier-BoldOblique
Helvetica
Helvetica-Bold
Helvetica-Oblique
Helvetica-BoldOblique
Symbol
Times-Roman
Times-Bold
Times-Italic
Times-BoldItalic
ZapfDingbats

备注：

可用于除复选框和单选按钮以外的所有表单。

20) #TextSize

类型：

Short

描述：

表单域中文本的大小

备注：

可用于除复选框和单选按钮外的所有表单

21) #Type

类型：

String

描述：

表单类型

备注：

表单类型可设置为: 文本域、按钮、组合框、列表框、复选框、单选按钮。

22) #Value

类型：

String

描述：

当前值

备注：

以下表单形式包含这种属性：

1）文本

2）组合框

3）单选按钮

4）复选框 <是&否>

5）列表框

23) #Tooltip

类型：

String

描述：

提示信息。

备注：

可用于所有表单类型。

24) #Orientation

类型：

Short

描述：

表单文本的旋转方向。

备注：

可用于所有表单类型。

25) #DirtyFlag

类型：

Short

描述：

该属性用于判断表单域是否被改动过。

“1”表示被改动过；

“0”表示未被改动过。

26) # ID

类型：

BSTR

描述：

表单域的序号。

方法

1) #PopulateListOrComboBox

给列表框或组合框中的条目分配数值。

函数原型：

Void PopulateListOrComboBox (const VARIANT& arrItems, const VARIANT& arrExportVal)

参数：

arrItem - 一个字符串数组，每个元素代表一个项目名称。

arrExportVal - 一个字符串数组，大小与第一个参数相同，每个元素代表一个导出值。

返回值：

无

2) #SetBackgroundColor

设置表单域的背景颜色。

函数原型：

Void SetBackgroundColor (LPCTSTR bstrColorSpace, float redC, float greenM, float blueY, float AlphaK)

参数：

bstrColorSpace - 可设置的颜色空间，包括透明、灰色、RGB 或 CMYK 色域。

redC - 有效值为 0~1。
greenM - 有效值为 0~1。
blueY - 有效值为 0~1。
AlphaK - 仅用于 CMYK 色域。

返回值：

无

备注：

使用 T、G、RGB 和 CMYK 分别代表四种颜色空间。

- 设置为 T 和 G 时，需要设置 redC；
- 设置为 RGB 时，需要设置 redC、greenM 和 blueY；
- 设置为 CMYK 时，需要设置 redC、greenM、blueY 和 AlphaK；

3) #SetBorderColor

设置表单域的边框颜色。

函数原型：

void SetBorderColor (LPCTSTR bstrColorSpace, float redC, float greenM, float blueY, float AlphaK)

参数：

bstrColorSpace - 可设置的颜色空间包括：透明、灰色、RGB 或 CMYK 色域。
redC - 有效值为 0~1。
greenM - 有效值为 0~1。
blueY - 有效值为 0~1。
AlphaK - 仅用于 CMYK 色域。

返回值：

无

备注：

使用 T、G、RGB 和 CMYK 分别代表四种颜色空间。

- 设置为 T 和 G 时，需要设置 redC；
- 设置为 RGB 时，需要设置 redC、greenM 和 blueY；
- 设置为 CMYK 时，需要设置 redC、greenM、blueY 和 AlphaK。

4) #SetForegroundColor

设置表单域的前背景色。

函数原型：

Void SetForegroundColor (LPCTSTR bstrColorSpace, float redC, float greenM, float blueY, float AlphaK)

参数：

bstrColorSpace - 可设置的颜色空间包括：透明、灰色、RGB 或 CMYK 色域。

redC - 有效值为 0~1。

greenM - 有效值为 0~1。

blueY - 有效值为 0~1。

AlphaK - 仅用于 CMYK 色域。

返回值：

无

备注：

使用 T、G、RGB 和 CMYK 分别代表以上四种颜色。

- 设置为 T 和 G 时，需要设置 redC;
- 设置为 RGB 时，需要设置 redC、greenM 和 blueY；
- 设置为 CMYK 时，需要设置 redC、greenM、blueY 和 AlphaK。

5) #SetButtonCaption

对按钮上显示的文本进行设置。

函数原型：

void SetButtonCaption (LPCTSTR bstrFace, LPCTSTR bstrCaption)

参数：

bstrFace - 一个字符串，指定说明文字的朝向。

有效值为：

N - 普通

D - 向下

R - 反转

bstrCaption - 按钮文字

返回值：

无

6) #SetButtonIcon

设置按钮图标。

函数原型:

Void SetButtonIcon (LPCTSTR bstrFace, LPCTSTR bstrFilePath)

参数:

bstrFace - 一个字符串，指定说明文字的朝向。

有效值为：

N = 普通

D = 向下

R = 反转

BstrFilePath - 图标文件的路径。

返回值：

无

7) #SetExportValues

导出单选按钮和复选框时，根据不同的选择（如：已选、未选、已勾选、未勾选等）设置导出值。

函数原型：

void SetExportValues (const VARIANT& arrExportVal)

参数：

arrExportVal - 导出值数组。

返回值：

无

8) #SetJavaScriptAction

设置 Javascript 脚本动作。

函数原型：

Void SetJavaScriptAction (LPCTSTR bstrTrigger, LPCTSTRbstrJavaScript)

参数：

bstrTrigger - 一个字符串，指定动作触发的条件。

有效的字符串包括：

up

down

enter

exit

calculate

validate

format

keystroke

bstrJavaScript - 脚本动作

返回值：

无

9) #SetResetFormAction

设置重置表单域动作。

函数原型：

void SetResetFormAction(LPCTSTR bstrTrigger, long bFlags, const VARIANT& arrFields)

参数：

bstrTrigger - 一个字符串， 指定动作触发的条件。

有效的字符串包括：

Up = 鼠标向上移动

down = 鼠标向下移动
enter = 鼠标输入
exit = 鼠标退出
bFlags - 一组定义动作特性的标记集。
arrFields - 待导出的表单元素组。

返回值：

无

10) #SetSubmitFormAction

设置提交表单域的动作。

函数原型:

Void SetSubmitFormAction (LPCTSTR bstrTrigger, LPCTSTR bstrURL, long bFlags, const VARIANT& arrFields)

参数：

bstrTrigger - 一个字符串，指定动作触发的条件
有效值为：
up = 鼠标向上移动
down = 鼠标向下移动
enter = 鼠标输入
exit = 鼠标退出
bstrURL - 含有 URL 的字符串。
bFlags - 一组定义动作特性的标记集。
arrFields - 待提交的表单元素组。

返回值：

无

11) #GetPageIndex

获取表单域所在的 PDF 页页码。

函数原型：

long GetPageIndex()

参数：

无

返回值：

返回表单域所在的 PDF 页页码。

12) #GetRectTop

获取表单域顶部的坐标。

函数原型：

float GetRectTop()

参数：

无

返回值：

返回表单域顶边的坐标。

13) #GetRectLeft

获取表单域左边的坐标。

函数原型：

float GetRectLeft()

参数：

无

返回值：

返回表单域左边的坐标。

14) #GetRectRight

获取表单域右边的坐标。

函数原型：

float GetRectRight()

参数：

无

返回值：

返回表单域右边的坐标。

15) #GetRectBottom

获取表单域底边的坐标。

函数原型：

float GetRectBottom()

参数：

无

返回值：

返回表单域底边的坐标。

^IPDFPageAnnots

方法

1) ^GetAnnot

获取特定 PDF 注释的指针。

函数原型：

CPDFAnnot GetAnnot(long AnnotIndex)

参数：

AnnotIndex - PDF 注释的索引值。

返回值：

成功则获取特定 PDF 注释的指针，反之则返回 NULL。

2) ^ReleaseAnnot

释放指定的 PDF 注释。

函数原型：

void ReleaseAnnot(IPDFAnnot* Annot)

参数：

Annot - PDF 注释对象。

返回值：

无

3) ^GetLTAnnot

获取特定 PDF 注释的指针。

函数原型：

IPDFAnnot* GetLTAnnot(long AnnotIndex)

参数：

AnnotIndex - PDF 注释的索引值。

返回值：

成功则获取特定 PDF 注释的指针，反之则返回 NULL。

返回值：

此接口是 GetAnnot 的扩展。它可用于 C#环境中，并且要求和 ReleaseLTAnnot 配合使用。

4) ^ReleaseLTAnnot

释放指定的 PDF 注释。

函数原型：

void ReleaseLTAnnot(IPDFAnnot* Annot)

参数：

Annot - PDF 注释对象。

返回值：

无

备注：

此接口可用于 C#环境中，并且要求和 GetLTAnnot 配合使用。

5) ^AddAnnot

添加一个新的 PDF 注释。

函数原型：

CPDFAnnot AddAnnot (LPDISPATCH AnnotToAddAfter, LPCTSTR SubType, float left, float top, float right, float bottom)

参数：

AnnotToAddAfter - 保留。设为 NULL。

SubType - 注释的子类型。

子类型包括:

"Rectangle", "Cloudy", "Ellipse", "Circle", "Arrow" , "Polygon", "Line", "Square", "PolyLine", "Pencil", "Highlight", "Squiggly", "Underline", "StrikeOut", "Replace", "Caret", "Note", "FileAttachment", "Typewriter", "Callout", "Textbox", "Image", "Movie", "Sound", "Rectangle Link", "Quadrilateral Link".

Left - 该注释矩形框的左边坐标。

Top - 该注释矩形框的顶边坐标。

Right - 该注释矩形框的右边坐标。

Bottom - 该注释矩形框的底边坐标。

所有坐标均基于 PDF 坐标空间。

返回值：

成功则返回新的注释对象，反之则返回 NULL。

6) ^RemoveAnnot

删除特定 PDF 注释。

函数原型：

Long RemoveAnnot(LPDISPATCH AnnotToRemove)

参数：

AnnotToRemove - 待删除 PDF 注释。

返回值：

成功则返回 0，反之则返回-1。

7) ^GetAnnotIndex

获取特定 PDF 注释的索引。

函数原型：

Long GetAnnotIndex(LPDISPATCH Annot)

参数：

Annot - 特定 PDF 注释的信息

返回值：

成功则返回特定 PDF 注释的索引，反之则返回-1。

8) ^GetAnnotsCount

获取当前 PDF 页面上 PDF 注释的总数量。

函数原型：

Long GetAnnotsCount()

参数：

无

返回值：

返回 PDF 注释的数量。

^IPDFAnnot

属性

1) ^Thickness

类型：

Short，读/写

描述：

PDF 注释边框的宽度。

备注：

用于图形标注、测量工具以及其他具有边框的注释（即云形、箭头、线条、方形、矩形、圆形、椭圆形、多边形、折线、铅笔工具、注释框、文本框、链接等）。

2) ^BorderStyle

类型：

Short，读/写

描述：

注释边框的类型。

备注：

边框类型包括七种：

1 - 实线
2 - 虚线类型 1
3 - 虚线类型 2
4 - 虚线类型 3
5 - 虚线类型 4
6 - 虚线类型 5
7 - 虚线类型 6
8 - 云形类型 1
9 - 云形类型 2

实线类型可用于：

"Rectangle", "Cloudy" , "Ellipse", "Circle", "Arrow" , "Polygon", "Line", "Square" , "PolyLine", "Callout", "Textbox" ,"Image", "Movie", "Sound", "Rectangle Link", "Quadrilateral Link".

虚线类型可用于：

"Rectangle", "Cloudy" , "Ellipse", "Circle", "Arrow" , "Polygon", "Line", "Square" , "PolyLine", "Callout", "Textbox".

云形类型可用于：

"Rectangle", "Cloudy", "Ellipse", "Circle", "Polygon", "Square" , "Callout", "Textbox"

3) ^Color

类型：

OLE_COLOR，读/写

描述：

PDF 注释的背景色。

备注：

可用于所有注释。

4) ^LineStartingStyle

类型：

Short，读/写

描述：

线条的起画格式。

备注：

用于线条、箭头、折线及注释框等注释。

包括以下 10 种格式：

0	无
1	方形
2	圆形
3	菱形
4	张开
5	闭合
6	对接
7	反向张开
8	反向闭合
9	斜线

5) ^LineEndingStyle

类型：

Short，读/写

描述：

线条的结束格式。

备注：

用于线条、箭头和折线等注释。

包括 10 种类型：

0	无
1	方形
2	圆形

3	菱形
4	张开
5	闭合
6	对接
7	反向张开
8	反向闭合
9	斜线

6) ^FillColor

类型：

OLE_COLOR，读/写

描述：

PDF 注释的填充色。

备注：

用于云形、箭头、线条、方形、矩形、圆圈、椭圆、多边形和折线等注释。

7) ^Opacity

类型：

Short，读/写

描述：

PDF 注释的透明度值。

备注：

不支持链接、视频和音频等注释。

8) ^Author

类型：

BSTR，读/写

描述：

PDF 注释的作者。

备注：

不支持图片、视频、音频和链接等注释。

9) ^Subject

类型：

BSTR，读/写

描述：

PDF 注释的主题。

备注：

不支持图片、视频、音频和链接等注释。

10) ^CreationDate

类型：

DATE，只读

描述：

PDF 注释的创建时间

11) ^ModificationDate

类型：

DATE，只读

描述：

PDF 注释的最后修改日期。

12) ^Locked

类型：

Boolean，读/写。

描述：

该属性值表示注释是否被锁定。

备注：

可用于所有注释。

13) ^Print

类型：

Boolean，读/写

描述：

该属性值表示是否可以打印注释。

备注：

适用于所有注释。

14) ^ReadOnly

类型：

Boolean，读/写

描述：

该属性值表示注释是否为只读。

备注：

适用于所有注释。

15) ^Description

类型：

BSTR，读/写

描述：

PDF 注释的描述。

备注：

仅适用于文件附件注释。

方法

1) ^GetType

获取 PDF 注释的类型。

函数原型：

BSTR GetType()

参数：

无

返回值：

返回 PDF 注释的类型。

2) ^GetSubType

获取 PDF 注释的子类型。

函数原型：

BSTR GetSubType()

参数：

无

返回值：

返回 PDF 注释的子类型。

3) ^GetContents

获取 PDF 注释的内容。

函数原型：

BSTR GetContents()

参数：

无

返回值：

返回 PDF 注释的内容。

4) ^SetContents

设置 PDF 注释的内容。

函数原型：

long SetContents(LPCTSTR Contents)

参数：

Contents - 需要设置的内容。

返回值：

成功则返回 0，反之则返回-1。

5) ^IsPopupOpen

设置打开弹出框。

函数原型：

BOOL IsPopupOpened()

参数：

无

返回值：

打开弹出框则返回 TRUE，反之则返回 FALSE。

6) ^SetPopupOpen

设置打开弹出框。

函数原型：

Long SetPopupOpened(BOOL Open)

参数：

Open - 该值表示是否打开弹出框。

返回值：

成功则返回 0，反之则返回-1。

7) ^HasPopup

指出注释是否含有弹出框。

函数原型：

BOOL HasPopup()

参数：

无

返回值：

注释含有弹出框则返回 TRUE，反之则返回 FALSE。

8) ^GetRect

获取注释的矩形区域。

函数原型：

Long GetRect(float* pLeft, float* pTop, float* pRight, float* pBottom)

参数：

pLeft - 输出值，用以接收注释矩形区域左边的坐标。

pTop - 输出值，用以接收注释矩形区域顶边的坐标。
pRight - 输出值，用以接收注释矩形区域右边的坐标。
pBottom - 输出值，用以接收注释矩形区域底边的坐标。

返回值：

成功则返回 0，反之则返回-1。

9) ^SetRect

设置注释的矩形区域。

函数原型：

long SetRect(float Left, float Top, float Right, float Bottom)

参数：

Left - 输入值，注释矩形区域左边的坐标。
Top - 输入值，注释矩形区域顶边的坐标。
Right - 输入值，注释矩形区域右边的坐标。
Bottom - 输入值，注释矩形区域底边的坐标。

返回值：

成功则返回 0，反之则返回-1。

10) ^SetLinkGoToAction

设置链接注释的跳转动作。

函数原型：

Void SetLinkGoToAction(long nPageIndex, float left, float top, float zoom)

参数：

nPageIndex - 页码的输入值。
left - Windows 左上角的位置。
top - Windows 左上角的位置。
zoom - 页面的放大系数。

返回值：

无

11) ^SetLinkURLAction

设置链接注释的 URL 动作

函数原型：

Void SetLinkURLAction(LPCTSTR sURL)

参数：

sURL - 动作执行时跳转的位置。

返回值：

无

12) ^DoAction

执行链接注释的动作。

函数原型：

long DoAction()

参数：

无

返回值：

成功则返回 0，反之则返回-1。

13) ^HasAction

指出链接注释是否含有动作。

函数原型：

BOOL HasAction()

参数：

无

返回值：

如果链接注释含有动作则返回 TRUE，反之则返回 FALSE。

14) ^GetMarkedState

获取注释的标记状态。

函数原型：

Long GetMarkedState()

参数：

无

返回值：

未标记时返回值为 0，标记时返回值为 1，出现错误时返回值为-1。

15) ^SetMarkedState

设置注释的标记状态。

函数原型：

long SetMarkedState(long state)

参数：

state - 设置注释的标记状态。

0 = 未标记

1 = 标记

返回值：

成功则返回 0，反之则返回-1。

备注：

不支持图片、视频、音频和链接等注释。

16) ^GetReviewState

获取注释的审阅状态。

函数原型：

long GetReviewState()

参数：

无

返回值：

返回表示审阅状态的值，有效值如下。出现错误则返回值为-1。

0 = NULL

1 = 接受

2 = 拒绝

3 = 取消

4 = 完成

17) ^SetReviewState

设置注释的审阅状态。

函数原型：

long SetReviewState(long state)

参数：

State - 注释的审阅状态。有效值为：

0 = NULL

1 = 接受

2 = 拒绝

3 = 取消

4 = 完成

返回值：

成功则返回 0，反之则返回-1。

备注：

不支持图片、视频、音频和链接等注释。

18) ^GetMigrationState

获取注释的迁移状态。

函数原型：

long GetMigrationState()

参数：

无

返回值：

返回值表示注释不同的迁移状态：

0 = NULL

1 = 未确认

2 = 已确认

-1 = 其他

19) ^SetMigrationState

设置注释的迁移状态。

函数原型：

long SetMigrationState(long state)

参数：

state - 输入值，指定注释的迁移状态。该值可以为 0、1 或 2，分别指定下列状态：

0 = NULL

1 = 未确认

2 = 已确认

返回值：

成功则返回 0，反之则返回-1。

备注：

不支持图片、视频、音频和链接等注释。

20) ^SetStartingPoint

对于线条和箭头注释，设置它们的起点。

函数原型：

long SetStartingPoint(float PointX, float PointY)

参数：

PointX - 起点的 X 坐标。

PointY - 起点的 Y 坐标。

返回值：

成功则返回 0，反之则返回-1。

21) ^GetStartingPoint

对于线条和箭头注释，获取这些注释的起点。

函数原型：

long GetStartingPoint(float* PointX, float* PointY)

参数：

PointX - [输出值]获取起点的 X 坐标。

PointY - [输出值]获取起点的 Y 坐标。

返回值：

成功则返回 0，反之则返回-1。

22) ^SetEndingPoint

设置注释的终点。仅适用于线条和箭头注释。

函数原型：

long SetEndingPoint(float PointX, float PointY)

参数：

PointX - 终点的 X 坐标。

PointY - 终点的 Y 坐标。

返回值：

成功则返回 0，反之则返回-1。

23) ^GetEndingPoint

获得注释的终点。

函数原型：

Long GetEndingPoint(float* PointX, float* PointY)

参数：

PointX - [输出值]终点的 X 坐标。

PointY - [输出值]设置终点的 Y 坐标。

返回值：

成功则返回 0，反之则返回-1。

备注：

仅适用于线条和箭头注释。

24) ^SetMediaPoster

为含海报的注释（如影像注释和音像注释）设置海报（图片）。

函数原型：

long SetMediaPoster(LPCTSTR ImageFilePath)

参数：

ImageFilePath - 输入图片文件的路径。

返回值：

成功则返回 0，反之则返回-1。

25) ^SetMultimedia

设置支持多媒体注释的多媒体内容。

函数原型：

Long SetMultimedia (LPCTSTR FilePath, LPCTSTR ContentType, BOOL Embed, BOOL bShowCtrlBar)

参数：

FilePath - 媒体文件路径。

ContentType - 媒体数据的 MIME 类型。

Embed - 该值表示是否在 PDF 文件中嵌入媒体。

bShowCtrlBar - 该值表示是否显示控制条。

返回值：

成功则返回 0，反之则返回-1。

26) ^SetLinkQuadPoints

设置链接注释的外观。

函数原型：

long SetLinkQuadPoints(long* PointsArray, long PointsCount)

参数：

PointsArray - 点阵。

PointsCount - 点阵的大小应为 4。

返回值：

成功则返回 0， 反之则返回-1。

27) ^SetPolygonVertices

设置多边形注释的外观。

函数原型：

long SetPolygonVertices(long* PointsArray, long PointsCount)

参数：

PointsArray - 点阵。

PointsCount - 点阵的大小。

返回值：

成功则返回 0，反之则返回-1。

28) ^SetPencilVertices

设置铅笔注释的外观。

函数原型：

long SetPencilVertices(long* PointsArray, long PointsCount)

参数：

PointsArray - 一组线条集，每条线代表一个点阵。

PointCount - 点阵的大小。

返回值：

成功则返回 0，反之则返回-1。

29) ^AttachFile

为附件注释指定添加的文件。

函数原型：

long AttachFile(LPCTSTR FileName)

参数：

FileName - 输入文件的路径

返回值：

成功则返回 0，反之则返回-1。

30) ^ GetReplyList

获取注释回复。

函数原型：

IPDFAnnotReplyList* GetReplyList()

参数：

无

返回值：

成功则返回注释回复，否则则返回 NULL。

31) ^ UpdateAnnotReplies

更新注释回复。

函数原型：

Void UpdateAnnotReplies(IPDFAnnotReplyList* pReplies)

参数：

pReplies - 待更新的注释回复。

备注：

更新注释的所有回复。

32) ^ GetRectTop

获取注释对象的顶部坐标。

函数原型：

float GetRectTop()

参数：

无

返回值：

返回指定注释对象的顶部坐标。

33) ^GetRectLeft

获取指定注释对象的左边坐标。

函数原型：

float GetRectLeft()

参数：

无

返回值：

返回指定注释对象的左边坐标。

34) ^GetRectRight

获取指定注释对象的右边坐标。

函数原型：

float GetRectRight()

参数：

无

返回值：

返回指定注释对象的右边坐标。

35) ^GetRectBottom

获取指定注释对象的底边坐标。

函数原型：

float GetRectBottom()

参数：

无

返回值：

返回指定注释对象的底边坐标。

事件

1) ^OnAnnotCreated

创建注释后触发该事件，该事件由第三方完成。

函数原型：

void OnAnnotCreated(long pageIndex, long annotIndex)

参数：

pageIndex - 页面索引。

annotIndex - 注释索引。

返回值：

无

2) ^OnAnnotDeleted

删除注释后触发该事件，该事件由第三方完成。

函数原型：

void OnAnnotDeleted(long pageIndex, long annotIndex)

参数：

pageIndex - 页面索引值。

annotIndex - 注释索引值。

返回值：

无

3) ^OnAnnotModified

编辑注释后触发该事件，该事件由第三方完成。

函数原型：

void OnAnnotModified(long pageIndex, long annotIndex)

参数：

pageIndex - 页面索引值。

annotIndex - 注释索引值。

返回值：

无

4) ^OnAnnotReplyCreated

创建注释回复后触发该事件，该事件由第三方完成。

函数原型：

void OnAnnotReplyCreated(long pageIndex, long annotindex, lpctstr replyNM)

参数：

pageIndex - 页面索引值。

annotIndex - 注释索引值。

pReply - 创建的回复。

返回值：

无

5) ^OnAnnotReplyDeleted

删除注释回复后触发该事件，该事件由第三方完成。

函数原型：

void OnAnnotReplyDeleted(long pageIndex, long annotindex, CPDFAnnotReply* pReply)

参数：

pageIndex - 页面索引值。。

annotIndex - 注释索引值。

pReply - 待删除的注释回复。

返回值：

无

6) ^OnAnnotReplyModified

修改注释回复后触发该事件，该事件由第三方完成。

函数原型：

Void OnAnnotReplyModified(long pageIndex, long annotindex, CPDFAnnotReply*pReply)

参数：

pageIndex - 页面索引值。

annotIndex - 注释索引值。

pReply - 修改的注释回复。

返回值：

无

7) ^OnAnnotRButtonDown

右击注释时触发该事件

函数原型：

void OnAnnotRButtonDown(IPDFAnnot* Annot, float x, float y, BOOL* bDefault)

参数：

Annot - 右击选中的注释。

x - 注释的 x 坐标（PDF 坐标系）。

y - 注释的 y 坐标（PDF 坐标系）。

bDefault - 值为 True 则启用按钮的默认动作，反之则禁用按钮的默认动作。

返回值：

无

8) ^OnAnnotRButtonUp

释放右键按钮时触发该事件。

函数原型：

void OnAnnotRButtonUp(IPDFAnnot* Annot, float x, float y, BOOL* bDefault)

参数：

Annot - 右击选中的注释。

x - 注释的水平坐标（PDF 坐标系）。

y - 注释的垂直坐标（PDF 坐标系）。

bDefault - 值为 True 则启用按钮的默认动作，反之则禁用按钮的默认动作。

返回值：

无

9) ^OnAnnotLButtonDbClick

双击注释时触发该事件

函数原型：

void OnAnnotLButtonDbClick(IPDFAnnot* Annot, float x, float y, BOOL* bDefault)

参数：

Annot - 双击选中的注释。

x - 注释的 x 坐标（PDF 坐标系）。

y - 注释的 y 坐标（PDF 坐标系）。

bDefault - 值为 True 则启用按钮的默认动作，反之则禁用按钮的默认动作。

返回值：

无

10) ^OnAnnotMoving

移动注释时触发该事件。

函数原型：

void OnAnnotMoving(IPDFAnnot* Annot, float x, float y, BOOL* bDefault)

参数：

Annot - 待移动的注释。

x - 注释的 x 坐标（PDF 坐标系）。

y - 注释的 y 坐标（PDF 坐标系）。

bDefault - 值为 True 则启用按钮的默认动作，反之则禁用按钮的默认动作。

返回值：

无

11) ^OnAnnotMouseEnter

鼠标移至注释时触发该事件

函数原型：

void OnAnnotMouseEnter(IPDFAnnot* Annot)

参数：

Annot - 鼠标移至的注释。

返回值：

无

12) ^OnAnnotMouseExit

鼠标移出注释时触发该事件。

函数原型：

Void OnAnnotMouseExit(IPDFAnnot* Annot)

参数原型：

Annot - 鼠标移出的注释

返回值：

无

^ IPDFAnnotReplyList

方法

1) ^ GetCount

获取回复的数量。

函数原型：

long GetCount()

参数：

无

返回值：

返回回复的数量。

2) ^ GetItem

获取指定注释的回复。

函数原型：

CPDFAnnotReply GetItem(long nIndex)

参数：

nIndex - 回复的索引值。

返回值：

成功则返回指定的回复，反之则返回 NULL。

3) ^ Remove

删除指定的回复。

函数原型：

void Remove(long nIndex)

参数：

nIndex - 待删除的回复的索引。

返回值：

无

4) ^ RemoveAll

删除回复列表中所有的回复。

函数原型：

void RemoveAll()

参数：

无

返回值：

无

5) ^ Add

添加新的回复。

函数原型：

Void Add(LPCTSTR Creator, LPCTSTR Content, DATE CreationDate, long nIndex)

参数：

Creator	-	回复的创建者。
Content	-	回复的内容。
CreationDate	-	回复的创建时间。
nIndex	-	添加回复的位置（从 0 开始），若为-1，则在回复列表末尾添加回复。

返回值：

无

备注：

假设原回复列表中有 N 条回复，则 nIndex 的范围为[0，N-1]。

^IPDFAnnotReply

方法

1) ^ SetCreator

设置注释回复的创建者。

函数原型：

void SetCreator(LPCTSTR Creator)

参数：

Creator - 注释回复的创建者。

返回值：

无

2) ^ GetCreator

获取注释回复的创建者。

函数原型：

BSTR GetCreator()

参数：

无

返回值：

返回注释回复的创建者。

3) ^ SetContent

设置注释回复的内容。

函数原型：

void SetContent(LPCTSTR Content)

参数：

Content - 注释回复内容。

返回值：

无

4) ^ GetContent

获取注释回复的内容。

函数原型:

BSTR GetContent()

参数：

无

返回值:

返回注释回复的内容。

5) ^ GetChildren

获取子注释回复。

函数原型：

IPDFAnnotReplyList* GetChildren()

参数：

无

返回值：

成功则返回子注释回复，若无子回复，则返回 NULL。

6) ^ GetParent

获取父类注释回复。

函数原型：

CPDFAnnotReply GetParent()

参数：

无

返回值：

成功则返回父类注释回复，若无子回复，则返回 NULL。

7) ^ GetCreationDate

获取注释回复的创建时间。

函数原型：

DATE GetCreationDate()

参数：

无

返回值：

返回注释回复的创建时间。

8) ^ SetCreationDate

设置注释回复的创建时间。

函数原型：

void SetCreationDate(DATE CreationDate)

参数：

无

参数：

CreationDate - 注释回复的创建时间。

返回值：

无

9) ^ SetReadonly

将注释回复设置为只读型。

函数原型：

void SetReadonly(BOOL bNewValue)

参数：

bNewValue - 值为 TRUE 则设置注释回复为只读型，值为 FALSE 则为读/写型。

返回值：

无

10) ^ GetReplyID

获取注释回复的唯一 ID。

函数原型：

long GetReplyID()

参数：

无

返回值：

返回注释回复的唯一 ID。

^IPDFormatTool

IPDFormatTool 提供了接口，允许用户在 PDF 应用程序中设置 free text 类型注释的格式。ActiveX 控件支持三种 free text 类型，分别是 Typewriter、Callout 和 TextBox。

在修改当前 free text 注释的同时，对应的 CPDFFormatTool 的的数据也会被修改。IPDFFormatTool 类仅对 free text 类型注释有效。

方法

1) ^SetFontName

设置字体。

函数原型：

Void SetFontName(BSTR FontName)

参数：

FontName - 字体名称。

返回值：

无

2) ^GetFontName

获取当前使用的字体名称。

函数原型：

BSTR GetFontName()

参数：

无

返回值：

返回当前正在使用的字体名称。

3) ^SetFontSize

设置字号。

函数原型：

Void SetFontSize(float FontSize)

参数：

FontSize - 字号。

返回值：

无

4) ^GetFontSize

获取字号。

函数原型：

Float GetFontSize()

参数：

无

返回值：

返回当前正在使用的字号。

5) ^SetFontColor

设置当前字体的颜色。

函数原型：

Void SetFontColor(OLE_COLOR FontColor)

参数：

FontColor - 字体颜色。

返回值：

无

6) ^GetFontColor

获取字体的颜色。

函数原型：

OLE_COLOR GetFontColor()

参数：

无

返回值：

返回当前字体的颜色。

7) ^SetBorderColor

设置字体边框的颜色。

函数原型：

Void SetBorderColor(OLE_COLOR color)

参数：

Color - 字体边框的颜色。

返回值：

无

8) ^GetBorderColor

获取当前字体的边框颜色。

函数原型：

OLE_COLOR GetBorderColor()

参数：

无

返回值：

返回当前字体的边框颜色。

9) ^SetFillColor

设置当前字体的填充颜色。

函数原型：

Void SetFillColor(OLE_COLOR FillColor)

参数：

FillColor - 字体的填充颜色。

返回值：

无

10) ^GetFillColor

获取字体的填充颜色。

函数原型：

OLE_COLOR GetFillColor()

参数：

无

返回值：

返回字体的填充颜色。

11) ^SetFontBold

设置文本是否显示为粗体。

函数原型：

Void SetFontBold(BOOL FontBold)

参数：

FontBold - 该值表示文本是否显示为粗体。

返回值：

无

备注：

只有在当前字体是 Courier, Helvetica, Times Roman 时 ,bold 属性和 italic 属性才允许被设置。

12) ^GetFontBold

获取文本显示的状态是否为粗体。

函数原型：

BOOL GetFontBold()

参数：

无

返回值：

如果文本显示状态为粗体，则返回值等于 TRUE，反之则等返回 FALSE。

13) ^GetFontBoldEnable

指明格式化工具中是否启用了字体加粗功能。

函数原型：

BOOL GetFontBoldEnalbe()

参数：

无

返回值：

若已启用字体加粗功能，则返回 TRUE，反之则返回 FALSE。

14) ^SetFontItalic

设置文本是否显示为斜体。

函数原型：

Void SetFontItalic(BOOL FontItalic)

参数：

FontItalic - 该值表示文本是否显示为斜体。

TRUE 表示显示为斜体；

FALSE 表示不显示为斜体。

返回值：

无

15) ^GetFontItalic

获取文本显示的状态是否斜体。

函数原型：

BOOL GetFontItalic()

参数：

无

返回值：

文本显示为斜体则返回 TRUE，反之则返回 FALSE。

16) ^GetFontItalicEnable

设置文本字体为斜体。

函数原型：

BOOL GetFontItalicEnable()

参数：

无

返回值：

如果文本显示状态为斜体，则返回值等于 TRUE，反之则等返回 FALSE。

17) ^SetAlign

设置文本对齐的方式。

函数原型：

Void SetAlign(AlignStyle Style)

参数：

Style - 文本对齐方式。有效值如下：

ASLEFT =0, 左对齐

ASMIDDLE =1, 居中对齐

ASRIGHT =2, 右对齐

返回值：

无

18) ^GetAlign

获取文本对齐的方式。

函数原型：

AlignStyle GetAlign()

参数：

无

返回值：

返回文本对齐的方式。返回值为：

ASLEFT =0, 左对齐

ASMIDDLE =1, 居中对齐

ASRIGHT =2, 右对齐

19) ^SetCharSpace

设置字符间距。

函数原型：

Void SetCharSpace(float CharSpace)

参数：

CharSpace - 字符间距。

返回值：

无

20) ^GetCharSpace

获取字符间距。

函数原型：

float GetCharSpace()

参数：

无

返回值：

返回字符间距。

21) ^SetCharHorzScale

设置字符的水平缩放比率。

函数原型：

Void SetCharHorzScale(float CharHorzScale)

参数：

CharHorzScale - 水平拉伸比率。

返回值：

无

22) ^GetCharHorzScale

获取字符水平缩放比率。

函数原型：

float GetCharHorzScale()

参数：

无

返回值：

返回字符水平缩放比率。

&IPDFSignatureMgr

方法

1) &Add

添加未签名的签名域。

函数原型：

IPDFSignatureField* Add(long pageIndex, float left, float top, float right, float bottom)

参数：

pageIndex	-	PDF 中待添加签名域的页面索引值。
left	-	签名矩形区域起始位置的水平坐标。
top	-	签名矩形区域起始位置的垂直坐标。
right	-	签名矩形区域结束位置的水平坐标。
bottom	-	签名矩形区域结束位置的垂直坐标。

返回值：

新建的 IPDFSignatureField 对象。

备注：

左下方是 PDF 文件的原始点。

2) &SignDocument

对未签名的签名域进行签名，默认签名成功时会关闭源文档，并打开签名后的文档。

函数原型：

boolean SignDocument(LPDISPATCH pSigField, BSTR signedFilePath, boolean bDefault)

参数：

pSigField	-	待签名的 IPDFSignatureField 对象。
signedFilePath	-	签名后 PDF 文件的保存路径。
bDefault	-	该值表示是否使用默认签名。

返回值：

成功则返回 TRUE，否则返回 FALSE。

备注：

若使用默认的算法添加签名，则调用该接口来完成签名；

若不是使用默认的算法添加签名，则调用以下接口运行第三方签名工具来签署 PDF 文件：

- 首先，通过 GetSourceBuffer 和 GetSourceBufferLen 接口来获取待签署文件的内容流；
- 然后，使用第三方工具添加签名；
- 最后，调用 CreateSignedDoc 来完成签名。

运用标准算法成功添加签名后，PDF 文件将会重新打开，同时 IPDFSignatureField 对象也会失效。

如果您需要继续操作 IPDFSignatureField 对象，则需要再次获取该对象。

3) &Verify

验证签名域。

函数原型：

Boolean Verify(LPDISPATCH pSigField, boolean bShowStateDialog)

参数：

pSigField - 指定待验证的签名域。

bDefaultVerified - 指明是否采用默认的算法来验证签名域。值为 TRUE 则表示采用默认的算法。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

备注：

若使用默认的算法来添加签名，则调用该接口来完成 PDF 文件验证。

若使用第三方签名算法来添加签名，则在调用该接口后，调用以下接口来验证 PDF 文件：

- 首先，通过 GetSourceBuffer 和 GetSignedBuffer 接口验证所签署的文件；
- 然后，调用 SetVerifyResult 接口将验证结果传至 ActiveX；
- 最后，通过 IPDFSignatureField 的 GetState 接口获取验证结果。

4) &GetCounts

计算签名个数。

函数原型：

long GetCounts()

参数：

无

返回值：

签名域的总个数（包括已签名域与未签名域）。

5) &Get

通过索引值获取签名域。

函数原型：

LPDISPATCH Get(long index)

参数：

index - 通过指定索引获取签名域对象。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

6) &Clear

清空签名域，即把签名的相关信息(包括外观)进行清除，在页面上留下一个未签名的签名域对象。

函数原型：

Boolean Clear(LPDISPATCH pSigField)

参数：

pSigField - 待清空的 IPDFSignatureField 对象。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

7) &Remove

移除未签名的签名域，即删除一个为签名的签名域对象。

函数原型：

boolean Remove(LPDISPATCH pSigField)

参数：

pSigField - 需要移除的 PDFSignatField 对象。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

8) &VerifyAll

验证所有已签名的签名域。

函数原型：

boolean VerifyAll()

返回值：

成功则返回 TRUE，反之则返回 FALSE。

备注：

一般在刚开发文档时要求验证所有签名域对象的情况下使用该接口，这时，在默认情况下，无需弹出状态对话框。

9) &InitStraddleValue

使用骑缝签章。

函数原型：

BOOL InitStraddleValue(IPDFSignatureField* pSigField)

参数:

pSigField - 待签名的 IPDFSignatureField 对象。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

备注：

调用该接口后，调用正常的签章流程（包括标准签名和第三方签名）来添加骑缝章。

10) &CreatePatternSigField

创建签章模板。

函数原型：

IPDFSignatureField* CreatePatternSigField()

参数：

无

返回值：

返回签章域。

11) &SetCurPattenSigField

设置签章模块

函数原型：

boolean SetCurPattenSigField(short nIndex);

参数：

nIndex - 选择的模板编号，从 0 开始。

返回值：

设置模板成功返回 true，错误返回 flase.

&IPDFSignatureField

属性

1) &Reason

类型：

String

描述：

签名的原因。

操作：

可读可写

2) &Location

类型：

String

描述：

签名物理位置。

操作：

可读可写

3) &Signer

类型：

String

描述：

签名者。

操作：

可读可写

4) &Filter

类型：

String

描述：

签名域对象的签名算法 Filter 名称，默认的 Foxit 签名算法 Filter 为 Adobe.PPKLite。

操作：

可读可写

5) &SubFilter

类型：

String

描述：

签名域对象的签名算法 subFilter 名称，默认的 Foxit 签名算法的 SubFilter 为 adbe.pkcs7.detached。

6) &State

类型：

Short

描述：

签章当前状态。如果是默认签名时，该状态由 ActiveX 内部定义；否则，该值由应用定义并用 SetVerifyResult 传给 ActiveX。

0 = 未知签名

1 = 未签名

2 = 验证通过

3 = 验证未通过

操作：

只读

方法

1) &SetAPOptions

设置客户化外观的外观显示选项值。

函数原型：

boolean SetAPOptions(long opts)

参数：

opts - 客户化外观显示选项设置（0-511）。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

备注：

客户化外观提供了下面的显示选项可供用户选择，需在 SignDocument 之前调用。

全部不显示 - 0

显示全部 - 511

显示文本 - 0x100L

显示图片 - 0x080L

显示签名者 - 0x040L

显示位置 - 0x020L

显示 DN - 0x010L

显示时间 - 0x008L
显示原因 - 0x004L
显示标签 - 0x002L
显示 FoxitFlag - 0x001L

2) &SetAPText

设置签名域 AP 显示的文本。

函数原型：

Boolean SetAPText(BSTR text)

参数：

text - 设置显示的文本。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

备注：

需要在调用 SignDocument 之前调用。

3) &SetAPImage

设置签名域中显示的图片

函数原型：

boolean SetAPImage(BSTR imageFilePath, boolean bSetMask, OLE_COLOR clrMask)

参数：

imageFilePath - 签名域的图片文件。
bSetMask - 该参数数值指定是否给图片背景设置遮罩效果(注 :图片背景必须纯色的)。
clrMask - 设置 Mask 色值。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

备注：

需在 SignDocument 之前调用。

4) &IsSigned

判断签名域是否已经被签名。

函数原型：

boolean IsSigned()

参数：

无

返回值：

已签署则返回 TRUE，反之则返回 FALSE。

5) &SetSignerDN

设置证书识别名称。

函数原型：

boolean SetSignerDN(BSTR dn)

参数：

dn - 证书识别名称。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

备注：

需在 SignDocument 之前调用。

6) &SetStatusImage

设置签名状态图章样式。

函数原型：

Boolean SetStatusImage(BSTR imagePath, short sState, short sMode, Boolean bRotate, boolean bSetMask, OLE_COLOR clrMask)

参数：

imagePath - 状态图片的路径。

sState - 签章的状态，该值由属性 Status 决定。

sMode - 该值表示图片显示的模式。

0 为默认的图片显示位置（即签章左上角）；

1 为图片覆盖整个章区域显示。

bRotate - 该值表示图标是否随着签章对象旋转而旋转（默认为不旋转）。

bSetMask - 该值表示是否为图片背景色设置遮罩效果（注：图片背景必须是纯色的）。

clrMask - 设置 Mask 色值。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

备注：

设置的图片不会保存至 PDF，只有在 Signature 状态改变前设置有效。

7) &GetPageIndex

获取签名的页面索引值。

函数原型：

long GetPageIndex()

参数：

无

返回值：

获取签名的页面索引值。

8) &GetSourceBuffer

获取待签名或待验证时源文档内容流。

函数原型：

VARIANT GetSourceBuffer()

参数：

无

返回值：

源文档内容流。

备注：

调用该接口前先调用 SignDocument 或 Verify 并返回 True。

内容流存在 VARIVANT 的 parray 域中，vc 中使用 parray->pvData 获得内容流的指针；js 使用 getArray()获取内容流数组。

9) &GetSourceBufferLen

获取待签名时或待验证时源文档内容流长度。

函数原型：

long GetSourceBufferLen()

参数：

无

返回值：

内容流长度。

备注：

调用该接口前先调用 SignDocument 或 Verify 并返回 True。

10) &GetSignedBuffer

获取签名后的签名内容流。

函数原型：

VARIANT GetSignedBuffer()

参数：

无

返回值：

签名内容流。

备注：

调用该接口前先调用 Verify 并返回 True。

内容流存在 VARIVANT 的 parray 域中，vc 中使用 parray->pvData 获得内容流的指针；js 使用 getArray()获取内容流数组。

11) &GetSignedBufferLen

获取签名后签名内容流的长度。

函数原型：

long GetSignedBufferLen()

参数：

无

返回值：

返回内容流的长度。

备注：

调用该接口前先调用 Verify 并返回 True。

12) &CreateSignedDoc

第三方签名生成所需的签名文档（路径设置在 SignDocument 接口中）。

函数原型：

boolean CreateSignedDoc(VARIANT signedBuf, long length)

参数：

signedBuf - 签名后的内容流。

length - 签名后内容流的长度。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

备注：

调用该接口前需先调用 SignDocument 设置签名文档保存路径和不使用默认签名。

signedBuf 在 vc 使用指针存在 VARIANT 的 pbVal 域中，js 则使用数组对象作为参数。

签名成功后，PDF 会重新打开，PDFSignatureField 对象会失效，需要重新获取。

13) &SetVerifyResult

设置第三方验证结果。

函数原型：

Boolean SetVerifyResult(short sResult)

参数：

sResult - 用户自定义验证结果。

0 = 未知签名

1 = 验证通过

2 = 验证未通过

3 = 未签名

返回值：

成功则返回 TRUE，反之则返回 FALSE。

14) &SetCertPath

设置证书内容。

函数原型：

boolean SetCertPath(BSTR certPath, BSTR pfxPsw)

参数：

certPath - 证书路径。

pfxPsw - 证书密码。

返回值：

成功则返回 TRUE， 反之则返回 FALSE。

备注：

pCertData 在 vc 使用指针存在 VARIANT 的 pbVal 域中，js 则使用数组对象作为参数。

15) &SetCertData

设置证书内容

函数原型：

boolean SetCertData(VARIANT pCertData, long length, BSTR pfxPsw)

参数：

pCertData - 证书内容流。

length - 内容流字节数。

pfxPsw - 证书密码。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

备注：

pCertData 在 vc 使用指针存在 VARIANT 的 pbVal 域中，js 则使用数组对象作为参数。

16) &SetCertContext

设置证书上下文。

函数原型：

boolean SetCertContext(long pCertContext)

参数：

pCertContext - 证书上下文类型 PCCERT_CONTEXT。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

备注：

&SetCertPath，&SetCertData，&SetCertContext 这三个接口只要设置其中一个，证书就可生效。

17) &SetAPImageData

设置签名域外观显示图像的字节流。

函数原型：

boolean SetAPImageData(VARIANT imageDataBuffer, BSTR imageType, long dataSize, boolean bSetMask, OLE_COLOR clrMask)

参数：

imageDataBuffer - 图片字节流。

imageType - 图片类型。

dataSize - 字节流长度。

bSetMask - 该属性值表示是否遮罩。

clrMask - 遮罩颜色。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

备注：

pCertData 在 vc 使用指针存在 VARIANT 的 pbVal 域中，js 则使用数组对象作为参数。

18) &SetStatusImageData

设置签名域外观图片的字节流。

函数原型：

Boolean SetStatusImageData(VARIANT imageDataBuffer, BSTR imageType, long dataSize, short sState, short sMode, boolean bRotate, boolean bSetMask, OLE_COLOR clrMask)

参数：

imageDataBuffer - 图片字节流。

imageType - 图片类型；可支持的类型包括：bmp、jpg、png 和 gif。

dataSize - 字节流长度。

sState - 签章状态，该值由属性 Status 决定。

sMode - 图片显示的模式。

0 为默认的图片显示位置（即为签章左上角）;

1 为图片覆盖整个章区域显示

bRotate - 该属性值表示是否图标随着签章对象旋转而旋转（默认不旋转）

bSetMask - 该属性值表示是否需要对图片背景色做遮罩（图片背景必须是纯色）

clrMask - Mask 色值。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

备注：

imageDataBuffer 在 vc 使用指针存在 VARIANT 的 pbVal 域中，js 则使用数组对象作为参数。

19) &TurnGray

签章显示灰化。

函数原型：

Boolean TurnGray(boolean bGray,boolean bCanModify)

参数：

bGray - 是否灰化。

bCanModify - 是否可以修改。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

20) &TurnBlur

签章显示雾化。

函数原型：

Boolean TurnBlur(boolean bBlur)

参数：

bBlur - 该值表示是否雾化。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

备注：

该接口只在控件中改变签章的显示，未改变 PDF 内容流。

21) &SetVisible

签章是否显示。

函数原型：

Boolean SetVisible(boolean bVisible);

参数：

bVisible - 该值表示是否显示签章。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

22) &GrayPrint

签章灰化打印。

函数原型：

Boolean GrayPrint(boolean bGrayPrint)

参数：

bGrayPrint - 该值表示是否灰化打印。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

备注：

该接口只在控件中改变签章的显示，未改变 PDF 内容流。

23) &SetStraddleType

设置骑缝章的类型。

函数原型：

BOOL SetStraddleType(short nType)

参数：

nType - 骑缝章的类型，有效值为 0, 1, 2, 3 和 4：

0 = 中骑缝章

1 = 左骑缝章

2 = 右骑缝章

3 = 上骑缝章

4 = 下骑缝章

返回值：

成功则返回 TRUE，反之则返回 FALSE。

24) &SetStraddlePos

设置骑缝章的位置。

函数原型：

BOOL SetStraddlePos(float fPos)

参数：

fPos - 若骑缝章位于顶端或底端，则该参数值为签名域的水平中间点；反之，则为签名域的垂直中间点。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

25) &SetStraddleBitmap

根据签名状态设置骑缝章的外观图片。

函数原型：

BOOL SetStraddleBitmap(short nState, BSTR sFilePath)

参数：

nState - 骑缝章的状态，有效值为：

0 - 表示未知状态；

1 - 表示未签署；

2 - 表示有效；

3 - 表示无效。

sFilePath - 图片文件的路径。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

26) &SetStraddlePages

设置签名显示的页码范围。

函数原型：

BOOL SetStraddlePages(BSTR sRange)

参数：

sRange - 骑缝章所在的 PDF 页的页码范围；例如，“0-10” 或 “0-2,3-10”。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

27) &SetStraddleFirstPagePercent

设置骑缝章所在的首个 PDF 页面上骑缝章外观的百分比。

函数原型：

BOOL SetStraddleFirstPagePercent(float fPercent)

参数：

fPercent - 骑缝章所在的首个 PDF 页面上骑缝章外观的百分比，有效值为 0~1。

返回值：

成功则返回 TRUE，反之则返回 FALSE。

28) &SetSigFieldAlpha

设置签章图片的透明度。

函数原型：

boolean SetSigFieldAlpha(short alpha)

参数：

alpha - 签章图片的透明度，有效值为 1~255。图片透明度设置是叠加模式，在原有的透明上叠加。

返回值：

成功则返回 TRUE，否则返回 FALSE。

29) &Refresh

刷新签章域。

函数原型：

void Refresh()

参数：

无

返回值：

无

备注：

此接口只有在签章未完成之前有效。

事件

1) &OnSetSignatureInfo

单击未签名的签名域或右击选择签名时触发该事件。在该事件中设置 PDFSignatureField 签名必要的属性完成签名。

函数原型：

void OnSetSignatureInfo(IPDFSignatureField* pSignature)

参数：

pSignature - 单击或右击的签名域对象。

2) &OnSigning

若对象未签名会继续触发该事件，在该事件中进行默认或第三方签名。

函数原型：

void OnSigning(IPDFSignatureField* pSignature)

参数：

pSignature - 当前单击的签名域对象。

3) &OnVerifying

单击已签名对象或右键选择验证时触发该事件，在该事件中进行默认或自定义验证。

函数原型：

void OnVerifying(IPDFSignatureField* pSignature)

参数：

pSignature - 当前单击的签名域对象

4) &OnShowSignaturePropertyDialog

右击签名域对象选择查看签名属性时触发该事件。

函数原型：

Void OnShowSignaturePropertyDialog (IPDFSignatureField*pSignature, boolean* bShowProperty)

参数：

pSignature - 当前右击的签名域对象。

bShowProperty - 该值表示是否弹出签名域的属性对话框。

*IPDFTextDoc

方法

1) *ReleasePDFTextDoc

释放 IPDFTextDoc 占用的所有资源。

函数原型：

void ReleasePDFTextDoc()

参数：

无

返回值：

无

2) *LoadPDFTextPage

初始化 PDF 页面所有文本字符的信息。

函数原型：

IPDFTextPage* LoadPDFTextPage(long pageIndex)

参数：

pageIndex - PDF 页的索引值，有效值从 0 开始。

返回值：

一个 IPDFTextPage 对象。

*IPDFTextPage

方法

1) *ReleasePDFTextPage

释放 IPDFTextPage 占用的所有资源。

函数原型：

void ReleasePDFTextPage()

参数：

无

返回值：

无

2) *CountChars

获取 PDF 页面中的文本字符数量。

函数原型：

long CountChars()

参数：

无

返回值：

PDF 页面中包含的文本字符个数。

3) *GetChars

获取 PDF 页面中指定范围内的文本内容。

函数原型：

BSTR GetChars(long start, long count)

参数：

start - 起始字符的位置。范围是 0 到 CountChars-1。

count - 期望获取的字符数量 ;-1 表示获取所有的字符。若 count>(CountChars-start)，则表示获取 start 之后所有剩余的字符。

返回值：

一个文本字符串。

4) *GetCharInfo

获取指定字符的信息。

函数原型：

IPDFCharInfo* GetCharInfo(long charIndex)

参数：

charIndex - 字符的索引值，有效值范围为 0~(CountChars-1)。

返回值：

一个 IPDFCharInfo 对象。

5) *GetCharIndexAtPos

获取 PDF 页面中指定位置上或指定位置附近的字符的索引值。

函数原型：

long GetCharIndexAtPos (float x, float y, float tolerance)

参数：

x - x 坐标（用户空间）。

y - y 坐标（用户空间）。

tolerance - 与指定位置的容差。单位为像素，且必须是非负数。

返回值：

在指定位置上或指定位置附近的字符的索引值。

6) *GetNextCharIndexByDirection

在指定的方向上，获取下一个字符的索引值。

函数原型：

long GetNextCharIndexByDirection(long curIndex, short direction)

参数：

curIndex - 当前字符的索引值。有效值从 0 开始计算。

direction - 获取下一个字符的索引值的方向。有效值是：

-1 = 当前字符的左侧。

1 = 当前字符的右侧。

-2 = 当前字符的上方。

2 = 当前字符的下方。

返回值：

返回下一个字符的索引值。有效值从 0 开始。以下三种情况表示错误：

-1 = 表示已到页面顶端。

-2 = 表示已到页面底部。

-3 = 表示存在其他未知的错误。

7) *SelectByRange

将指定范围的字符转换为一个 IPDFTextSelection 对象。

函数原型：

IPDFTextSelection* SelectByRange (long start, long count)

参数：

start - 起始字符的索引值。有效值从 0 开始。

count - 计划获取的字符数量。-1 表示获取所有的字符。

返回值：

一个 IPDFTextSelection 对象。

8) *SelectByRectangle

将指定矩形区域内的字符转换为一个 IPDFTextSelection 对象。

函数原型：

IPDFTextSelection* SelectByRectangle (long left, long top, long right, long bottom)

参数：

left - 矩形区域的左侧。

top - 矩形区域的顶部。

right - 矩形区域的右侧。

bottom - 矩形区域的底部。

返回值：

一个 IPDFTextSelection 对象。

9) *StartSearch

准备开始一个 PDF 文本查找操作。

函数原型：

IPDFTextSearch* StartSearch(BSTR searchPattern, long flags, long startIndex)

参数：

searchPattern - 进行查找的文本关键字。

flags - 文本查找的设置。可以是以下数值的一个或多个的组合。

0 = 没有限制。

1 = 区别大小写。

2 = 全字匹配。

4 = Consecutive。

startIndex - 查询的起始位置。从 0 开始计算。-1 表示从页面尾部开始查询。

返回值：

成功则返回 IPDFTextSearch。

10) *ExtractLinks

提取 PDF 页面中 URL 格式的内容。

函数原型：

IPDFTextLink* ExtractLinks ()

参数：

无

返回值：

一个 IPDFTextLink 对象。

11) *ExtractPageText

提取 PDF 页面中的文本内容。

函数原型：

BSTR ExtractPageText ()

参数：

无

返回值：

文本字符串。

*IPDFCharInfo

属性

1) *state

类型：

short，只读。

描述：

字符的状态。有效值为：

1 = Normal character.

2 = Character is generated by Foxit, such as space character.

3 = Character doesn't have its own unicode value.

2) *fontSize

类型：

float，只读。

描述：

字符的字号。单位为 1/72 英寸。

3) *originX

类型：

float，只读。

描述：

基于 PDF 页面坐标系，字符的 X 值。-1 表示错误。

4) *originY

类型：

float，只读。

描述：

基于 PDF 页面坐标系，字符的 Y 值。

5) *fontName

类型：

BSTR，只读。

描述：

字体名称。

6) *fontAscent

类型：

long，只读。

描述：

从基线（baseline）到字符顶部的距离。

7) *fontDescent

类型：

long，只读。

描述：

从基线（baseline）到字符底部的距离。

方法

1) *GetBBox

获取字符的范围。

原型：

VARIANT GetBBox()

参数：

无

返回值：

一个包含四个浮点数的数组，表示一个矩形区域。

0 = 左侧

1 = 顶部

2 = 右侧

3 = 底部

2) *GetMatrix

获取字符的矩阵变换。

原型：

VARIANT GetMatrix()

参数：

无

返回值：

一个包含 6 个浮点数的数组，表示一个矩阵变换。

0 = a

1 = b

2 = c

3 = d

4 = e

5 = f

*IPDFTextSelection

方法

1) *ReleaseTextSelection

释放 IPDFTextSelection 占用的所有资源。

原型：

void ReleaseTextSelection()

参数：

无

返回值：

无

2) *GetBBox

获取 IPDFTextSelection 所包含的范围（一个矩形区域）。

原型：

VARIANT GetBBox()

参数：

无

返回值：

一个包含四个浮点数的数组，表示一个矩形区域。

0 = 左侧

1 = 顶部

2 = 右侧

3 = 底部

3) *GetChars

从 IPDFTextSelection 中提取所有的文本字符。

原型：

BSTR GetChars()

参数：

无

返回值：

一个文本字符串。

4) *CountPieces

获取 IPDFTextSelection 中包含的矩形区域的数量，一个矩形区域包含一段文本。（常用于跨行选

择)

原型：

long CountPieces ()

参数：

无

返回值：

IPDFTextSelection 中包含的矩形区域的数量。

5) *GetPieceRect

获取 IPDFTextSelection 中指定的矩形区域。

原型：

VARIANT GetPieceRect(long pieceIndex)

参数：

pieceIndex - 从 0 开始，范围由 CountPieces 返回值决定。

返回值：

一个包含四个浮点数的数组，表示一个矩形区域。

0 = 左侧

1 = 右侧

2 = 顶部

3 = 底部

6) *GetPieceCharStart

获取 IPDFTextSelection 中指定矩形区域范围内的第一个字符的位置。

原型：

long GetPieceCharStart(long pieceIndex)

参数：

pieceIndex - 矩形区域的索引值。有效值从 0 开始，范围由 CountPieces 返回值决定。

返回值：

指定矩形区域范围内的第一个字符的位置。-1 表示错误。

7) *GetPieceCharCount

获取 IPDFTextSelection 中指定矩形区域范围内的字符数量。

函数原型：

long GetPieceCharCount(long pieceIndex)

参数：

pieceIndex - 矩形区域的索引值。有效值从 0 开始，范围由 CountPieces 返回值决定。

返回值：

指定矩形区域范围内的字符数量。-1 表示错误。

*IPDFTextSearch

方法

1) *ReleaseTextSearch

释放 IPDFTextSearch 占用的所有资源。

函数原型：

void ReleaseTextSearch()

参数：

无

返回值：

无

2) *FindNext

从页头向后查找下一个与条件匹配的结果。

函数原型：

BOOL FindNext ()

参数：

无

返回值：

成功则返回 TRUE，否则返回 FALSE。

3) *FindPrev

从页尾向前查找上一个与条件匹配的结果。

函数原型：

BOOL FindPrev ()

参数：

无

返回值：

成功则返回 TRUE，否则返回 FALSE。

4) *GetSelection

从当前符合条件的搜索结果中获取 IPDFTextSelection 对象。

函数原型：

IPDFTextSelection* GetSelection ()

参数：

无

返回值：

一个 IPDFTextSelection 对象。

*IPDFTextLink

方法

1) *CountLinks

获取 PDF 页面中符合 URL 格式的数量。

函数原型：

long CountLinks()

参数：

无

返回值：

返回符合 URL 格式的个数。-1 表示错误。

2) *GetLink

获取与指定的超链接相关联的 URL 地址。

函数原型：

BSTR GetLink (long linkIndex)

参数：

linkIndex - 超链接的索引值。有效值从 0 开始，范围由 CountLinks 的返回值决定。

返回值：

返回一个 URL 地址。

3) *GetSelection

从指定的超链接中获取 IPDFTextSelection 对象。

函数原型：

IPDFTextSelection*GetSelection (long linkIndex)

参数：

linkIndex - 超链接的索引值。有效值从 0 开始，范围由 CountLinks 的返回值决定。

返回值：

一个 IPDFTextSelection 对象。

联系我们

如果您需要了解更多信息或对我们的产品有任何疑问，请随时联系我们，我们将竭诚为您服务。

- **福建福昕软件开发股份有限公司**

地址： 福州市鼓楼区软件园 G 区 5 号楼
总机： 0591-38509898
传真： 0591-38509708
邮编： 350003

- **福建福昕软件开发股份有限公司北京分公司**

地址： 北京市海淀区知春路 56 号中海实业大厦 9 层
电话： 010-50951668
传真： 010-50951666
邮编： 100098

- **福建福昕软件开发股份有限公司南京分公司**

地址： 江苏省南京市中山东路 532－ 2 号金蝶科技园 D 栋 601 室
电话： 025-84866095 84866195
传真： 025-84866295
邮编： 210000

- **福建福昕软件开发股份有限公司台湾联络处**

地址：新北市淡水區自強路 9 號 1 樓
电话： +886-2-2809-2969
传真： +886-2-2808-6566
邮编： 25162

- **电子邮件：**

邮箱咨询 - sales@foxitsoftware.cn
商务合作 - BD@foxitsoftware.cn
市场服务 - marketing@foxitsoftware.cn
客服支持 -请点击在线支持系统提交您的需求或问题
媒体关系 - PR@foxitsoftware.cn
网站问题 - webmaster@foxitsoftware.com
网址： www.foxitsoftware.cn