Adobe Acrobat 7.0.5

Acrobat Interapplication Communication Reference

October 7, 2005

Adobe Solutions Network — http://partners.adobe.com
### Preface
Description: 15
Audience: 15
Prerequisites: 16
Related Documents:
  - Other Interapplication Communication Documentation: 16
  - Core API Documentation: 16
  - File Format Documentation: 16
Conventions Used in This Book: 17

### Apple Events

#### Chapter 1 Apple Event Objects
- annotation: 21
- application: 23
- AVPageView: 26
- bookmark: 27
- conversion: 29
- document: 30
- EPS Conversion: 32
- Link Annotation: 33
- menu: 34
- menu item: 35
- page: 36
- PDAnnot: 37
- PDBookMark: 37
- PDLinkAnnot: 37
- PDPage: 37
- PDTextAnnot: 37
- PDF Window: 38
Chapter 2  Apple Events  ........................................ 43

Postscript Conversion................................................. 40
Text Annotation......................................................... 41

Chapter 2  Apple Events  ........................................ 43

Required Suite.......................................................... 43
  open........................................................................ 43
  print....................................................................... 44
  quit....................................................................... 45
  run........................................................................ 46

Core Suite................................................................. 47
  close...................................................................... 47
  count..................................................................... 48
  delete.................................................................... 49
  exists.................................................................... 50
  get......................................................................... 51
  make...................................................................... 52
  move...................................................................... 53
  open...................................................................... 54
  quit....................................................................... 55
  save. ..................................................................... 56
  set. ....................................................................... 57

Acrobat application Suite............................................. 58
  bring to front........................................................ 58
  clear selection...................................................... 59
  close all docs........................................................ 60
  create thumbs....................................................... 61
  delete pages.......................................................... 62
  delete thumbs....................................................... 63
  execute................................................................... 64
  find next note....................................................... 65
  find text.................................................................. 66
  get info................................................................... 67
  go backward.......................................................... 68
  go forward............................................................ 69
  goto....................................................................... 70
  goto next............................................................... 71
  goto previous......................................................... 72
  insert pages............................................................ 73
  is toolbutton enabled........................................... 74
  maximize................................................................ 75
  perform.................................................................. 76
  print pages............................................................. 77
  read page down..................................................... 78
  read page up.......................................................... 79
## OLE Automation

### Chapter 3  OLE Automation Objects ................................. 89

<table>
<thead>
<tr>
<th>Class</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcroExch.App</td>
<td>89</td>
</tr>
<tr>
<td>AcroExch.AVDoc</td>
<td>89</td>
</tr>
<tr>
<td>AcroExch.AVPageView</td>
<td>89</td>
</tr>
<tr>
<td>AcroExch.Hilite</td>
<td>89</td>
</tr>
<tr>
<td>AcroExch.PDAannot</td>
<td>89</td>
</tr>
<tr>
<td>AcroExch.PDBBookmark</td>
<td>90</td>
</tr>
<tr>
<td>AcroExch.PDDoc</td>
<td>90</td>
</tr>
<tr>
<td>AcroExch.PDPage</td>
<td>90</td>
</tr>
<tr>
<td>AcroExch.PDTextSelect</td>
<td>90</td>
</tr>
<tr>
<td>AxAcroPDFLib.AxAcroPDF</td>
<td>91</td>
</tr>
</tbody>
</table>

### Chapter 4  OLE Automation Methods .................................. 93

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AxAcroPDFLib.AxAcroPDF</td>
<td>91</td>
</tr>
<tr>
<td>AcroExch.App</td>
<td>93</td>
</tr>
<tr>
<td>CloseAllDocs</td>
<td>93</td>
</tr>
<tr>
<td>Exit</td>
<td>94</td>
</tr>
<tr>
<td>GetActiveDoc</td>
<td>95</td>
</tr>
<tr>
<td>GetActiveDoc</td>
<td>95</td>
</tr>
<tr>
<td>GetActiveTool</td>
<td>96</td>
</tr>
<tr>
<td>GetAVDoc</td>
<td>97</td>
</tr>
<tr>
<td>GetFrame</td>
<td>98</td>
</tr>
<tr>
<td>GetInterface</td>
<td>99</td>
</tr>
<tr>
<td>GetLanguage</td>
<td>100</td>
</tr>
<tr>
<td>GetNumAVDocs</td>
<td>101</td>
</tr>
<tr>
<td>GetPreference</td>
<td>102</td>
</tr>
<tr>
<td>GetPreferenceEx</td>
<td>103</td>
</tr>
<tr>
<td>Hide</td>
<td>104</td>
</tr>
<tr>
<td>Lock</td>
<td>105</td>
</tr>
<tr>
<td>Minimize</td>
<td>106</td>
</tr>
<tr>
<td>Maximize</td>
<td>107</td>
</tr>
<tr>
<td>MenuItemExecute</td>
<td>108</td>
</tr>
<tr>
<td>MenuItemIsEnabled</td>
<td>109</td>
</tr>
</tbody>
</table>
GetZoomType .......................................................... 158
Goto ....................................................................... 159
PointToDevice .......................................................... 160
ReadPageDown ........................................................ 161
ReadPageUp .............................................................. 162
ScrollTo ..................................................................... 163
ZoomTo ..................................................................... 164
AcroExch.HiliteList .................................................. 165
Add ......................................................................... 165
AcroExch.PDAnnot .................................................... 166
GetColor ................................................................. 166
GetContents ............................................................. 167
GetDate ................................................................. 168
GetRect ................................................................. 169
GetSubtype ............................................................. 170
GetTitle ..................................................................... 171
IsEqual ...................................................................... 172
isOpen ....................................................................... 173
IsValid ...................................................................... 174
Perform ...................................................................... 175
SetColor ..................................................................... 176
SetContents ............................................................ 177
SetDate ................................................................. 178
SetOpen ...................................................................... 179
SetRect ...................................................................... 180
SetTitle ..................................................................... 181
AcroExch.PDBookmark .............................................. 182
Destroy ...................................................................... 182
GetByTitle .............................................................. 183
GetTitle ..................................................................... 184
IsValid ...................................................................... 185
Perform ...................................................................... 186
SetTitle ..................................................................... 187
AcroExch.PDDoc ......................................................... 188
AcquirePage ............................................................ 188
ClearFlags .................................................................. 189
Close ......................................................................... 190
Create ....................................................................... 191
CreateTextSelect ..................................................... 192
CreateThumbs .......................................................... 193
CropPages .................................................................. 194
DeletePages ............................................................. 195
DeleteThumbs ........................................................... 196
GetFileName ............................................................ 197
GetFlags ..................................................................... 198
GetInfo ................................................................. 199
GetInstanceID ....................................................... 200
GetJSObject .......................................................... 201
GetNumPages .......................................................... 202
GetPageMode .......................................................... 203
GetPermanentID ......................................................... 204
InsertPages ............................................................. 205
MovePage ............................................................... 206
Open ................................................................. 207
OpenAVDoc ............................................................. 208
ReplacePages .......................................................... 209
Save ................................................................. 210
SetFlags ............................................................... 211
SetInfo ............................................................... 212
SetPageMode .......................................................... 213

**AcroExch.PDPage** .................................................. 214
AddAnnot .............................................................. 214
AddNewAnnot .......................................................... 215
CopyToClipboard ...................................................... 216
CreatePageHilite ...................................................... 217
CreateWordHilite ...................................................... 218
CropPage .............................................................. 219
Draw ................................................................. 220
DrawEx ............................................................... 221
GetAnnot .............................................................. 222
GetAnnotIndex .......................................................... 223
GetDoc ............................................................... 224
GetNumAnnots .......................................................... 225
GetNumber ........................................................... 226
GetRotate ............................................................. 227
GetSize ............................................................... 228
RemoveAnnot ........................................................... 229
SetRotate ............................................................. 230

**AcroExch.PDTextSelect** .......................................... 232
Destroy .............................................................. 232
GetBoundingRect ....................................................... 233
GetNumText ........................................................... 234
GetPage ............................................................. 235
GetText ............................................................. 236

**AxAcroPDFLib.AxAcroPDF** ......................................... 237
GetVersions ........................................................... 237
GoBackwardStack ...................................................... 238
GoForwardStack ....................................................... 239
GotoFirstPage .......................................................... 240
GotoLastPage .......................................................... 241
### Chapter 5 OLE Automation Properties

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>GotoNextPage</td>
<td>242</td>
</tr>
<tr>
<td>GotoPreviousPage</td>
<td>243</td>
</tr>
<tr>
<td>LoadFile</td>
<td>244</td>
</tr>
<tr>
<td>Print</td>
<td>245</td>
</tr>
<tr>
<td>PrintAll</td>
<td>246</td>
</tr>
<tr>
<td>PrintAllFit</td>
<td>247</td>
</tr>
<tr>
<td>PrintPages</td>
<td>248</td>
</tr>
<tr>
<td>PrintPagesFit</td>
<td>249</td>
</tr>
<tr>
<td>PrintWithDialog</td>
<td>250</td>
</tr>
<tr>
<td>SetCurrentHighlight</td>
<td>251</td>
</tr>
<tr>
<td>SetCurrentPage</td>
<td>252</td>
</tr>
<tr>
<td>SetLayoutMode</td>
<td>253</td>
</tr>
<tr>
<td>SetNamedDest</td>
<td>254</td>
</tr>
<tr>
<td>SetPageMode</td>
<td>255</td>
</tr>
<tr>
<td>SetShowScrollbars</td>
<td>256</td>
</tr>
<tr>
<td>SetShowToolbar</td>
<td>257</td>
</tr>
<tr>
<td>SetView</td>
<td>258</td>
</tr>
<tr>
<td>SetViewRect</td>
<td>259</td>
</tr>
<tr>
<td>SetViewScroll</td>
<td>260</td>
</tr>
<tr>
<td>SetZoom</td>
<td>261</td>
</tr>
<tr>
<td>SetZoomScroll</td>
<td>262</td>
</tr>
<tr>
<td>PrintWithDialog</td>
<td>263</td>
</tr>
</tbody>
</table>

**AcroExch.Rect**

<table>
<thead>
<tr>
<th>Property</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bottom</td>
<td>267</td>
</tr>
<tr>
<td>Left</td>
<td>268</td>
</tr>
<tr>
<td>Right</td>
<td>269</td>
</tr>
<tr>
<td>Top</td>
<td>270</td>
</tr>
</tbody>
</table>

**AcroExch.Time**

<table>
<thead>
<tr>
<th>Property</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>271</td>
</tr>
<tr>
<td>Hour</td>
<td>272</td>
</tr>
<tr>
<td>Millisecond</td>
<td>273</td>
</tr>
<tr>
<td>Minute</td>
<td>274</td>
</tr>
<tr>
<td>Month</td>
<td>275</td>
</tr>
<tr>
<td>Second</td>
<td>276</td>
</tr>
<tr>
<td>Year</td>
<td>277</td>
</tr>
</tbody>
</table>

**AxAcroPDFLib.AxAcroPDF**

<table>
<thead>
<tr>
<th>Property</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src.</td>
<td>278</td>
</tr>
</tbody>
</table>
## DDE

### Chapter 6  DDE Messages  

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppExit</td>
<td>281</td>
</tr>
<tr>
<td>AppHide</td>
<td>282</td>
</tr>
<tr>
<td>AppShow</td>
<td>283</td>
</tr>
<tr>
<td>CloseAllDocs</td>
<td>284</td>
</tr>
<tr>
<td>DocClose</td>
<td>285</td>
</tr>
<tr>
<td>DocDeletePages</td>
<td>286</td>
</tr>
<tr>
<td>DocFind</td>
<td>287</td>
</tr>
<tr>
<td>DocGoTo</td>
<td>288</td>
</tr>
<tr>
<td>DocGoToNameDest</td>
<td>289</td>
</tr>
<tr>
<td>DocInsertPages</td>
<td>290</td>
</tr>
<tr>
<td>DocOpen</td>
<td>291</td>
</tr>
<tr>
<td>DocPageDown</td>
<td>292</td>
</tr>
<tr>
<td>DocPageLeft</td>
<td>293</td>
</tr>
<tr>
<td>DocPageRight</td>
<td>294</td>
</tr>
<tr>
<td>DocPageUp</td>
<td>295</td>
</tr>
<tr>
<td>DocPrint</td>
<td>296</td>
</tr>
<tr>
<td>DocReplacePages</td>
<td>297</td>
</tr>
<tr>
<td>DocSave</td>
<td>298</td>
</tr>
<tr>
<td>DocSaveAs</td>
<td>299</td>
</tr>
<tr>
<td>DocScrollTo</td>
<td>300</td>
</tr>
<tr>
<td>DocSetViewMode</td>
<td>301</td>
</tr>
<tr>
<td>DocZoomTo</td>
<td>302</td>
</tr>
<tr>
<td>FileOpen</td>
<td>303</td>
</tr>
<tr>
<td>FileOpenEx</td>
<td>304</td>
</tr>
<tr>
<td>FilePrint</td>
<td>305</td>
</tr>
<tr>
<td>FilePrintEx</td>
<td>306</td>
</tr>
<tr>
<td>FilePrintSilent</td>
<td>307</td>
</tr>
<tr>
<td>FilePrintSilentEx</td>
<td>308</td>
</tr>
<tr>
<td>FilePrintTo</td>
<td>309</td>
</tr>
<tr>
<td>FilePrintToEx</td>
<td>310</td>
</tr>
<tr>
<td>FullMenus</td>
<td>311</td>
</tr>
<tr>
<td>HideToolbar</td>
<td>312</td>
</tr>
<tr>
<td>MenuItemExecute</td>
<td>313</td>
</tr>
<tr>
<td>ShortMenus</td>
<td>314</td>
</tr>
<tr>
<td>ShowToolbar</td>
<td>315</td>
</tr>
</tbody>
</table>

## Plug-Ins

### Chapter 7  Acrobat Catalog  

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acrobat Catalog</td>
<td>319</td>
</tr>
</tbody>
</table>
Chapter 13  Acroform Automation Properties .................................................. 353

Field ........................................................................................................... 353
  Alignment ................................................................................................. 353
  BorderStyle ............................................................................................. 354
  BorderWidth ............................................................................................ 355
  ButtonLayout ........................................................................................... 356
  CalcOrderIndex ......................................................................................... 357
  CharLimit .................................................................................................. 358
  DefaultValue ............................................................................................ 359
  Editable ..................................................................................................... 360
  Highlight ................................................................................................... 361
  IsHidden ................................................................................................... 362
  IsMultiline ................................................................................................. 363
  IsPassword ............................................................................................... 364
  IsReadOnly ............................................................................................... 365
  IsRequired ............................................................................................... 366
  IsTerminal ................................................................................................. 367
  Name ......................................................................................................... 368
  NoViewFlag ............................................................................................. 369
  PrintFlag ................................................................................................... 370
  Style ......................................................................................................... 371
  TextFont ................................................................................................... 372
  TextSize .................................................................................................... 373
  Type .......................................................................................................... 374
  Value ........................................................................................................ 375

Fields ......................................................................................................... 376
  Count .......................................................................................................... 376
  Item .......................................................................................................... 377
  _NewEnum ............................................................................................... 378

Chapter 14  Acrobat Search ................................................................. 379

Contents .................................................................................................... 380

Chapter 15  Search DDE Messages .................................................. 381
Preface

The Acrobat® Software Development Kit (SDK) provides a set of Acrobat Core API calls for creating plug-ins and other programs. You may use a subset of these calls for implementing interapplication (IAC) functionality and PDF browser controls. These Acrobat calls support Apple® Events (including the use of AppleScript), Microsoft® OLE automation, and DDE interapplication interfaces.

For more information, see http://partners.adobe.com/asn/.

Description

This document provides a detailed reference of all the calls needed for Apple Events, OLE and DDE. You need only read the section that applies to the interface with which you are working. Each section has the following structure:

- **Description.** A complete description of the syntax and any other related information.
- **Object descriptions**, if applicable.
- **Event, message, or method descriptions.** Detailed descriptions of each item.
- **IAC-specific information.** Description of associated declarations, constants, or any other relevant details. Use these items with any of the supported interfaces.

**NOTE:** There is no IAC support for the UNIX versions of Acrobat. There is no IAC support in the Japanese version of Acrobat.

**NOTE:** See the Acrobat and PDF Library API Reference for information on Acrobat application constants such as tool and menu names (formerly in an appendix to this document).

Audience

If you are writing plug-ins that need to communicate with or use multiple applications, you should read this document.
Prerequisites

You should already be familiar with at least one of these technologies:

- Apple events
- AppleScript
- DDE
- OLE

If you are not, see the list of documents that describe them in “Related Documents”.

You should also be familiar with the Acrobat core API. Many of the IAC capabilities are actually a subset of those provided in the Acrobat core API, and many of the IAC messages are similar to core API methods.

Related Documents

The Acrobat SDK includes many other books that you might find useful. If for some reason you did not install the entire SDK onto your system and you do not have all of the documentation, please visit the Adobe Solutions Network web site (http://partners.adobe.com/asn/) to find the books you need.

Developer Documentation

The Acrobat SDK User’s Guide describes the capabilities of the Acrobat SDK, and provides a general overview of its usage.

Developing for Adobe Reader provides an introduction to those portions of the Adobe Acrobat Software Development Kit (SDK) that pertain to your development efforts for Adobe Reader.

Other Interapplication Communication Documentation

Acrobat Interapplication Communication Overview provides overview information on the Apple Event, DDE, and OLE support in Acrobat applications.

Core API Documentation

Acrobat and PDF Library API Overview provides an overview of the objects and methods in the Acrobat core API.

Acrobat and PDF Library API Reference contains detailed descriptions of the objects, methods and callbacks in the Acrobat core API.
File Format Documentation

*PDF Reference* provides a description of the PDF file format, as well as guidelines for producing efficient PDF files.

Platform-Specific Documentation


---

Conventions Used in This Book

The Acrobat documentation uses text styles according to the following conventions.

<table>
<thead>
<tr>
<th>Font</th>
<th>Used for</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>monospaced</td>
<td>Paths and filenames</td>
<td>C:\templates\mytmpl.fm</td>
</tr>
<tr>
<td>monospaced</td>
<td>Code examples set off from plain text</td>
<td>These are variable declarations:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AVMenu commandMenu, helpMenu;</td>
</tr>
<tr>
<td>monospaced bold</td>
<td>Code items within plain text</td>
<td>The <strong>GetExtensionID</strong> method ...</td>
</tr>
<tr>
<td></td>
<td>Parameter names and literal values in</td>
<td>The enumeration terminates if <strong>proc</strong> returns <strong>false</strong>.</td>
</tr>
<tr>
<td></td>
<td>reference documents</td>
<td></td>
</tr>
<tr>
<td>monospaced italic</td>
<td>Pseudocode</td>
<td><strong>ACCB1</strong> void <strong>ACCB2</strong> <strong>ExeProc</strong>(void) { do <em>something</em> }</td>
</tr>
<tr>
<td></td>
<td>Placeholders in code examples</td>
<td><strong>APSimple_Calculate</strong>(cFunction, cFields)</td>
</tr>
<tr>
<td>Font</td>
<td>Used for</td>
<td>Examples</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>blue</td>
<td>Live links to Web pages</td>
<td>The Adobe Solutions Network URL is: <a href="http://partners.adobe.com/asn/">http://partners.adobe.com/asn/</a></td>
</tr>
<tr>
<td></td>
<td>Live links to sections within this document</td>
<td>See Using the SDK.</td>
</tr>
<tr>
<td></td>
<td>Live links to code items within this document</td>
<td>Test whether an ASA.com exists.</td>
</tr>
<tr>
<td>bold</td>
<td>PostScript® language and PDF operators, keywords, dictionary key names</td>
<td>The <code>setpagedevice</code> operator</td>
</tr>
<tr>
<td></td>
<td>User interface names</td>
<td>The <strong>File</strong> menu</td>
</tr>
<tr>
<td>italic</td>
<td>Document titles that are not live links</td>
<td>Acrobat Core API Overview</td>
</tr>
<tr>
<td></td>
<td>New terms</td>
<td><code>User space</code> specifies coordinates for...</td>
</tr>
<tr>
<td></td>
<td>PostScript variables</td>
<td><code>filename deletfile</code></td>
</tr>
</tbody>
</table>
This reference contains the following sections:

**Apple Event Objects.** This section describes each object in the Apple Event interface and lists its elements, properties, and methods to which it responds.

**Apple Events.** Each Apple Event description includes information for its usage within AppleScript. In addition, the descriptions of Acrobat-specific events contain information for using them in a programming language. If you are using AppleScript, ignore the "Apple Event ID" and "Apple Event Parameters" information. For information about other Apple Event constants used in Acrobat, consult the header file AcroAETypes.h. See the header file AERegistry.h (or *The Apple Event Registry: Standard Suites*) for a list of the constants in the required and core event suites.
The object and event descriptions have the following conventions.

● Object Descriptions
  The abbreviation r/o is used for properties that are read-only.

● Event Descriptions
  All AppleScript examples use the English dialect of AppleScript syntax.
  Optional items are enclosed in square brackets [ ].
  Each AppleScript code sample assumes that it is being executed within an appropriate tell — end tell construct, as in this example:
    
    tell application "Acrobat 7.0"
    ...sample code here...
    end tell
This chapter details Apple Event Objects, with descriptions of each object’s elements and properties.

**annotation**

**Description**

An annotation on a page in a PDF file that corresponds to Acrobat’s internal `PDAnnot` class.

Acrobat has two built-in annotation types: **Link Annotation** and **Text Annotation**.

**NOTE:** This object was formerly known as `PDAnnot`.

**Elements**

None.

**Plural form**

Annotations

**Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>best type</td>
<td>type class [r/o]</td>
<td>The best descriptor type.</td>
</tr>
<tr>
<td>bounds</td>
<td>a list of small real</td>
<td>The boundary rectangle for the annotation in PDF space (left, top, right, bottom).</td>
</tr>
<tr>
<td>class</td>
<td>type class [r/o]</td>
<td>The class.</td>
</tr>
<tr>
<td>color</td>
<td>'RGB'</td>
<td>The color of the border around the annotation.</td>
</tr>
<tr>
<td>contents</td>
<td>international text</td>
<td><strong>Text annotations only:</strong> The textual contents of the note.</td>
</tr>
<tr>
<td>default type</td>
<td>type class [r/o]</td>
<td>The default descriptor type.</td>
</tr>
<tr>
<td>destination page number</td>
<td>integer</td>
<td><strong>Link annotations only:</strong> The page number to appear in the PDF window when the annotation link is activated.</td>
</tr>
</tbody>
</table>
### Property Class Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>destination rectangle</td>
<td>a list of small real</td>
<td><em>Link annotations only:</em> The boundary rectangle (specified in user space) for the view of the destination. Coordinates are specified in the following order: (left, top, right, bottom).</td>
</tr>
<tr>
<td>fit type</td>
<td>constant</td>
<td><em>Link annotations only:</em> Determines how the destination rectangle is fitted to the window when the link is activated. Values are: Left Top Zoom, Fit Page, Fit Width, Fit Height, Fit Rect, Fit BBox, Fit BB Width, Fit BB Height. These are described in the PDF Reference.</td>
</tr>
<tr>
<td>index</td>
<td>integer [r/o]</td>
<td>The annotation's index within the page object.</td>
</tr>
<tr>
<td>modification date</td>
<td>date</td>
<td>The date and time the annotation was last modified.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td><em>Text annotations only:</em> The annotation's label.</td>
</tr>
<tr>
<td>open state</td>
<td>boolean</td>
<td><em>Text annotations only:</em> Whether the annotation is open.</td>
</tr>
<tr>
<td>subtype</td>
<td>international text [r/o]</td>
<td>The subtype of the annotation.</td>
</tr>
<tr>
<td>zoom factor</td>
<td>small real</td>
<td><em>Link annotations only:</em> If fit type is Left Top Zoom, this specifies the zoom factor; otherwise it is ignored. Setting this property automatically sets fit type to Left Top Zoom.</td>
</tr>
</tbody>
</table>

### Related Methods

- delete
- perform
application

Description

The Acrobat or Adobe Reader application itself.

Elements

<table>
<thead>
<tr>
<th>Elements</th>
<th>Can be accessed by</th>
</tr>
</thead>
<tbody>
<tr>
<td>document</td>
<td>name, numeric index</td>
</tr>
<tr>
<td>PDF Window</td>
<td>name, numeric index</td>
</tr>
<tr>
<td>menu</td>
<td>name, numeric index</td>
</tr>
<tr>
<td>menu item</td>
<td>name</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active doc</td>
<td>reference</td>
<td>The active document.</td>
</tr>
<tr>
<td>active tool</td>
<td>international</td>
<td>The type of the currently active tool. See the Acrobat and PDF Library API</td>
</tr>
<tr>
<td></td>
<td>text</td>
<td>Reference for a list of tool names.</td>
</tr>
<tr>
<td>anti_alias text</td>
<td>boolean</td>
<td>Determines whether to anti-alias text and monochrome images.</td>
</tr>
<tr>
<td>best type</td>
<td>type class [r/o]</td>
<td>The best descriptor type.</td>
</tr>
<tr>
<td>case sensitivity</td>
<td>boolean</td>
<td>Determines whether searches are case-sensitive.</td>
</tr>
<tr>
<td>class</td>
<td>type class [r/o]</td>
<td>The class.</td>
</tr>
<tr>
<td>default type</td>
<td>type class [r/o]</td>
<td>The default descriptor type.</td>
</tr>
<tr>
<td>default zoom factor</td>
<td>small real</td>
<td>The default zoom factor, in percent, used for displaying new documents. For example, a value of 100 corresponds to a zoom factor of 1.0 (100%).</td>
</tr>
<tr>
<td>default zoom type</td>
<td>constant</td>
<td>The default zoom type when opening a new document. Valid values are &quot;no vary&quot;, &quot;fit page&quot;, &quot;fit width&quot;, &quot;fit height&quot;, and &quot;fit visible width.&quot;</td>
</tr>
<tr>
<td>Property</td>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>download entire file</td>
<td>boolean</td>
<td>Determines whether to download the entire file.</td>
</tr>
<tr>
<td>frontmost</td>
<td>boolean</td>
<td>Determines whether Acrobat is the frontmost application. Value can be set to true only.</td>
</tr>
<tr>
<td>fullscreen click advances</td>
<td>boolean</td>
<td>Determines whether mouse click advances in fullscreen mode.</td>
</tr>
<tr>
<td>fullscreen cursor</td>
<td>boolean</td>
<td>Determines whether to hide the cursor in fullscreen mode.</td>
</tr>
<tr>
<td>fullscreen escape</td>
<td>boolean</td>
<td>Determines whether the &lt;Esc&gt; key can be used to exit fullscreen mode.</td>
</tr>
<tr>
<td>fullscreen loop</td>
<td>boolean [r/o]</td>
<td>Determines whether the document's pages are displayed in a loop while in fullscreen mode.</td>
</tr>
<tr>
<td>fullscreen timer delay</td>
<td>integer</td>
<td>The number of seconds to advance to the next page in fullscreen mode.</td>
</tr>
<tr>
<td>fullscreen transition</td>
<td>international text [r/o]</td>
<td>Default fullscreen transition.</td>
</tr>
<tr>
<td>highlight color</td>
<td>'RGB '</td>
<td>Color used to highlight selections.</td>
</tr>
<tr>
<td>maximum documents</td>
<td>integer [r/o]</td>
<td>Maximum number of open documents.</td>
</tr>
<tr>
<td>name</td>
<td>string [r/o]</td>
<td>The application's name.</td>
</tr>
<tr>
<td>note color</td>
<td>'RGB '</td>
<td>A list of three values between 0 and 65535 representing the color of the border around text annotations. The following example sets the note color to deep blue: set the note color to {0, 0, 32768}.</td>
</tr>
<tr>
<td>note font name</td>
<td>international text</td>
<td>NOTE: Deprecated.</td>
</tr>
<tr>
<td>note font size</td>
<td>integer</td>
<td>NOTE: Deprecated.</td>
</tr>
<tr>
<td>open in place</td>
<td>boolean</td>
<td>Determines whether to open cross-document links in the same window.</td>
</tr>
<tr>
<td>page layout</td>
<td>international text</td>
<td>Default page layout. Values are: Single Page, Continuous, Facing, and Continuous - Facing.</td>
</tr>
<tr>
<td>Property</td>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>page units</td>
<td>international text</td>
<td>Default page display units: Points, Inches or Millimeters</td>
</tr>
<tr>
<td>PS level</td>
<td>integer</td>
<td>Note: Deprecated. Set PostScript level when using save or print pages commands.</td>
</tr>
<tr>
<td>save as linearize</td>
<td>boolean</td>
<td>Determines whether to save the document as a linearized file. Primarily used to optimize document viewing in a web browser.</td>
</tr>
<tr>
<td>show splash at startup</td>
<td>boolean</td>
<td>Determines whether the splash screen is shown at startup.</td>
</tr>
<tr>
<td>skip warnings</td>
<td>boolean</td>
<td>Determines whether to skip warning dialog boxes during program execution.</td>
</tr>
<tr>
<td>shrink to fit</td>
<td>boolean</td>
<td>Note: Deprecated.</td>
</tr>
<tr>
<td>text note label</td>
<td>international text</td>
<td>The text that will appear in the title bar of all newly created text notes.</td>
</tr>
<tr>
<td>toolbar visibility</td>
<td>boolean</td>
<td>Determines whether the toolbar is visible.</td>
</tr>
<tr>
<td>UI language</td>
<td>international text [r/o]</td>
<td>A three-character language code identifying which language is used in the Acrobat user interface. Example: ENU represents English.</td>
</tr>
<tr>
<td>use fullscreen timer</td>
<td>boolean</td>
<td>Determines whether to use a timer to advance pages in fullscreen mode</td>
</tr>
<tr>
<td>version</td>
<td>string [r/o]</td>
<td>The version number of the application.</td>
</tr>
<tr>
<td>whole word searching</td>
<td>boolean</td>
<td>Determines whether searches are applied to whole words only.</td>
</tr>
</tbody>
</table>

**Related Methods**

- close all docs
- count
- make
- open
- print
- quit
- run
AVPageView

**NOTE:** This object has been deprecated and is only shown for backward compatibility. Use PDF Window now.
bookmark

Description
A bookmark on a page in a PDF file. Corresponds to Acrobat’s **PDBookmark** object.

**Note:** This object was formerly known as **PDBookmark**. That name is obsolete; use this object.

Elements
None.

Plural form
Bookmarks.

Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>best type</td>
<td>type class [r/o]</td>
<td>The best descriptor type.</td>
</tr>
<tr>
<td>class</td>
<td>type class [r/o]</td>
<td>The class.</td>
</tr>
<tr>
<td>default type</td>
<td>type class [r/o]</td>
<td>The default descriptor type.</td>
</tr>
<tr>
<td>destination page number</td>
<td>integer</td>
<td>The page number to which the PDF Window goes when the bookmark’s action is performed.</td>
</tr>
<tr>
<td>destination rectangle</td>
<td>list of small real</td>
<td>Boundary rectangle (specified in user space) for the view of the destination when the bookmark’s action is performed. Coordinates are specified in the following order: (left, top, right, bottom).</td>
</tr>
</tbody>
</table>

**Note:** Set this only after setting **fit type**.
<table>
<thead>
<tr>
<th>Property</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fit type</td>
<td>constant</td>
<td>Controls how the destination rectangle is fitted to the window when the bookmark's action is performed. Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Left Top Zoom</strong> — Sets a specified zoom and a specified location on the page.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Fit Page</strong> — Sets the zoom factor so that the entire page fits into the window.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Fit Width</strong> — Sets the zoom factor so that the width of the page fits into the window.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Fit Height</strong> — Sets the zoom factor so that the height of the page fits into the window.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Fit Rect</strong> — Sets the zoom factor so that the specified rectangle fits into the window.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Fit BBox</strong> — Sets the zoom so that the rectangle enclosing all marks on the page (known as the bounding box) fits into the window.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Fit BB Width</strong> — Sets the zoom factor so that the width of the bounding box fits into the window.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Fit BB Height</strong> — Sets the zoom factor so that the height of the bounding box fits into the window.</td>
</tr>
<tr>
<td>index</td>
<td>integer [r/o]</td>
<td>The bookmark's index within the document.</td>
</tr>
<tr>
<td>name</td>
<td>international text</td>
<td>The bookmark's title.</td>
</tr>
<tr>
<td>zoom factor</td>
<td>small real</td>
<td>The zoom factor used when fit type is Left Top Zoom; ignored otherwise. Setting this property automatically sets fit type to Left Top Zoom.</td>
</tr>
</tbody>
</table>

**Related Methods**

- insert pages
- perform
conversion

Description
A file type converter that exports PDF files into other formats. Conversions correspond to the list of formats specified in Acrobat’s **Save As** menu. A list of formats may be obtained as follows:

get every conversion

Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>best type</td>
<td>type class [r/o]</td>
<td>The best descriptor type.</td>
</tr>
<tr>
<td>class</td>
<td>type class [r/o]</td>
<td>The class.</td>
</tr>
<tr>
<td>default type</td>
<td>type class [r/o]</td>
<td>The default descriptor type.</td>
</tr>
<tr>
<td>index</td>
<td>integer [r/o]</td>
<td>The index number of the converter.</td>
</tr>
<tr>
<td>name</td>
<td>international text</td>
<td>The conversion’s description.</td>
</tr>
</tbody>
</table>

Related Methods

**save**
**document**

**Description**

Represents a single open document in Acrobat or Adobe Reader.

**Elements**

<table>
<thead>
<tr>
<th>Element</th>
<th>Can be accessed</th>
</tr>
</thead>
<tbody>
<tr>
<td>page</td>
<td>by numeric index. The first page in a document is page 1.</td>
</tr>
<tr>
<td>bookmark</td>
<td>by name or numeric index.</td>
</tr>
<tr>
<td>PDF Window</td>
<td>No document has more than one PDF Window, so it may be</td>
</tr>
<tr>
<td></td>
<td>accessed by using an index of 1 or via the <code>some</code> keyword</td>
</tr>
<tr>
<td></td>
<td>in AppleScript.</td>
</tr>
</tbody>
</table>

**Plural form**

documents.

**Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>best type</td>
<td>type class [r/o]</td>
<td>The best descriptor type.</td>
</tr>
<tr>
<td>bounds</td>
<td>bounding rectangle [r/o]</td>
<td>The boundary rectangle for the document’s window, in screen coordinates (left, top, right, bottom).</td>
</tr>
<tr>
<td>class</td>
<td>type class [r/o]</td>
<td>The class.</td>
</tr>
<tr>
<td>default type</td>
<td>type class [r/o]</td>
<td>The default descriptor type.</td>
</tr>
<tr>
<td>file alias</td>
<td>alias [r/o]</td>
<td>An alias for the file to which the document will be saved if no other name is specified; this is usually the same path from which the document was read.</td>
</tr>
<tr>
<td>modified</td>
<td>boolean [r/o]</td>
<td>Determines whether the document has been modified and should be saved.</td>
</tr>
<tr>
<td>name</td>
<td>international text [r/o]</td>
<td>The document’s name as it appears in the window’s titlebar.</td>
</tr>
<tr>
<td>view mode</td>
<td>constant</td>
<td>The viewing mode of the document. Possible values: just pages, pages and thumbs, or pages and bookmarks.</td>
</tr>
</tbody>
</table>
Related Methods

- bring to front
- clear selection
- close
- count
- create thumbs
- delete
- delete pages
- delete thumbs
- find next note
- find text
- get info
- insert pages
- maximize
- print pages
- replace pages
- save
- set info
**EPS Conversion**

**Description**
A file type converter that exports PDF files into Encapsulated PostScript format.

**Properties**
Inherits from Postscript Conversion.

**Related Methods**
save
Link Annotation

Description
A link annotation on a page in a PDF file. Can only be used as the target of a `make` event. All other access is via the `annotation` class.

**Note:** This object was formerly known as `PDLinkAnnot`.

Elements
None.

Properties
Inherits from `annotation`.

Related Methods
- `delete`
- `perform`
menu

Description
A menu in Acrobat or Adobe Reader’s menu bar.

Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Can be accessed by</th>
</tr>
</thead>
<tbody>
<tr>
<td>menu item</td>
<td>name, numeric index.</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>best type</td>
<td>type class [r/o]</td>
<td>The best descriptor type.</td>
</tr>
<tr>
<td>class</td>
<td>type class [r/o]</td>
<td>The class.</td>
</tr>
<tr>
<td>default type</td>
<td>type class [r/o]</td>
<td>The default descriptor type.</td>
</tr>
<tr>
<td>name</td>
<td>international text[r/o]</td>
<td>The menu’s name (a language-independent name that uniquely identifies the menu). See the Acrobat And PDF Library API Reference for a list of menu names.</td>
</tr>
<tr>
<td>title</td>
<td>string [r/o]</td>
<td>The menu’s title as it would appear in the user interface.</td>
</tr>
</tbody>
</table>

Related Methods
execute
**menu item**

**Description**

A menu item contained within a menu in Acrobat or Adobe Reader.

**Elements**

None.

**Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>best type</td>
<td>type class [r/o]</td>
<td>The best descriptor type.</td>
</tr>
<tr>
<td>class</td>
<td>type class [r/o]</td>
<td>The class.</td>
</tr>
<tr>
<td>default type</td>
<td>type class [r/o]</td>
<td>The default descriptor type.</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean [r/o]</td>
<td>Determines whether the menu item is enabled.</td>
</tr>
<tr>
<td>has submenu</td>
<td>boolean [r/o]</td>
<td>Determines whether the menu item has a hierarchical sub-menu.</td>
</tr>
<tr>
<td>marked</td>
<td>boolean [r/o]</td>
<td>Determines whether the menu item is checked.</td>
</tr>
<tr>
<td>name</td>
<td>international text [r/o]</td>
<td>The menu item's language-independent name. See the Acrobat And PDF Library API Reference for a list of menu item names.</td>
</tr>
<tr>
<td>title</td>
<td>string [r/o]</td>
<td>The menu's title as it would appear in the user interface.</td>
</tr>
</tbody>
</table>

**Related Methods**

*execute*
**page**

**Description**
A single page in the PDF representation of a document. Corresponds to Acrobat’s internal **PDPage** object.

**NOTE:** This object was formerly known as **PDPage**.

**Elements**

<table>
<thead>
<tr>
<th>Element</th>
<th>Can be accessed by</th>
</tr>
</thead>
<tbody>
<tr>
<td>annotation</td>
<td>numeric index.</td>
</tr>
</tbody>
</table>

**Plural form**
Pages.

**Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>best type</td>
<td>type class [r/o]</td>
<td>The best descriptor type.</td>
</tr>
<tr>
<td>bounds</td>
<td>list of small real</td>
<td>The boundary rectangle for the page in user space (left, top, right, bottom).</td>
</tr>
<tr>
<td>class</td>
<td>type class [r/o]</td>
<td>The class.</td>
</tr>
<tr>
<td>default type</td>
<td>type class [r/o]</td>
<td>The default descriptor type.</td>
</tr>
<tr>
<td>page number</td>
<td>integer [r/o]</td>
<td>The page's number. The first page in a document is page 1.</td>
</tr>
<tr>
<td>rotation</td>
<td>integer</td>
<td>The rotation angle of the page in degrees (0, 90, 180, or 270).</td>
</tr>
</tbody>
</table>

**Related Methods**
- delete pages
- insert pages
- replace pages
- goto
- move
PDAnnot

NOTE: Deprecated. Use annotation now.

PDBookMark

NOTE: Deprecated. Use bookmark now.

PDLinkAnnot

NOTE: Deprecated. Use Link Annotation now.

PDPage

NOTE: Deprecated. Use page now.

PDTextAnnot

NOTE: Deprecated. Use Text Annotation now.
PDF Window

Description

The area of Acrobat or Adobe Reader's window that displays the contents of a page within the document. Corresponds to Acrobat's internal `AvPageView` object. Documents that are not visible don't have PDF Windows.

**NOTE:** This object was formerly known as `AvPageView`.

Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Can be accessed by...</th>
</tr>
</thead>
<tbody>
<tr>
<td>page</td>
<td>numeric index. The first page in a document is page 1.</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>best type</td>
<td>type class [r/o]</td>
<td>The best descriptor type.</td>
</tr>
<tr>
<td>bounds</td>
<td>bounding rectangle</td>
<td>The boundary rectangle for the window.</td>
</tr>
<tr>
<td>class</td>
<td>type class [r/o]</td>
<td>The class.</td>
</tr>
<tr>
<td>default type</td>
<td>type class [r/o]</td>
<td>The default descriptor type.</td>
</tr>
<tr>
<td>document</td>
<td>document [r/o]</td>
<td>The document that owns this window.</td>
</tr>
<tr>
<td>index</td>
<td>integer</td>
<td>The number of the window.</td>
</tr>
<tr>
<td>name</td>
<td>international text [r/o]</td>
<td>The document’s name as shown in the window’s titlebar.</td>
</tr>
<tr>
<td>page number</td>
<td>integer</td>
<td>The number of the currently displayed page.</td>
</tr>
<tr>
<td>position</td>
<td>point [r/o]</td>
<td>The upper left coordinates of the window.</td>
</tr>
<tr>
<td>visible</td>
<td>boolean [r/o]</td>
<td>Whether the window is visible.</td>
</tr>
<tr>
<td>zoomed</td>
<td>boolean</td>
<td>Whether the window is zoomed.</td>
</tr>
<tr>
<td>zoom factor</td>
<td>small real</td>
<td>The current zoom factor specified as a percentage. For example, a value of 100 corresponds to a zoom factor of 1.0 (100%).</td>
</tr>
<tr>
<td>zoom type</td>
<td>constant</td>
<td>The zooming and content fitting algorithm currently employed. Possible values: no vary, fit page, fit width, fit height, and fit visible width.</td>
</tr>
</tbody>
</table>
Related Methods

- go backward
- go forward
- goto
- goto next
- goto previous
- read page down
- read page up
- scroll
- select text
- zoom
Postscript Conversion

Description
A file type converter that exports PDF files into PostScript format.

Properties
Inherits other properties from conversion.

<table>
<thead>
<tr>
<th>Property</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>annotations</td>
<td>boolean</td>
<td>Determines whether to include annotations.</td>
</tr>
<tr>
<td>binary</td>
<td>boolean</td>
<td>Determines whether the output file should be in binary or ASCII text format.</td>
</tr>
<tr>
<td>embedded fonts</td>
<td>boolean</td>
<td>Determines whether to include fonts.</td>
</tr>
<tr>
<td>halftones</td>
<td>boolean</td>
<td>Determines whether to halftone screens.</td>
</tr>
<tr>
<td>images</td>
<td>boolean</td>
<td>Determines whether to include RGB and LAB images.</td>
</tr>
<tr>
<td>postScript level</td>
<td>integer</td>
<td>The PostScript Language level. Only levels 2 and 3 are supported.</td>
</tr>
<tr>
<td>preview</td>
<td>boolean</td>
<td>Determines whether to include preview in output.</td>
</tr>
<tr>
<td>TrueType</td>
<td>boolean</td>
<td>Determines whether to convert TrueType fonts to Type 1.</td>
</tr>
</tbody>
</table>

Related Methods
save
Text Annotation

Description

A PDF text annotation (note) on a page in a PDF file. Can only be used as the target of a make event. All other access is via the annotation class.

**NOTE:** This object was formerly known as TextAnnot.

Elements

None.

Properties

Inherits from annotation.

Related Methods

- find next note
- perform
- replace pages
Required Suite

This section details the Apple events in Acrobat’s Required Suite: **open**, **quit**, **print** and **run**.

**NOTE:** Most of these have counterparts in the Core suite that have greater functionality. The Required Suite is not listed in the AppleScript dictionary, even though it is implemented.

---

**open**

**Description**

Opens a file.

**AppleScript Syntax**

```apple
open reference
```

**AppleScript Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td>The file or files to open.</td>
</tr>
</tbody>
</table>

**Return Value**

None
**print**

**Description**

Prints one or more files.

**AppleScript Syntax**

```
print reference
```

**AppleScript Parameters**

| **print** | The file or files to print. |

**Return Value**

None
quit

Description
Terminates an application. See the `quit` event in the Core suite for a variant that accepts options.

AppleScript Syntax
```applescript
quit
```

AppleScript Parameters

None

Return Value

None
run

Description
Launches the application and invokes its standard startup procedures.

AppleScript Syntax
run

AppleScript Parameters

Return Value
None
Core Suite

This section details the Apple events in Acrobat’s Core Suite.

close

Description
Closes a document.

AppleScript Syntax

close reference [saving constant] [linearize boolean]

AppleScript Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>close</td>
<td>The document to close.</td>
</tr>
</tbody>
</table>
| saving    | Determines whether to save a document that has been modified before quitting. Possible values:
  yes — Save the document.
  no — Do not save the document.
  ask — Ask the user whether to save the document.
The default value is ask. |
| linearize | Determines whether the document should be linearized when saving before closing. |

Return Value
None

Related Events
open
count

Description
Counts the number of instances of a particular class.

AppleScript Syntax
```
count type class of reference
```

AppleScript Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>The class whose instances are to be counted.</td>
</tr>
<tr>
<td>each</td>
<td>The class whose instances are to be counted.</td>
</tr>
</tbody>
</table>

**NOTE:** The keyword `each` is optional.

AppleScript Example
```
count annotation of document "dev_acro.pdf"
count menu item of menu "View"
count document 1 each bookmark
```
delete

Description
Deletes one or more objects.

AppleScript Syntax
delete reference

AppleScript Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>delete</td>
<td>The object to be deleted.</td>
</tr>
</tbody>
</table>

Return Value
None

Related Events
make
eexists

AppleScript Example
delete first bookmark of document "test.pdf"
exists

Description
Tests whether a specified object exists.

AppleScript Syntax
reference exists
exists reference

AppleScript Parameters

| exists   | Object whose existence is checked. |

Return Value
true if the object exists, false otherwise.

AppleScript Example
exists second document
second document exists
get

Description
Retrieves the value of an object or property.

AppleScript Syntax

```
[get] reference [as class]
```

**NOTE:** The keyword `get` is optional.

AppleScript Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>get</code></td>
<td>The object or property whose value is returned.</td>
</tr>
<tr>
<td><code>as</code></td>
<td>The form in which the data is returned.</td>
</tr>
</tbody>
</table>

Return Value
The value of the specified property or object. If the specified object does not exist, no result is returned.

Related Events
```
set
```

AppleScript Example
```
get the name of last bookmark
get the index of last bookmark as string
```
make

Description

Creates a new object.

AppleScript Syntax

```
make [new] type class [at location reference ] [with data anything ] [with properties record]
```

AppleScript Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>make [new]</td>
<td>The class of the new object.</td>
</tr>
<tr>
<td>at</td>
<td>The location at which to insert the new object.</td>
</tr>
<tr>
<td>with data</td>
<td>The initial data for the new object.</td>
</tr>
<tr>
<td>with properties</td>
<td>The initial values for the properties of the new object.</td>
</tr>
</tbody>
</table>

Return Value

A reference to the newly created object.

Related Events

del**ete

exists

AppleScript Example

```
set myAnnot to make TextAnnotation at beginning
set name of myAnnotation to "Werner Heisenberg"
set contents of myAnnotation to "Might have been here"
```
move

Description
Moves a page object.

AppleScript Syntax
move reference to location reference

AppleScript Parameters

<table>
<thead>
<tr>
<th>move</th>
<th>The page object to move. The first page in a document is page 1.</th>
</tr>
</thead>
<tbody>
<tr>
<td>to</td>
<td>The new location for the page.</td>
</tr>
</tbody>
</table>

Return Value
A reference to the page that is moved.

AppleScript Example
move page 3 to before page 1
open

Description
Opens a document or documents.

AppleScript Syntax
open { list of alias } [invisible boolean] [options string]

AppleScript Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td>The document or documents to open.</td>
</tr>
<tr>
<td>invisible</td>
<td>Whether the opened document should be hidden. Default is false.</td>
</tr>
<tr>
<td>options</td>
<td>Optional parameter string of open actions.</td>
</tr>
</tbody>
</table>

Return Value
None

Related Events
close
**quit**

**Description**
Causes the Acrobat application to quit.

**AppleScript Syntax**
```applescript
quint [saving constant ]
```

**AppleScript Parameters**

| saving  | Determines whether to save documents that have been modified before quitting. Possible values:  
|---------|-----------------------------------------------------------------------------------------------|
| yes     | — Save the document.  
| no      | — Do not save the document.  
| ask     | — If the documents have been modified, ask the user whether to save them.  
|         | The default value is **ask**. |

**Return Value**
None

**AppleScript Example**
```applescript
quit saving yes
```
save

Description
Saves a document. Specifying the to parameter is equivalent to doing a SaveAs... You can save a document in one of the supported formats with the using option.

AppleScript Syntax
save reference [to file specification] [using reference] [linearize boolean]

AppleScript Parameters

<table>
<thead>
<tr>
<th>save</th>
<th>The document to be saved.</th>
</tr>
</thead>
<tbody>
<tr>
<td>to</td>
<td>The file into which the document is to be saved.</td>
</tr>
<tr>
<td></td>
<td>NOTE: This parameter is optional in Acrobat 6.0 and higher.</td>
</tr>
<tr>
<td>linearize</td>
<td>Determines whether the document should be saved as a linearized file.</td>
</tr>
<tr>
<td>using</td>
<td>The conversion method used to save the document in the desired format. Supported conversions by name are EPS Conversion and Postscript Conversion. All others can be specified by index using the conversion object.</td>
</tr>
</tbody>
</table>

Return Value
None

AppleScript Example
save document 1 to file "MyHardDrive:tempBig.ps" using PostScript Conversion with embedded fonts, images, preview, and annotation without binary given postScript level: 1
**set**

**Description**
Sets an object’s data or properties.

**AppleScript Syntax**
```
set reference to anything
```

**AppleScript Parameters**

<table>
<thead>
<tr>
<th>set</th>
<th>The object or property whose value is set.</th>
</tr>
</thead>
<tbody>
<tr>
<td>to</td>
<td>The new value.</td>
</tr>
</tbody>
</table>

**Return Value**
None

**Related Events**
```
get
```

**AppleScript Example**
```
set the name of first bookmark to "Chapter 1"
```
**Acrobat application Suite**

There are a number of Acrobat API calls for the Apple Event interface that are specific to Acrobat applications. This section details those calls.

---

### bring to front

**Description**

Brings the specified document’s window to the front.

**AppleScript Syntax**

```
bring to front reference
```

**AppleScript Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bring to front</td>
<td>The document to be displayed as the &quot;active document&quot; in the front window.</td>
</tr>
</tbody>
</table>

**Return Value**

None

**AppleScript Example**

```
bring to front document "AppleEvt.pdf"
```

**Apple Event ID**

```
kAEBringToFront ("bfrt")
```
clear selection

Description
Clears the document’s current selection, if any.

AppleScript Syntax
```
clear selection
```

AppleScript Parameters

<table>
<thead>
<tr>
<th>clear selection</th>
<th>The document containing the selection to be cleared</th>
</tr>
</thead>
</table>

Return Value
None

Related Events

select text

AppleScript Example
```
clear selection document "PLUGINS.PDF"
```

Apple Event ID
```
kAEClearSelection (‘cls1’)
```
close all docs

Description
Closes all documents.

AppleScript Syntax
```
close all docs [saving constant]
```

AppleScript Parameters

<table>
<thead>
<tr>
<th>saving</th>
<th>Determines whether to save modified documents before closing. Possible values:</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>— Save the document.</td>
</tr>
<tr>
<td>no</td>
<td>— Do not save the document.</td>
</tr>
<tr>
<td>ask</td>
<td>— If the document has been modified, ask the user whether to save it.</td>
</tr>
<tr>
<td></td>
<td>The default value is ask.</td>
</tr>
</tbody>
</table>

Return Value
None

Related Events
- open (required suite)
- open (core suite)

AppleScript Example
```
close all docs
```

Apple Event ID
```
kAECloseAllDocs (‘cldc‘)
```
create thumbs

Description
Creates thumbnail images for all pages in the document.

AppleScript Syntax
create thumbs reference

AppleScript Parameters

| create thumbs | The document in which thumbnails are created. |

Return Value
None

Related Events
delete thumbs

AppleScript Example
create thumbs document "roadmap.pdf"

Apple Event ID
kABCreateThumbs (‘crtb’)
delete pages

Description
Deletes the specified pages in the document.

AppleScript Syntax
delete pages reference first integer last integer

AppleScript Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>delete pages</td>
<td>The document containing the pages to be deleted.</td>
</tr>
<tr>
<td>first</td>
<td>The first page to be deleted.</td>
</tr>
<tr>
<td>last</td>
<td>The last page to be deleted.</td>
</tr>
</tbody>
</table>

Return Value
None

Related Events
insert pages
replace pages

AppleScript Example
delete pages document "AppleEvt.pdf" first 1 last 3

Apple Event ID
kAEDeletePages (‘dlpg’)

Apple Event Parameters
keyAEFirstPage (‘frpg’)
keyAELastPage (‘lapg’)
delete thumbs

Description
Deletes all thumbnails from the document.

AppleScript Syntax
```
dele tes thumbs reference
```

AppleScript Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>delete thumbs</td>
<td>The document from which thumbnails are deleted.</td>
</tr>
</tbody>
</table>

Return Value
None

Related Events
create thumbs

AppleScript Example
```
dele tes thumbs document "AppleEvt.pdf"
```

Apple Event ID
```
KAEDeleteThumbs (‘dltb’)
```
execute

Description
Executes the specified menu item.

AppleScript Syntax
execute reference

AppleScript Parameters

| execute | The menu item to execute. See the "Lists" section in the Acrobat And PDF Library API Reference for a list of menu item names. |

Return Value
None

AppleScript Example
activate
eexecute menu item "Open"

Apple Event ID
kAEExecute ('exec')
**find next note**

**Description**
Finds and selects the next text note in a document.

**AppleScript Syntax**
```
find next note reference [wrap around boolean]
```

**AppleScript Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>find next note</td>
<td>The document in which to find the next text note.</td>
</tr>
<tr>
<td>wrap around</td>
<td>Determines whether to continue the search at the beginning of a document if a note has not been found after the end of the document is reached. If true, the search wraps around; otherwise it does not. The default value is false.</td>
</tr>
</tbody>
</table>

**Return Value**
The text annotation found.

**Related Events**
- **find text**

**AppleScript Example**
```
find next note document "dev_acro.pdf"
```

**Apple Event ID**
```
kAEFindNextNote ('fnnt')
```

**Apple Event Parameters**
```
keyAEWrapAround ('wrar')
```
find text

Description
Finds text in a document.

AppleScript Syntax
```
find text reference string international text [case sensitive boolean] [whole words boolean] [wrap around boolean]
```

AppleScript Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>find text</td>
<td>The document to be searched.</td>
</tr>
<tr>
<td>string</td>
<td>The string to be found.</td>
</tr>
<tr>
<td>case sensitive</td>
<td>Determines whether searching is case-sensitive. The default value is false.</td>
</tr>
<tr>
<td>whole words</td>
<td>Determines whether to search only for whole words. The default value is false.</td>
</tr>
<tr>
<td>wrap around</td>
<td>Determines whether to continue the search at the beginning of a document if the specified text has not been found after the end of the document is reached. If true, the search wraps around; otherwise it does not. The default value is false.</td>
</tr>
</tbody>
</table>

Return Value
None

Related Events
find next note

AppleScript Example
```
find text document "PLUGINS.PDF" string "Develop" whole words true
```

Apple Event ID
```
kAEFindText ('ftxt')
```

Apple Event Parameters
```
keyAESearchString ('sstr')
keyAECaseSensitive ('case')
keyAEWholeWordsOnly ('whwd')
keyAEWrapAround ('wrar')
```
get info

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

AppleScript Syntax
get info reference key international text

AppleScript Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>get info</td>
<td>The document from which to obtain the Info dictionary entry.</td>
</tr>
<tr>
<td>key</td>
<td>The case-sensitive Info dictionary key whose value is to be obtained. The predefined keys are: Creator, Producer, CreationDate, Author, Title, Subject, and Keywords. None of these is required in the PDF file.</td>
</tr>
</tbody>
</table>

Return Value
A string containing the specified key's value, or an empty string if the key is not found.

AppleScript Example
get info document "PLUGINS.PDF" key "CreationDate"

Apple Event ID
kAEGetInfo (‘gnfo’)

Apple Event Parameters
keyAEInfoKey (‘inky’)

**go backward**

**Description**
Goes to the previous view in the stored view history. Does nothing if the current view is the first view in the history.

**AppleScript Syntax**
go backward reference

**AppleScript Parameters**

| go backward | A PDF Window object |

**Return Value**
None

**Related Events**
go forward
goto
goto next
goto previous

**AppleScript Example**
go backward first PDF Window

**Apple Event ID**
kAEGoBack ('gbck')
go forward

Description
Goes to the next view in the stored view history. Does nothing if the current view is the last view in the history.

AppleScript Syntax
```
go forward reference
```

AppleScript Parameters

| go forward | A PDF Window object |

Return Value
None

Related Events
```
go backward
goto
goto next
goto previous
```

AppleScript Example
```
go forward first PDF Window
```

Apple Event ID
```
kAEGoForward (‘gfwd’)```
### goto

#### Description
Displays the page that has the specified page number.

#### AppleScript Syntax

```plaintext
goto reference page integer
```

#### AppleScript Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>goto</td>
<td>The PDF Window object in which to change the page.</td>
</tr>
<tr>
<td>page</td>
<td>The page number of the page to be displayed. The first page in a document is page 1.</td>
</tr>
</tbody>
</table>

#### Return Value
None

#### Related Events
- go backward
- go forward
- goto next
- goto previous

#### AppleScript Example

```plaintext
goto first PDF Window page 2
```

#### Apple Event ID

```plaintext
kABGotoPage {'gtpg'}
```

#### Apple Event Parameters

```plaintext
keyAEPageNumber {'pg #'}
```
### goto next

**Description**

Displays the next page after the one currently displayed in the PDF Window. Does nothing if the current page is the last page in the document.

**AppleScript Syntax**

```applescript
goto next reference
```

**AppleScript Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>goto next</td>
<td>The PDF Window object in which to change the page.</td>
</tr>
</tbody>
</table>

**Return Value**

None

**Related Events**

- go backward
- go forward
- goto
- goto previous

**AppleScript Example**

```applescript
goto next first PDF Window
```

**Apple Event ID**

```applescript
kABGotoNextPage (‘nxpg’)
```
goto previous

Description
Displays the previous page before the one currently displayed in the PDF Window. Does nothing if the current page is the first page in the document.

AppleScript Syntax

goto previous reference

AppleScript Parameters

| goto previous | The PDF Window object in which to change the page. |

Return Value
None

Related Events

go backward
go forward
goto
goto next

AppleScript Example

goto previous first PDF Window

Apple Event ID

kABGotoPrevPage (‘pvpg’)
**insert pages**

**Description**

Inserts one or more pages from one document into another.

**AppleScript Syntax**

```
insert pages reference after integer from reference starting with integer
number of pages integer [insert bookmarks boolean]
```

**AppleScript Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>insert pages</td>
<td>The target document in which to insert the page or pages.</td>
</tr>
<tr>
<td>after</td>
<td>The number of the page after which the pages will be inserted. The first page in a document is page 1.</td>
</tr>
<tr>
<td>from</td>
<td>The source document containing the page or pages to be inserted.</td>
</tr>
<tr>
<td>starting with</td>
<td>The first page to be inserted.</td>
</tr>
<tr>
<td>number of pages</td>
<td>The number of pages to be inserted.</td>
</tr>
<tr>
<td>insert bookmarks</td>
<td>Determines whether to copy bookmarks that point to the inserted pages. Default is true.</td>
</tr>
</tbody>
</table>

**Return Value**

None

**Related Events**

**delete pages**

**AppleScript Example**

```
insert pages document "AppleEvt.pdf" after 2 from document
"dev_acro.pdf" starting with 1 number of pages 4
```

**Apple Event ID**

```
KAEInsertPages ('inpg')
```

**Apple Event Parameters**

- `keyAEInsertAfter ('inaf')`
- `keyAESourceDoc ('srdc')`
- `keyAESourceStartPage ('stpg')`
- `keyAENumPages ('nmpg')`
- `keyAEInsertBookmarks ('inbm')`
is toolbutton enabled

**Description**
Determine whether the specified toolbar button is enabled.

**AppleScript Syntax**
```
is toolbutton enabled named international text
```

**AppleScript Parameters**

| named | Button name. See the Acrobat And PDF Library API Reference for a list of toolbar button names. |

**Return Value**
true if the toolbar button is enabled, false otherwise.

**Related Events**
remove toolbutton

**AppleScript Example**
```
is toolbutton enabled named "AcroSrch:Query"
```

**Apple Event ID**
```
kAEIsToolButtonEnabled ('tben')
```

**Apple Event Parameters**

| keyAEButtonname ('tnm') |
maximize

Description
Sets the document’s window size to either its maximum or original size.

AppleScript Syntax
maximize reference max size boolean

AppleScript Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maximize</td>
<td>The document whose window is to be resized.</td>
</tr>
<tr>
<td>max size</td>
<td>If true, the document’s window is set to full size. If false, the window is returned to its original size.</td>
</tr>
</tbody>
</table>

Return Value
None

AppleScript Example
maximize document "AppleEvt.pdf" max size false

Apple Event ID
kAEMaximize (‘maxi’)

Apple Event Parameters
keyAEMaxSize (‘mxsz’)

perform

Description
Executes a bookmark's or link annotation's action.

AppleScript Syntax
perform reference

AppleScript Parameters

| object | The bookmark or page object whose action is to be performed. |

Return Value
None

AppleScript Example
perform last bookmark

Apple Event ID
kAEPperform (‘prfm’)
print pages

Description
Prints one or more pages from a document without displaying a modal Print dialog box.

AppleScript Syntax
print pages reference [first integer] [last integer] [PS Level integer] [binary output boolean] [shrink to fit boolean]

AppleScript Parameters

<table>
<thead>
<tr>
<th>print pages</th>
<th>The document containing the page or pages to be printed. This keyword and the actual filename must be specified.</th>
</tr>
</thead>
<tbody>
<tr>
<td>first</td>
<td>The first page to be printed. The default value is 1.</td>
</tr>
<tr>
<td>last</td>
<td>The last page to print. The default value is the number of the last page in the document.</td>
</tr>
<tr>
<td>PS Level</td>
<td>The PostScript language level (1 or 2) to use when printing to a PostScript printer. The default value is 1.</td>
</tr>
<tr>
<td>binary output</td>
<td>Determines whether binary output is permitted (used for PostScript printing only). The default value is false.</td>
</tr>
<tr>
<td>shrink to fit</td>
<td>Determines whether pages should be shrunk to fit paper in printer. The default value is false.</td>
</tr>
</tbody>
</table>

Return Value
None

AppleScript Example
print pages document "AppleEvt.pdf" first 1 last 3 PS Level 2 binary output true shrink to fit true

Apple Event ID
kAEPrintPages (‘prpg’)

Apple Event Parameters
keyAEFirstPage (‘frpg’)
keyAELastPage (‘lapg’)
keyAEPSLevel (’pslv’)
keyAEBinaryOK (‘binO’)
keyAEShrinkToFit (‘s2ft’)

Description
Prints one or more pages from a document without displaying a modal Print dialog box.
### read page down

**Description**

Scrolls forward through the document by one screen.

**AppleScript Syntax**

```apple
read page down reference
```

**AppleScript Parameters**

| read page down | The PDF Window object to be scrolled. |

**Return Value**

None

**Related Events**

- `read page up`
- `scroll`

**AppleScript Example**

```apple
read page down first PDF Window
```

**Apple Event ID**

```swift
kAEReadPageDown ('pgdn')
```
read page up

Description
Scrolls backward through the document by one screen.

AppleScript Syntax
read page up reference

AppleScript Parameters

| read page up       | The PDF Window object to be scrolled. |

Return Value
None

Related Events
read page down
scroll

AppleScript Example
read page up first PDFPageWindow

Apple Event ID
kAEReadPageUp ('pgup')
remove toolbutton

Description

Removes the specified button from the toolbar.

AppleScript Syntax

remove toolbutton named international text

AppleScript Parameters

| named | The name of the toolbar button to be removed. See the Acrobat And PDF Library API Reference for a list of toolbar button names. |

Return Value

None

Related Events

is toolbar enabled

AppleScript Example

remove toolbutton named "ZoomIn"

Apple Event ID

kAERemoveToolButton (‘rmtb’)

Apple Event Parameters

keyAEButtonname (‘tbnm’)
replace pages

Description

Replaces one or more pages in a document with pages from another document.

AppleScript Syntax

```
replace pages reference over integer from reference starting with integer number of pages integer [merge notes boolean]
```

AppleScript Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>replace pages</td>
<td>The target document whose pages are to be replaced.</td>
</tr>
<tr>
<td>over</td>
<td>The first page to be replaced. The first page in a document is page 1.</td>
</tr>
<tr>
<td>from</td>
<td>The source document from which the replacement page or pages are obtained.</td>
</tr>
<tr>
<td>starting with</td>
<td>The first page in the source document to be copied.</td>
</tr>
<tr>
<td>number of pages</td>
<td>The number of pages to be replaced.</td>
</tr>
<tr>
<td>merge notes</td>
<td>Determines whether to copy notes from the source document. The default value is true.</td>
</tr>
</tbody>
</table>

Return Value

None

Related Events

- delete pages
- insert pages

AppleScript Example

```
replace pages document "AppleEvt.pdf" over 2 from document "dev_acro.pdf" starting with 1 number of pages 4 merge notes false
```

Apple Event ID

```
kAEReplacePages ('rppg')
```

Apple Event Parameters

- keyAEDestStartPage ('dtpg')
- keyAESourceDoc ('srdc')
- keyAESourceStartPage ('stpg')
- keyAENumPages ('nmpg')
- keyAEMergeNotes ('mgnt')
scroll

Description

Scrolls the view of a page by the specified amount.

AppleScript Syntax

scroll reference X Amount integer Y Amount integer

AppleScript Parameters

<table>
<thead>
<tr>
<th>scroll</th>
<th>The PDF Window object in which to scroll the view.</th>
</tr>
</thead>
<tbody>
<tr>
<td>X Amount</td>
<td>The amount to scroll in the horizontal direction, in pixels. Positive values move the view to the right.</td>
</tr>
<tr>
<td>Y Amount</td>
<td>The amount to scroll in the vertical direction, in pixels. Positive values move the view down.</td>
</tr>
</tbody>
</table>

Return Value

None

Related Events

read page down
read page up

AppleScript Example

scroll first PDFWindow X Amount 20 Y Amount 100

Apple Event ID

kAEScroll (‘scrl’)  

Apple Event Parameters

keyAEXDelta (‘xdlt’)  
keyAEYDelta (‘ydlt’)


select text

Description
Selects text as specified by either character or word offsets.

AppleScript Syntax
```
select text reference [from words list of integer] [from chars list of integer]
```

AppleScript Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>select text</td>
<td>The PDF Window object in which to select text.</td>
</tr>
<tr>
<td>from words</td>
<td>The words to be selected. This consists of one or more pairs of word offsets from the beginning of the document and word lengths (the number of contiguous words).</td>
</tr>
<tr>
<td>from chars</td>
<td>Characters to be selected. This consists of one or more pairs of character offsets from the beginning of the document and character lengths (the number of contiguous characters).</td>
</tr>
</tbody>
</table>

Return Value
None

Related Events
```
clear selection
```

AppleScript Example
```
repeat with i from 1 to 10
    repeat with j from 1 to (10 - i)
        select text from words {i, j}
    end repeat
end repeat
```

Apple Event ID
```
kAESetTextSelection ('stxs')
```

Apple Event Parameters
```
keyAEWordList ('fmwd')
keyAEBCharList ('fmch')
```
**set info**

**Description**
Sets the value of a specified key in the document’s *Info* dictionary.

**AppleScript Syntax**
```
set info reference key international text value international text
```

**AppleScript Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>info</td>
<td>The <em>PDF Window</em> in which to set the value of an <em>Info</em> dictionary entry.</td>
</tr>
<tr>
<td>key</td>
<td>The <em>Info</em> dictionary key whose value is to be set.</td>
</tr>
<tr>
<td>value</td>
<td>The value to be stored.</td>
</tr>
</tbody>
</table>

**Return Value**
None

**AppleScript Example**
```
set info document "PlugIns.pdf" key "Author" value "Wolfgang Pauli"
```

**Apple Event ID**
```
kAESetInfo ('snfo')
```

**Apple Event Parameters**
```
keyAEInfoKey ('inky')
keyAEInfoValue ('invl')
```
**zoom**

**Description**
Changes the zoom level of the specified **PDF Window**.

**AppleScript Syntax**
```
zoom reference to small real
```

**AppleScript Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>zoom</code></td>
<td>The <strong>PDF Window</strong> object to be zoomed.</td>
</tr>
<tr>
<td><code>to</code></td>
<td>The zoom factor specified as a percentage. For example, a value of 100 (100%) displays the document with a magnification of 1.0.</td>
</tr>
</tbody>
</table>

**Return Value**
None

**AppleScript Example**
```
zoom first PDFWindow to 150
```

**Apple Event ID**
```
kAEZoomTo ('zmto')
```

**Apple Event Parameters**
```
keyAEZoomFactor ('zmft')
```
Miscellaneous Apple Events

Acrobat provides an Apple Event that does not fall into one of the regular suites.

**do script**

**Description**

Executes the specified Acrobat JavaScript script.

**AppleScript Syntax**

```
do script international text [file alias]
```

**AppleScript Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>do script</td>
<td>The Acrobat JavaScript script to be executed.</td>
</tr>
<tr>
<td>file</td>
<td>File holding the JavaScript script to be executed.</td>
</tr>
</tbody>
</table>

**Return Value**

Result of JavaScript execution as text.

**AppleScript Example**

```
do script MyJavaScriptFile.js
```
This reference contains the following sections:

- **OLE Automation Objects.** The Acrobat objects represented as OLE objects.
- **OLE Automation Methods.** Detailed description of each OLE method, including its parameters, return value, and related methods.
- **OLE Automation Properties.** A description of OLE Automation properties.

If you are using C and C++ you can find the header file you need in the IAC directory of the SDK. Visual Basic users do not need these header files.

The syntax used in this section follows that used in Microsoft Visual Basic 3.0.
This chapter details the objects found in the OLE Automation interface. Note that the names "AcroExch.App" and "AxAcroPDFLib.AxAcroPDF" signify the external strings OLE clients use to create objects of certain types. The Acrobat developer type libraries call them "CAcro.App" and "AcroPDFLib", respectively.

**AcroExch.App**

The Acrobat application itself. This is a creatable interface. From the application layer, you may control the appearance of Acrobat, whether Acrobat appears, and the size of the application window. This object provides access to the menu bar and the toolbar, as well as the visual representation of a PDF file on the screen (via an AVDoc object).

**AcroExch.AVDoc**

A view of a PDF document in a window. This is a creatable interface. There is one AVDoc object per displayed document. Unlike a PDDoc object, an AVDoc object has a window associated with it.

**AcroExch.AVPageView**

The area of the Acrobat application's window that displays the contents of a document's page. This is a non-creatable interface. Every AVDoc object has an AVPageView object and vice versa. The object provides access to the PDDoc and PDPage objects for the document being displayed.

**AcroExch.Hilite**

A highlighted region of text in a PDF document. This is a creatable interface. This object has a single method and is used by the PDPage object to create PDTextSelect objects.

**AcroExch.PDAnnot**

An annotation on a page in a PDF file. This is a non-creatable interface. Acrobat applications have two built-in annotation types: PDTextAnnot and PDLinkAnnot. The object provides access to the physical attributes of the annotation. Plug-ins may add movie and
OLE Automation Objects

Widget (form field) annotations, and developers can define new annotation subtypes by creating new annotation handlers.

**AcroExch.PDBookmark**

A bookmark for a page in a PDF file. This is a creatable interface. Each bookmark has a title that appears on screen, and an action that specifies what happens when a user clicks on the bookmark. Bookmarks can either be created interactively by the user through the Acrobat application's user interface or programmatically generated. The typical action for a user-created bookmark is to move to another location in the current document, although any action can be specified.

**AcroExch.PDDoc**

The underlying PDF representation of a document. This is a creatable interface. There is a correspondence between a PDDoc object and an ASFile object (an opaque representation of an open file made available through an interface encapsulating Acrobat’s access to file services), and the PDDoc object is the hidden object behind every AVDoc object. An ASFile object may have zero or more underlying files, so a PDF file does not always correspond to a single disk file. For example, an ASFile object may provide access to PDF data in a database.

Through PDDoc objects, your application can perform most of the Document menu items from Acrobat (delete pages, replace pages, and so on), create and delete thumbnails, and set and retrieve document information fields.

**AcroExch.PDPage**

A single page in the PDF representation of a document. This is a non-creatable interface. Just as PDF files are partially composed of their pages, PDDoc objects are composed of PDPage objects. A page contains a series of objects representing the objects drawn on the page (PDGraphic objects), a list of resources used in drawing the page, annotations (PDAnnot objects), an optional thumbnail image of the page, and the beads used in any articles that occur on the page. The first page in a PDDoc object is page 0.

**AcroExch.PDTextSelect**

A selection of text on a single page that may contain more than one disjointed group of words. This is a non-creatable interface. A text selection is specified by one or more ranges of text, with each range containing the word numbers of the selected words. Each range specifies a start and end word, where "start" is the first of a series of selected words and "end" is the next word after the last in the selection.
AxAcroPDFLib.AxAcroPDF

An object containing a set of methods that provide access to PDF browser controls. This is a creatable interface. This object makes it possible to load a file, move to various pages within the file, and specify various display and print options.
AcroExch.App

All methods in this section belong to the AcroExch.App class.

CloseAllDocs

VARIANT_BOOL CloseAllDocs();

Description

Closes all open documents. You can close each individual AVDoc object by calling AVDoc.Close.

NOTE: You must explicitly close all documents or call App.CloseAllDocs. Otherwise, the process will never exit.

Parameters

None

Return Value

-1 if successful, 0 if not.

Related Methods

AVDoc.Close
AVDoc.Open
AVDoc.OpenInWindow
AVDoc.OpenInWindowEx
PDDoc.Close
PDDoc.Open
PDDoc.OpenAVDoc
Exit

VARIANT_BOOL Exit();

Description
Exits Acrobat. Applications should call App.Exit before exiting.

NOTE: Use App.CloseAllDocs to close all the documents before calling this method.

Parameters
None

Return Value
Returns -1 if the entire shutdown process succeeded. This includes closing any open documents, releasing OLE references, and finally exiting the application. If any step fails, the function returns 0, and the application will continue running. This method will not work if the application is visible (if the user is in control of the application). In such cases, if the Show() method had previously been called, you may call Hide() and then Exit().

Related Methods
App.CloseAllDocs
**GetActiveDoc**

```cpp
LPDISPATCH GetActiveDoc();
```

**Description**

Gets the frontmost document.

**Parameters**

None

**Return Value**

The LPDISPATCH for the frontmost `AcroExch.AVDoc` object. If there are no documents open, it will return `NULL`.

**Related Methods**

`App.GetAVDoc`
GetActiveTool

BSTR GetActiveTool();

Description

Gets the name of the currently active tool.

Parameters

None

Return Value

Returns NULL if there is no active tool. Returns the name of the currently active tool otherwise. See the Acrobat And PDF Library API Reference for a list of tool names.

Related Methods

App.SetActiveTool
GetAVDoc

LPDISPATCH GetAVDoc(long nIndex);

Description

Gets an AcroExch.AVDoc object via its index within the list of open AVDoc objects. Use App.GetNumAVDocs to determine the number of AcroExch.AVDoc objects.

Parameters

| nIndex | The index of the document to get. |

Return Value

The LPDISPATCH for the specified AcroExch.AVDoc document, or NULL if nIndex is greater than the number of open documents.

Related Methods

App.GetActiveTool
GetFrame

LPDISPATCH GetFrame();

Description

Gets the window's frame.

**NOTE:** GetFrame is not useful when the PDF file was opened with **AVDoc.OpenInWindow**. GetFrame returns the application window's frame (not the document window's frame). However, the application's window is hidden when a document is opened using **OpenInWindow**, and does not change in size as document windows are moved and resized.

Parameters

None

Return Value

The LPDISPATCH for the window's frame, specified as an **AcroExch.Rect**.

Related Methods

- **App.Maximize**
- **App.SetFrame**
GetInterface

LPDISPATCH GetInterface (BSTR szName);

Description

Gets an IDispatch interface for a named object, typically a third-party plug-in. This is an entry point to functionality that is undefined at the time of this writing, and which must be provided by the plug-in author. If you are accessing third-party functionality through GetInterface, ask the author for additional information.

Parameters

<table>
<thead>
<tr>
<th>szName</th>
<th>Name of the object.</th>
</tr>
</thead>
</table>

Return Value

The LPDISPATCH for the object's interface or NULL if the object was not found.

Related Methods

None.
GetLanguage

BSTR GetLanguage();

Description

Gets a code that specifies which language the Acrobat application's user interface is using.

Parameters

None

Return Value

String containing a three-letter language code. Must be one of the following:

- DEU – German
- ENU – English
- ESP – Spanish
- FRA – French
- ITA – Italian
- NLD – Dutch
- SVE – Swedish

Related Methods

App.GetPreference
App.SetPreference
GetNumAVDocs

long GetNumAVDocs();

**Description**

Gets the number of open `AcroExch.AVDoc` objects. The maximum number of documents the Acrobat application can open at a time is specified by the `avpMaxOpenDocuments` preference, which can be obtained with `App.GetPreferenceEx` and set by `App.SetPreferenceEx`.

**Parameters**

None

**Return Value**

The number of open `AcroExch.AVDoc` objects.

**Related Methods**

- `App.GetActiveDoc`
- `App.GetAVDoc`
GetPreference

IMPORTANT: This method has been deprecated; use GetPreferenceEx instead. GetPreference is unable to accept important data types such as strings, but GetPreferenceEx can convert many data types into acceptable formats.

long GetPreference(short nType);

Description

Gets a value from the preferences file. Zoom values (used in avpDefaultZoomScale and avpMaxPageCacheZoom) are returned as percentages (for example, 1.00 is returned as 100). Colors (used in avpNoteColor -- PDcolorValue) are automatically converted to RGB values from the representation used in the preferences file.

Parameters

| nType | The preferences item whose value is set. See the Acrobat And PDF Library API Reference for a list of preference items. |

Return Value

The value of the specified preference item.

Related Methods

App.GetLanguage
App.SetPreference
GetPreferenceEx

VARIANT GetPreferenceEx(short nType);

Description

Gets the specified application preference, using the VARIANT type to pass values.

Parameters

| nType     | The name of the preferences item whose value is obtained. |

Return Value

The value of the specified preference item.

Related Methods

App.GetLanguage
App.SetPreferenceEx
Hide

VARIANT_BOOL Hide();

Description
Hides the Acrobat application. When the viewer is hidden, the user has no control over it, and the Acrobat application exits when the last automation object is closed.

Parameters
None

Return Value
-1 if successful, 0 if not.

Related Methods
App.Show
Lock

VARIANT_BOOL Lock(BSTR szLockedBy);

Description

Locks the Acrobat application. Typically, this method is called when using AVDoc.OpenInWindowEx to draw into another application's window. If you call App.Lock, you should call App.UnlockEx when you are done using OLE automation.

There are some advantages and disadvantages of locking the viewer when using AVDoc.OpenInWindowEx. You must weigh these before deciding whether to lock the viewer:

- Locking prevents problems that can sometimes occur if two processes are trying to open a file at the same time.
- Locking prevents a user from using Acrobat's user interface (such as adding annotations) in your application's window.
- Locking can prevent any other application, including the Acrobat application, from opening PDF files. This problem can be minimized by calling App.UnlockEx as soon as the file has been opened.

Parameters

| szLockedBy | A string that is used as the name of the application that has locked the Acrobat application. |

Return Value

-1 if the Acrobat application was locked successfully, 0 otherwise. Locking will fail if the Acrobat application is visible.

Related Methods

App.UnlockEx
Minimize

VARIANT_BOOL Minimize(long BMinimize);

Description
Minimizes the Acrobat application.

Parameters

| BMinimize | If a positive number, the Acrobat application is minimized. If 0, the Acrobat application is returned to its normal state. |

Return Value
-1 if successful, 0 if not.

Related Methods
App.GetFrame
App.SetFrame
Maximize

VARIANT_BOOL Maximize(long bMaximize);

Description
Maximizes the Acrobat application.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bMaximize</td>
<td>If a positive number, the Acrobat application is maximized. If 0, the Acrobat application is returned to its normal state.</td>
</tr>
</tbody>
</table>

Return Value
-1 if successful, 0 if not.

Related Methods
App.GetFrame
App.SetFrame
**MenuItemExecute**

```c
VARIANT_BOOL MenuItemExecute(BSTR szMenuItemName);
```

**Description**

Executes the menu item whose language-independent menu item name is specified.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>szMenuItemName</td>
<td>The language-independent name of the menu item to execute. See the Acrobat And PDF Library API Reference for a list of menu item names.</td>
</tr>
</tbody>
</table>

**Return Value**

Returns -1 if the menu item executes successfully, 0 if the menu item is missing or is not enabled.

**Related Methods**

- `App.MenuItemIsEnabled`
- `App.MenuItemIsMarked`
- `App.MenuItemRemove`
**MenuItemIsEnabled**

VARIANT_BOOL MenuItemIsEnabled(BSTR szMenuItemName);

**Description**

Determines whether the specified menu item is enabled.

**Parameters**

| szMenuItemName | The language-independent name of the menu item whose enabled state is obtained. See the Acrobat And PDF Library API Reference for a list of menu item names. |

**Return Value**

-1 if the menu item is enabled, 0 if it is disabled or does not exist.

**Related Methods**

- `App.MenuItemExecute`
- `App.MenuItemIsMarked`
- `App.MenuItemRemove`
**MenuItemIsMarked**

```c
VARIANT_BOOL MenuItemIsMarked(BSTR szMenuItemName);
```

**Description**

Determines whether the specified menu item is marked.

**Parameters**

| szMenuItemName | The language-independent name of the menu item whose marked state is obtained. See the Acrobat And PDF Library API Reference for a list of menu item names. |

**Return Value**

-1 if the menu item is marked, 0 if it is not marked or does not exist.

**Related Methods**

- App(MenuItemExecute)
- App(MenuItemIsEnabled)
- App(MenuItemRemove)
**MenuItemRemove**

VARIANT_BOOL MenuItemRemove(BSTR szMenuItemName);

**Description**

Removes the menu item whose language-independent menu item is specified.

**Parameters**

| szMenuItemName | The language-independent name of the menu item to remove. See the Acrobat And PDF Library API Reference for a list of menu item names. |

**Return Value**

-1 if the menu item was removed, 0 if the menu item does not exist.

**Related Methods**

App.MenuItemExecute
App.MenuItemIsEnabled
App.MenuItemIsMarked
**Restore**

VARIANT_BOOL Restore(long bRestore);

**Description**

Restores the main window of the Acrobat application. Calling this with `bRestore` set to a positive number causes the main window to be restored to its original size and position and become active.

**Parameters**

| bRestore | If a positive number, the Acrobat application is restored, 0 otherwise. |

**Return Value**

-1 if successful, 0 if not.

**Related Methods**

- `App.GetFrame`
- `App.SetFrame`
SetActiveTool

VARIANT_BOOL SetActiveTool(BSTR szButtonName, long bPersistent);

**Description**
Sets the active tool according to the specified name, and determines whether the tool is to be used only once or should remain active after being used (persistent).

**Parameters**

<table>
<thead>
<tr>
<th>szButtonName</th>
<th>The name of the tool to set as the active tool. See the Acrobat And PDF Library API Reference for a list of tool names.</th>
</tr>
</thead>
<tbody>
<tr>
<td>bPersistent</td>
<td>A request indicating whether the tool should be persistent. A positive number indicates a request to the Acrobat application for the tool to remain active after it has been used. If 0 is specified, the Acrobat application reverts to the previously active tool after this tool is used once.</td>
</tr>
</tbody>
</table>

**Return Value**
-1 if the tool was set, 0 otherwise.

**Related Methods**
- App.SetActiveTool
- App.ToolButtonIsEnabled
- App.ToolButtonRemove
SetFrame

VARIANT_BOOL SetFrame(LPDISPATCH iAcroRect);

Description
Sets the window's frame to the specified rectangle.

Parameters

| iAcroRect | The LPDISPATCH for an AcroExch.Rect specifying the window frame. iAcroRect contains the instance variable m_lpDispatch, which contains the LPDISPATCH. |

Return Value
-1 if the frame was set, 0 if iAcroRect is not of type AcroExch.Rect.

Related Methods
App.GetFrame
App.Maximize
**SetPreference**

**IMPORTANT:** This method has been deprecated; use `SetPreferenceEx` instead. `SetPreference` is unable to accept important data types such as strings, but `SetPreferenceEx` can convert many data types into acceptable formats.

```
VARIANT_BOOL SetPreference(short nType, long nValue);
```

**Description**

Sets a value in the preferences file. Zoom values (used in `avpDefaultZoomScale` and `avpMaxPageCacheZoom`) must be passed as percentages and are automatically converted to fixed point numbers (for example, 100 is automatically converted to 1.0). Colors (used in `avpHighlightColor` or `avpNoteColor`) are automatically converted from RGB values to the representation used in the preferences file.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nType</td>
<td>The preferences item whose value is set. See the Acrobat And PDF Library API Reference for a list of preference items.</td>
</tr>
<tr>
<td>nValue</td>
<td>The value to set.</td>
</tr>
</tbody>
</table>

**Return Value**

-1 if successful, 0 if not.

**Related Methods**

- `App.GetLanguage`
- `App.GetPreferenceEx`
SetPreferenceEx

VARIANT_BOOL SetPreferenceEx(short nType, VARIANT* pVal);

Description
Sets the application preference specified by nType to the value stored at pVal. If pVal has a non-conforming VARTYPE, SetPreferenceEx will perform type conversion. For example, a string representation of an integer will be successfully converted to an actual integer.

Parameters

<table>
<thead>
<tr>
<th>nType</th>
<th>The preferences item whose value is set. See the Acrobat And PDF Library API Reference for a list of preference items.</th>
</tr>
</thead>
<tbody>
<tr>
<td>pVal</td>
<td>The value to set.</td>
</tr>
</tbody>
</table>

Return Value
Returns -1 if nType is a supported type or the type conversion is successful, 0 otherwise.

Related Methods
App.GetLanguage
App.GetPreferenceEx
Show

VARIANT_BOOL Show();

Description
Shows the Acrobat application. When the viewer is shown, the user is in control, and the Acrobat application does not automatically exit when the last automation object is destroyed. (However, it will exit if there are no documents being displayed.)

Parameters
None

Return Value
-1 if successful, 0 if not.

Related Methods
App.Hide
ToolButtonIsEnabled

VARIANT_BOOL ToolButtonIsEnabled(BSTR szButtonName);

Description

Determines whether the specified toolbar button is enabled.

Parameters

<table>
<thead>
<tr>
<th>szButtonName</th>
</tr>
</thead>
<tbody>
<tr>
<td>The name of the button whose enabled state is checked. See the Acrobat And PDF Library API Reference for a list of toolbar button names.</td>
</tr>
</tbody>
</table>

Return Value

-1 if the button is enabled, 0 if it is not enabled or does not exist.

Related Methods

App.SetActiveTool
App.SetActiveTool
App.ToolButtonRemove
**ToolButtonRemove**

```
VARIANT_BOOL ToolButtonRemove(BSTR szButtonName);
```

**Description**

Removes the specified button from the toolbar.

**Parameters**

| szButtonName | The name of the button to remove. See the Acrobat And PDF Library API Reference for a list of toolbar button names. |

**Return Value**

-1 if the button was removed, 0 otherwise.

**Related Methods**

- `App.SetActiveTool`
- `App.SetActiveTool`
- `App.ToolButtonIsEnabled`
Unlock

VARIANT_BOOL Unlock();

Description
Unlocks the Acrobat application if it was previously locked. This method clears a flag that indicates the viewer is locked. If you called App.Lock, you should call App.Unlock when you are done using OLE automation.

**NOTE:** Use App.Lock and App.UnlockEx if you call OpenInWindow.

**NOTE:** In version 4.0 or later, use App.UnlockEx instead.

Typically, you call App.Lock when your application initializes and App.Unlock in your application's destructor method.

Parameters

None

Return Value

-1 if successful, 0 if not.

Related Methods

App.Lock
App.UnlockEx
UnlockEx

VARIANT_BOOL UnlockEx (BSTR szLockedBy);

Description
Unlocks the Acrobat application if it was previously locked.

**NOTE:** It is strongly recommended that you use this method in version 4.0 or later.

Parameters

| szLockedBy    | A string indicating the name of the application to be unlocked. |

Return Value
-1 if successful, 0 if not.

Related Methods
App.Lock

Parameters
**AcroExch.AVDoc**

All methods in this section belong to the **AcroExch.AVDoc** class.

---

**BringToFront**

```cpp
VARIANT_BOOL BringToFront();
```

**Description**

Brings the window to the front.

**Parameters**

None

**Return Value**

Returns 0 if no document is open, -1 otherwise.

**Related Methods**

None
ClearSelection

VARIANT_BOOL ClearSelection();

Description
Clears the current selection.

Parameters
None

Return Value
Returns -1 if the selection was cleared, 0 if no document is open or the selection could not be cleared.

Related Methods
AVDocSetTextSelection
AVDocShowTextSelect
PDDocCreateTextSelect
PDPagedCreatePageHilite
PDPagedCreateWordHilite
PDTextSelectDestroy
PDTextSelectGetBoundingRect
PDTextSelectGetNumText
PDTextSelectGetPage
PDTextSelectGetText
Close

VARIANT_BOOL Close(long bNoSave);

Description
Closes a document. You can close all open AVDoc objects by calling App.CloseAllDocs.
To reuse an AVDoc object, close it with AVDoc.Close, then use the AVDoc object’s LPDISPATCH for AVDoc.OpenInWindow.

Parameters

| bNoSave | If a positive number, the document is closed without saving it. If 0 and the document has been modified, the user is asked whether or not the file should be saved. |

Return Value
Always returns -1, even if no document is open.

Related Methods
- App.CloseAllDocs
- AVDoc.Open
- AVDoc.OpenInWindow
- AVDoc.OpenInWindowEx
- PDDoc.Close
- PDDoc.Open
- PDDoc.OpenAVDoc
FindText

VARIANT_BOOL FindText(BSTR szText, long bCaseSensitive, long bWholeWordsOnly, long bReset);

Description
Finds the specified text, scrolls so that it is visible, and highlights it.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>szText</td>
<td>The text to be found.</td>
</tr>
<tr>
<td>bCaseSensitive</td>
<td>If a positive number, the search is case-sensitive. If 0, it is case-insensitive.</td>
</tr>
<tr>
<td>bWholeWordsOnly</td>
<td>If a positive number, the search matches only whole words. If 0, it matches partial words.</td>
</tr>
<tr>
<td>bReset</td>
<td>If a positive number, the search begins on the first page of the document. If 0, it begins on the current page.</td>
</tr>
</tbody>
</table>

Return Value
-1 if the text was found, 0 otherwise.

Related Methods
None
GetAVPageView

LPDISPATCH GetAVPageView();

Description
Gets the AcroExch.AVPageView associated with an AcroExch.AVDoc.

Parameters
None

Return Value
The LPDISPATCH for the AcroExch.AVPageView or NULL if no document is open.

Related Methods
AVDoc.GetPDDoc
AVDoc.SetViewMode
AVPageView.GetAVDoc
AVPageView.GetDoc
GetFrame

    LPDISPATCH GetFrame();

Description

Gets the rectangle specifying the window's size and location.

Parameters

None

Return Value

The LPDISPATCH for an AcroExch.Rect containing the frame, or NULL if no document is open.

Related Methods

AVDoc.SetFrame
GetPDDoc

LPDISPATCH GetPDDoc();

Description
Gets the `AcroExch.PDDoc` associated with an `AcroExch.AVDoc`.

Parameters
None

Return Value
The `LPDISPATCH` for the `AcroExch.PDDoc` or `NULL` if no document is open.

Related Methods
- `AVDoc.GetAVPageView`
- `AVPageView.GetAVDoc`
- `AVPageView.GetDoc`
**GetTitle**

BSTR GetTitle();

**Description**

Gets the window's title.

**Parameters**

None

**Return Value**

The window's title or **NULL** if no document is open.

**Related Methods**

AVDoc.Open
AVDoc.SetTitle
PDDoc.OpenAVDoc
**GetViewMode**

```c
long GetViewMode();
```

**Description**

Gets the current document view mode (pages only, pages and thumbnails, or pages and bookmarks).

**Parameters**

None

**Return Value**

The current document view mode or 0 if no document is open. The return value will be one of the values listed in View Mode (see document in Chapter 1, “Apple Event Objects” for an explanation of the View Mode property).

**Related Methods**

- `AVDoc.GetAVPageView`
- `AVDoc.SetViewMode`
IsValid

VARIANT_BOOL IsValid();

Description
Determines whether the AcroExch.AVDoc is still valid. This method only checks whether the document has been closed or deleted; it does not check the internal structure of the document.

Parameters
None

Return Value
-1 if the document can still be used, 0 otherwise.

Related Methods
App.GetAVDoc
AVPageView.GetAVDoc
Maximize

VARIANT_BOOL Maximize(long bMaxSize);

**Description**

Maximizes the window if **bMaxSize** is a positive number.

**Parameters**

| **bMaxSize** | Indicates whether window should be maximized. |

**Return Value**

-1 if a document is open, 0 otherwise.

**Related Methods**

- AVDoc.GetFrame
- AVDoc.SetFrame
**Open**

```c
VARIANT_BOOL Open(BSTR szFullPath, BSTR szTempTitle);
```

**Description**

Opens a file. A new instance of `AcroExch.AVDoc` must be created for each displayed PDF file.

**NOTE:** An application must explicitly close any `AVDoc` that it opens by calling `AVDoc.Close` (the destructor for the `AcroExch.AVDoc` class does not call `AVDoc.Close`).

**Parameters**

| szFullPath | The full pathname of the file to open. |
| szTempTitle | An optional title for the window in which the file is opened. If `szTempTitle` is NULL or the empty string, it is ignored. Otherwise, `szTempTitle` is used as the window title. |

**Return Value**

-1 if the file was opened successfully, 0 otherwise.

**Related Methods**

- `App.CloseAllDocs`
- `AVDoc.Close`
- `AVDoc.GetTitle`
- `AVDoc.OpenInWindow`
- `AVDoc.OpenInWindowEx`
- `AVDoc.SetTitle`
- `PDDoc.Close`
- `PDDoc.Open`
- `PDDoc.OpenAVDoc`
OpenInWindow

VARIANT_BOOL OpenInWindow(BSTR fileName, short hWnd);

Description

IMPORTANT: As of Acrobat 3.0, this method simply returns false. Use the method AVDoc.OpenInWindowEx instead.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>The full pathname of the file to open.</td>
</tr>
<tr>
<td>hWnd</td>
<td>Handle for the window in which the file is displayed.</td>
</tr>
</tbody>
</table>

Return Value

-1.

Related Methods

App.CloseAllDocs
AVDoc.Close
AVDoc.Open
AVDoc.OpenInWindowEx
PDDoc.Close
PDDoc.Open
PDDoc.OpenAVDoc
OpenInWindowEx

VARIANT_BOOL OpenInWindowEx(LPCTSTR szFullPath, long hWnd, long openFlags, long useOpenParams, long pgNum, short pageMode, short zoomType, long zoom, short top, short left);

Description

Opens a PDF file and displays it in a user-specified window. The default Windows file system is used to open the file.

**NOTE:** Acrobat uses only its built-in implementation of the file opening code—not any replacement file system version that a developer might have added with a plug-in.

An application must explicitly close any AVDoc that it opens by calling **AVDoc.Close** (the destructor for the **AcroExch.AVDoc** class does not call **AVDoc.Close**).

Do not set the view mode to **Close** with **AVDoc.SetViewMode** when using **AVDoc.OpenInWindowEx**; this will cause the viewer and application to hang.

If you use a view mode of **AV_PAGE_VIEW**, the **pagemode** parameter will be ignored.

See **AVApp.Lock** for a discussion of whether to lock the viewer before making this call.
# OLE Automation Methods

**AcroExch.AVDoc**

## Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>szFullPath</td>
<td>The full pathname of the file to open.</td>
</tr>
<tr>
<td>hWnd</td>
<td>Handle for the window in which the file is displayed.</td>
</tr>
<tr>
<td>openFlags</td>
<td>Type of window view. Must be one of the following:</td>
</tr>
<tr>
<td></td>
<td>- <strong>AV_EXTERNAL_VIEW</strong> — Display the <strong>AVPageView</strong>, scrollbars, toolbar, and bookmark or thumbnails pane. Annotations are active.</td>
</tr>
<tr>
<td></td>
<td>- <strong>AV_DOC_VIEW</strong> — Display the <strong>AVPageView</strong>, scrollbars, and bookmark or thumbnails pane. Annotations are active.</td>
</tr>
<tr>
<td></td>
<td>- <strong>AV_PAGE_VIEW</strong> — Display only the <strong>AVPageView</strong> (the window that displays the PDF file). Do not display scrollbars, the toolbar, and bookmark or thumbnails pane. Annotations are active.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: It is recommended that either <strong>AV_DOC_VIEW</strong> or <strong>AV_PAGE_VIEW</strong> be used. Use <strong>AV_EXTERNAL_VIEW</strong> only if you do not want the application to display its own toolbar. Use <strong>AV_PAGE_VIEW</strong> to open the file with no scrollbars and no status window at the bottom of the page.</td>
</tr>
<tr>
<td>useOpenParams</td>
<td>0 indicates that the open action of the file is used; a positive number indicates that the action is overridden with the parameters that follow.</td>
</tr>
<tr>
<td>pgNum</td>
<td>Page number at which the file is to be opened if <strong>useOpenParams</strong> is a positive number. The first page is zero.</td>
</tr>
<tr>
<td>pageMode</td>
<td>Specifies page view mode if <strong>useOpenParams</strong> is a positive number. See View Mode (in document) for a list of possible views.</td>
</tr>
<tr>
<td>zoomType</td>
<td>Zoom type of the page view if <strong>useOpenParams</strong> is a positive number. See zoom type (in document) for a list of possible zoom types.</td>
</tr>
<tr>
<td>zoom</td>
<td>Zoom factor, used only for <strong>AVZoomNoVary</strong> if <strong>useOpenParams</strong> is a positive number.</td>
</tr>
<tr>
<td>top</td>
<td>Used for certain zoom types (such as <strong>AVZoomNoVary</strong>) if <strong>useOpenParams</strong> is a positive number. See the <em>PDF Reference</em> for information on views.</td>
</tr>
<tr>
<td>left</td>
<td>Used for certain zoom types (such as <strong>AVZoomNoVary</strong>) if <strong>useOpenParams</strong> is a positive number. See the <em>PDF Reference</em> for information on views.</td>
</tr>
</tbody>
</table>
Return Value

-1 if the document was opened successfully, 0 otherwise.

Related Methods

App.CloseAllDocs
AVDoc.Close
AVDoc.Open
AVDoc.OpenInWindow
PDDoc.Close
PDDoc.Open
PDDoc.OpenAVDoc
**PrintPages**

```c
VARIANT_BOOL PrintPages(long nFirstPage,
long nLastPage,long nPSLevel,
long bBinaryOk, long bShrinkToFit);
```

**Description**

Prints a specified range of pages displaying a print dialog box. **PrintPages** always uses the default printer setting.

**Parameters**

- **nFirstPage**
  The first page to be printed. The first page in a **PDDoc** object is page 0.

- **nLastPage**
  The last page to be printed.

- **nPSLevel**
  Valid values are 2 and 3. If 2, PostScript Level 2 operators are used. If 3, PostScript Language Level 3 operators are also used.

- **bBinaryOk**
  If a positive number, binary data may be included in the PostScript program. If 0, all data is encoded as 7-bit ASCII.

- **bShrinkToFit**
  If a positive number, the page is shrunk (if necessary) to fit within the imageable area of the printed page. If 0, it is not.

**Return Value**

0 if there were any exceptions while printing or if no document was open, -1 otherwise.

**Related Methods**

- **AVDoc.PrintPagesEx**
- **AVDoc.PrintPagesSilent**
- **AVDoc.PrintPagesSilentEx**
PrintPagesEx

VARIANT_BOOL PrintPagesEx(long nFirstPage, long nLastPage,
    long nPSLevel, long bBinaryOk,
    long bShrinkToFit, long bReverse,
    long bFarEastFontOpt, long bEmitHalftones,
    long iPageOption);

Description
Prints a specified range of pages, displaying a print dialog box. PrintPagesEx always uses the default printer setting.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nFirstPage</td>
<td>The first page to be printed. The first page in a PDDoc object is page 0.</td>
</tr>
<tr>
<td>nLastPage</td>
<td>The last page to be printed.</td>
</tr>
<tr>
<td>nPSLevel</td>
<td>If 2, PostScript Level 2 operators are used. If 3, PostScript Language Level 3 operators are also used.</td>
</tr>
<tr>
<td>bBinaryOk</td>
<td>If a positive number, binary data may be included in the PostScript program. If 0, all data is encoded as 7-bit ASCII.</td>
</tr>
<tr>
<td>bShrinkToFit</td>
<td>If a positive number, the page is shrunk (if necessary) to fit within the imageable area of the printed page. If 0, it is not.</td>
</tr>
<tr>
<td>bReverse</td>
<td>(PostScript printing only) If a positive number, print the pages in reverse order. If false, print the pages in the regular order.</td>
</tr>
<tr>
<td>bFarEastFontOpt</td>
<td>(PostScript printing only) Set to a positive number if the destination printer has multibyte fonts; set to 0 otherwise.</td>
</tr>
<tr>
<td>bEmitHalftones</td>
<td>(PostScript printing only) If a positive number, emit the halftones specified in the document. If 0, do not.</td>
</tr>
<tr>
<td>iPageOption</td>
<td>Pages in the range to print. Must be one of: PDAllPages, PDEvenPagesOnly, or PDOddPagesOnly.</td>
</tr>
</tbody>
</table>

Return Value

0 if there were any exceptions while printing or if no document was open, -1 otherwise.
Related Methods

AVDoc.PrintPages
AVDoc.PrintPagesSilent
AVDoc..PrintPagesSilentEx
PrintPagesSilent

VARIANT_BOOL PrintPagesSilent(long nFirstPage, long nLastPage, long nPSLevel, long bBinaryOk, long bShrinkToFit);

Description

Prints a specified range of pages without displaying any dialog box. This method is identical to AVDoc.PrintPages except for not displaying the dialog box. PrintPagesSilent always uses the default printer setting.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nFirstPage</td>
<td>The first page to be printed. The first page in a PDDoc object is page 0.</td>
</tr>
<tr>
<td>nLastPage</td>
<td>The last page to be printed.</td>
</tr>
<tr>
<td>nPSLevel</td>
<td>If 2, PostScript Level 2 operators are used. If 3, PostScript Language Level 3 operators are also used.</td>
</tr>
<tr>
<td>bBinaryOk</td>
<td>If a positive number, binary data may be included in the PostScript program. If 0, all data is encoded as 7-bit ASCII.</td>
</tr>
<tr>
<td>bShrinkToFit</td>
<td>If a positive number, the page is shrunk (if necessary) to fit within the imageable area of the printed page. If 0, it is not.</td>
</tr>
</tbody>
</table>

Return Value

0 if there were any exceptions while printing or if no document was open, -1 otherwise.

Related Methods

AVDoc.PrintPages
AVDoc.PrintPagesEx
AVDoc.PrintPagesSilentEx
PrintPagesSilentEx

VARIANT_BOOL PrintPagesSilentEx(long nFirstPage,
    long nLastPage,
    long nPSLevel, long bBinaryOk,
    long bShrinkToFit, long bReverse,
    long bFarEastFontOpt,
    long bEmitHalftones,
    long iPageOption);

Description
Prints a specified range of pages without displaying any dialog box. This method is identical to AVDoc.PrintPages except for not displaying the dialog box. PrintPagesSilent always uses the default printer setting.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nFirstPage</td>
<td>The first page to be printed.</td>
</tr>
<tr>
<td>nLastPage</td>
<td>The last page to be printed.</td>
</tr>
<tr>
<td>nPSLevel</td>
<td>If 2, PostScript Level 2 operators are used. If 3, PostScript Language Level 3 operators are also used.</td>
</tr>
<tr>
<td>bBinaryOk</td>
<td>If a positive number, binary data may be included in the PostScript program. If 0, all data is encoded as 7-bit ASCII.</td>
</tr>
<tr>
<td>bShrinkToFit</td>
<td>If a positive number, the page is shrunk (if necessary) to fit within the imageable area of the printed page. If 0, it is not.</td>
</tr>
<tr>
<td>bReverse</td>
<td><em>(PostScript printing only)</em> If a positive number, print the pages in reverse order. If false, print the pages in the regular order.</td>
</tr>
<tr>
<td>bFarEastFontOpt</td>
<td><em>(PostScript printing only)</em> Set to a positive number if the destination printer has multibyte fonts; set to 0 otherwise.</td>
</tr>
<tr>
<td>bEmitHalftones</td>
<td><em>(PostScript printing only)</em> If a positive number, emit the halftones specified in the document. If 0, do not.</td>
</tr>
<tr>
<td>iPageOption</td>
<td>Pages in the range to print. Must be one of: PDAllPages, PDEvenPagesOnly, or PDOddPagesOnly.</td>
</tr>
</tbody>
</table>

Return Value
0 if there were any exceptions while printing, -1 otherwise.
Related Methods

AVDoc.PrintPages
AVDoc.PrintPagesEx
AVDoc.PrintPagesSilentEx
SetFrame

VARIANT_BOOL SetFrame(LPDISPATCH iAcroRect);

Description

Sets the window's size and location.

Parameters

| iAcroRect | The LPDISPATCH for an AcroExch.Rect specifying the window frame. iAcroRect’s instance variable m_lpDispatch contains this LPDISPATCH. |

Return Value

Always returns -1.

Related Methods

AVDoc.GetFrame
SetTextSelection

VARIANT_BOOL SetTextSelection(LPDISPATCH iAcroPDTextSelect);

Description
Sets the document's selection to the specified text selection. Before calling this method, use one of the following to create the text selection:

- PDDoc.CreateTextSelect — Creates from a rectangle
- PDPAGE.CreatePageHilite — Creates from a list of character offsets and counts
- PDPAGE.CreateWordHilite — Creates from a list of word offsets and counts

After calling this method, use AVDoc.ShowTextSelect to show the selection.

Parameters

- iAcroPDTextSelect The LPDISPATCH for the text selection to use. 
  iAcroPDTextSelect contains the instance variable m_lpDispatch, which contains the LPDISPATCH.

Return Value
Returns -1 if successful. Returns 0 if no document is open or the LPDISPATCH is not a PDTextSelect object.

Related Methods
AVDoc.ClearSelection
AVDoc.ShowTextSelect
PDDoc.CreateTextSelect
PDPAGE.CreatePageHilite
PDPAGE.CreateWordHilite
PDTextSelect.Destroy
PDTextSelect.GetBoundingRect
PDTextSelect.GetNumText
PDTextSelect.GetPage
PDTextSelect.GetText
**SetTitle**

```c
VARIANT_BOOL SetTitle(BSTR szTitle);
```

**Description**

Sets the window's title.

**Parameters**

| szTitle | The title to be set. This method cannot be used for document windows, but only for windows created by plug-ins. |

**Return Value**

Returns 0 if no document is open, -1 otherwise.

**Related Methods**

- `AVDoc.GetTitle`
- `AVDoc.Open`
- `PDDoc.OpenAVDoc`
**SetViewMode**

```c
VARIANT_BOOL SetViewMode(long nType);
```

**Description**

Sets the mode in which the document will be viewed (pages only, pages and thumbnails, or pages and bookmarks).

**Parameters**

- **nType**
  
  The view mode to be set. Must be one of the values (except for **PDFullScreen**) listed in View Mode (see document in Chapter 1, "Apple Event Objects" for an explanation of the View Mode property).
  
  Possible values:
  - **PDDontCare** (0 - leave the view mode as it is)
  - **PDUseNone** (1 - display without bookmarks or thumbnails)
  - **PDUseThumbs** (2 - display using thumbnails)
  - **PDUseBookmarks** (3 - display using bookmarks)
  - **PDFullScreen** (4 - display in full screen mode)

  **Note**: Do not set the view mode with `AVDoc.SetViewMode`.

**Return Value**

- 0 if an error occurred while setting the view mode or if no document was open, -1 otherwise.

**Related Methods**

- `AVDoc.GetAVPageView`
- `AVDoc.GetViewMode`
**ShowTextSelect**

```vba
VARIANT_BOOL ShowTextSelect();
```

**Description**
Changes the view so that the current text selection is visible.

**Parameters**
None

**Return Value**
Returns 0 if no document is open, -1 otherwise.

**Related Methods**
- AVDoc.ClearSelection
- AVDoc.SetTextSelection
- PDDoc.CreateTextSelect
- PDPage.CreatePageHilite
- PDPage.CreateWordHilite
- PDTextSelect.Destroy
- PDTextSelect.GetBoundingRect
- PDTextSelect.GetNumText
- PDTextSelect.GetPage
- PDTextSelect.GetText
**AcroExch.AVPageView**

All methods in this section belong to the *AcroExch.AVPageView* class.

---

**DevicePointToPage**

```c
LPDISPATCH DevicePointToPage(LPDISPATCH iAcroPoint);
```

**Description**

Converts the coordinates of a point from device space to user space.

**Parameters**

- **iAcroPoint**
  
> The LPDISPATCH for the *AcroExch.Point* whose coordinates are converted. *iAcroPoint* contains the instance variable `m_lpDispatch`, which contains the LPDISPATCH.

**Return Value**

The LPDISPATCH for an *AcroExch.Point* containing the converted coordinates.

**Related Methods**

*AVPageView.PointToDevice*
DoGoBack

VARIANT_BOOL DoGoBack();

Description
Goes to the previous view on the view history stack, if any.

Parameters
None

Return Value
Always returns -1.

Related Methods
AVPageView.DoGoForward
DoGoForward

    VARIANT_BOOL DoGoForward();

Description

    Goes to the next view on the view history stack, if any.

Parameters

    None

Return Value

    Always returns -1.

Related Methods

    AVPageView.DoGoBack
GetAperture

CAcroRect* GetAperture();

Description
Gets the aperture of the specified page view. The aperture is the rectangular region of the window in which the document is drawn, measured in device space units.

Parameters
None

Return Value
A pointer to the aperture rectangle. Its coordinates are specified in device space.

Related Methods
AVDoc.GetAVPageView
AVPageView.GetAVDoc
AVPageView.GetDoc
AVPageView.GetPage
AVPageView.GetZoomType
GetAVDoc

LPDISPATCH GetAVDoc();

Description

Gets the AcroExch.AVDoc associated with the current page.

Parameters

None

Return Value

The LPDISPATCH for the AcroExch.AVDoc.

Related Methods

AVDoc.GetAVPageView
AVDoc.GetPDDoc
AVPageView.GetDoc
**GetDoc**

```cpp
LPDISPATCH GetDoc();
```

**Description**

Gets the `AcroExch.PDDoc` corresponding to the current page.

**Parameters**

- None

**Return Value**

The `LPDISPATCH` for the `AcroExch.PDDoc`.

**Related Methods**

- `AVDoc.GetAVPageView`
- `AVDoc.GetPDDoc`
- `AVPageView.GetAVDoc`
GetPage

LPDISPATCH GetPage();

Description
Gets the AcroExch.PDPage corresponding to the current page.

Parameters

None

Return Value
The LPDISPATCH for the AcroExch.PDPage.

Related Methods
AVPageView.GetPageNum
PDDoc.AcquirePage
PDDoc.GetNumPages
PDPage.GetDoc
PDPage.GetNumber
PDPage.GetRotate
PDPage.GetSize
PDTextSelect.GetPage
GetPageNum

    long GetPageNum();

Description

Gets the page number of the current page. The first page in a document is page zero.

Parameters

None

Return Value

The current page’s page number.

Related Methods

AVPageView.GetPage
PDDoc.AcquirePage
PDDoc.GetNumPages
PDPage.GetDoc
PDPage.GetNumber
PDPage.GetRotate
PDPage.GetSize
PDTextSelect.GetPage
GetZoom

long GetZoom();

Description

Gets the current zoom factor, specified as a percent (for example, 100 is returned if the
magnification is 1.0).

Parameters

None

Return Value

The current zoom factor.

Related Methods

App.GetPreference
AVPageView.GetZoomType
AVPageView.ZoomTo
GetZoomType

    short GetZoomType();

Description

    Gets the current zoom type.

Parameters

    None

Return Value

    Zoom type. See zoom type (in document in Chapter 1, “Apple Event Objects”) for a list of zoom types.

Related Methods

    App.GetPreference
    AVPageView.GetZoomType
    AVPageView.ZoomTo
OLE Automation Methods
AcroExch.AVPageView

Goto

VARIANT_BOOL GoTo(long nPage);

Description
Goes to the specified page.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nPage</td>
<td>Page number of the destination page. The first page in a PDDoc object is page 0.</td>
</tr>
</tbody>
</table>

Return Value
-1 if the Acrobat application successfully went to the page, 0 otherwise.

Related Methods
AVPageView.DoGoBack
AVPageView.DoGoForward
AVPageView.ReadPageDown
AVPageView.ReadPageUp
AVPageView.ScrollTo
AVPageView.ZoomTo
PointToDevice

LPDISPATCH PointToDevice(LPDISPATCH iAcroPoint);

Description

Converts the coordinates of a point from user space to device space.

**IMPORTANT:** *Deprecated: do not use this method.*

Parameters

- **iAcroPoint**
  
  The LPDISPATCH for the AcroExch.Point whose coordinates are converted. *iAcroPoint* contains the instance variable *m_lpDispatch*, which contains this LPDISPATCH.

Return Value

The LPDISPATCH for an AcroExch.Point containing the converted coordinates.

Related Methods

- AVPageView.DevicePointToPage
ReadPageDown

VARIANT_BOOL ReadPageDown();

Description
Scrolls forward through the document by one screen area.

Parameters
None

Return Value
Always returns -1.

Related Methods
- AVPageView.DoGoBack
- AVPageView.DoGoForward
- AVPageView.Goto
- AVPageView.ReadPageUp
- AVPageView.ScrollTo
- AVPageView.ZoomTo
ReadPageUp

VARIANT_BOOL ReadPageUp();

Description
Scrolls backward through the document by one screen area.

Parameters
None

Return Value
Always returns -1.

Related Methods
AVPageView::DoGoBack
AVPageView::DoGoForward
AVPageView::Goto
AVPageView::ReadPageDown
AVPageView::ScrollTo
AVPageView::ZoomTo
ScrollTo

VARIANT_BOOL ScrollTo(short nX, short nY);

Description
Scrolls to the specified location on the current page.

Parameters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>nX</td>
<td>x-coordinate of the destination.</td>
</tr>
<tr>
<td>nY</td>
<td>y-coordinate of the destination.</td>
</tr>
</tbody>
</table>

Return Value
-1 if the Acrobat application successfully scrolled to the specified location, 0 otherwise.

Related Methods
AVPageView.DoGoBack
AVPageView.DoGoForward
AVPageView.Goto
AVPageView.ReadPageDown
AVPageView.ReadPageUp
AVPageView.ZoomTo
ZoomTo

VARIANT_BOOL ZoomTo(short nType, short nScale); 

Description
Zooms to the specified magnification.

Parameters

<table>
<thead>
<tr>
<th>nType</th>
<th>Zoom type. See zoom type (in document in Chapter 1, “Apple Event Objects) for a list of zoom types.</th>
</tr>
</thead>
<tbody>
<tr>
<td>nScale</td>
<td>The desired zoom factor, expressed as a percentage (for example, 100 is a magnification of 1.0).</td>
</tr>
</tbody>
</table>

Return Value
-1 if the magnification was set successfully, 0 otherwise.

Related Methods

AVPageView.GetZoomType
AVPageView.Goto
AVPageView.ScrollTo
**AcroExch.HiliteList**

The methods in this section work with highlights and highlight lists.

**Add**

```c
VARIANT_BOOL Add(short nOffset, short nLength);
```

**Description**

Adds the highlight specified by `nOffset` and `nLength` to the current highlight list. Highlight lists are used to highlight one or more contiguous groups of characters or words on a single page.

Highlight lists are used both for character- and word-based highlighting, although a single highlight list cannot contain a mixture of character and word highlights. After creating a highlight list, use `PDPAGE.CreatePageHilite` or `PDPAGE.CreateWordHilite` (depending on whether the highlight list is used for characters or words) to create a text selection from the highlight list.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nOffset</code></td>
<td>Offset of the first word or character to be highlighted, the first of which has an offset of zero.</td>
</tr>
<tr>
<td><code>nLength</code></td>
<td>The number of consecutive words or characters to be highlighted.</td>
</tr>
</tbody>
</table>

**Return Value**

Always returns -1.

**Related Methods**

- `PDPAGE.CreatePageHilite`
- `PDPAGE.CreateWordHilite`
AcroExch.PDAnnot

All methods in this section belong to the AcroExch.PDAnnot class.

**GetColor**

```c
long GetColor();
```

**Description**

Gets an annotation's color.

**Parameters**

None

**Return Value**

The annotation's color, a long value of the form 0x00BBGGRR where the first byte from the right (RR) is a relative value for red, the second byte (GG) is a relative value for green, and the third byte (BB) is a relative value for blue. The high-order byte must be 0.

**Related Methods**

PDAnnot.SetColor
GetContents

BSTR GetContents();

Description

Gets a text annotation's contents.

Parameters

None

Return Value

The annotation's contents.

Related Methods

PDAnnot.SetContents
PDAnnot.GetDate
PDAnnot.GetRect
PDAnnot.GetSubtype
PDAnnot.GetTitle
**GetDate**

```cpp
LPDISPATCH GetDate();
```

**Description**

Gets an annotation's date.

**Parameters**

None

**Return Value**

The `LPDISPATCH` for an `AcroExch.Time` object containing the date.

**Related Methods**

- `PDAnnot.GetContents`
- `PDAnnot.GetRect`
- `PDAnnot.GetSubtype`
- `PDAnnot.GetTitle`
- `PDAnnot.SetDate`
GetRect

LPDISPATCH GetRect();

Description

Gets an annotation's bounding rectangle.

Parameters

None

Return Value

The LPDISPATCH for an AcroExch.Rect containing the annotation's bounding rectangle.

Related Methods

PDAnnot.GetContents
PDAnnot.GetDate
PDAnnot.GetSubtype
PDAnnot.GetTitle
PDAnnot.SetRect
**GetSubtype**

```c
BSTR GetSubtype();
```

**Description**

Gets an annotation's subtype.

**Parameters**

None

**Return Value**

The annotation's subtype. The built-in subtypes are "Text" and "Link".

**Related Methods**

- `PDAnnot.GetContents`
- `PDAnnot.GetDate`
- `PDAnnot.GetRect`
- `PDAnnot.GetTitle`
**GetTitle**

BSTR GetTitle();

**Description**

Gets a text annotation's title.

**Parameters**

None

**Return Value**

The annotation's title.

**Related Methods**

PDAnnot.GetContents
PDAnnot.GetDate
PDAnnot.GetRect
PDAnnot.GetSubtype
PDAnnot.SetTitle
IsEqual

VARIANT_BOOL IsEqual(LPDISPATCH PDAnnot);

Description
Determines whether an annotation is the same as the specified annotation.

Parameters

| PDAnnot | The LPDISPATCH for the AcroExch.PDAnnot to be tested. PDAnnot contains the instance variable m_lpDispatch, which contains the LPDISPATCH. |

Return Value
-1 if the annotations are the same, 0 otherwise.

Related Methods
PDAnnot.GetContents
PDAnnot.GetDate
PDAnnot.GetRect
PDAnnot.GetSubtype
PDAnnot.GetTitle
PDAnnot.IsOpen
PDAnnot.IsValid
IsOpen

VARIANT_BOOL IsOpen();

Description
Tests whether a text annotation is open.

Parameters

None

Return Value

-1 if open, 0 otherwise.

Related Methods

PDAnnot.GetContents
PDAnnot.GetDate
PDAnnot.GetRect
PDAnnot.GetSubtype
PDAnnot.GetTitle
PDAnnot.IsEqual
PDAnnot.IsValid
PDAnnot.SetOpen
IsValid

VARIANT_BOOL IsValid();

Description
Tests whether an annotation is still valid. This method is intended only to test whether the annotation has been deleted, not whether it is a completely valid annotation object.

Parameters
None

Return Value
-1 if the annotation is valid, 0 otherwise.

Related Methods
PDAnnot.GetContents
PDAnnot.GetDate
PDAnnot.GetRect
PDAnnot.GetSubtype
PDAnnot.GetTitle
PDAnnot.AreEqual
PDAnnot.IsOpen
Perform

VARIANT_BOOL Perform(LPDISPATCH iAcroAVDoc);

Description
Performs a link annotation's action.

Parameters

| iAcroAVDoc | The LPDISPATCH for the AcroExch.AVDoc in which the annotation is located. iAcroAVDoc contains the instance variable m_lpDispatch, which contains the LPDISPATCH. |

Return Value
-1 if the action was executed successfully, 0 otherwise.

Related Methods
PDAnnot.IsValid
SetColor

VARIANT_BOOL SetColor(long nRGBColor);

Description
Sets an annotation's color.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nRGBColor</td>
<td>The color to use for the annotation.</td>
</tr>
</tbody>
</table>

Return Value

-1 if the annotation’s color was set, 0 if the Acrobat application does not support editing.

nRGBColor is a long value with the form 0x00BBGGRR where the first byte from the right (RR) is a relative value for red, the second byte (GG) is a relative value for green, and the third byte (BB) is a relative value for blue. The high-order byte must be 0.

Related Methods

PDAnnot以色
PDAnnot.SetContents
PDAnnot.SetDate
PDAnnot.SetOpen
PDAnnot.SetRect
PDAnnot.SetTitle
SetContents

VARIANT_BOOL SetContents(BSTR szContents);

Description
Sets a text annotation's contents.

Parameters

| szContents  | The contents to use for the annotation. |

Return Value
0 if the Acrobat application does not support editing, -1 otherwise.

Related Methods

PDAnnot.GetContents
PDAnnot.SetColor
PDAnnot.SetDate
PDAnnot.SetOpen
PDAnnot.SetRect
PDAnnot.SetTitle
SetDate

VARIANT_BOOL SetDate(LPDISPATCH iAcroTime);

Description

Sets an annotation's date.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iAcroTime</td>
<td>The LPDISPATCH for the date and time to use for the annotation. iAcroTime's instance variable m_lpDispatch contains this LPDISPATCH.</td>
</tr>
</tbody>
</table>

Return Value

-1 if the date was set, 0 if the Acrobat application does not support editing.

Related Methods

PDAnnot.GetTitle
PDAnnot.SetColor
PDAnnot.SetContents
PDAnnot.SetOpen
PDAnnot.SetRect
PDAnnot.SetTitle
SetOpen

VARIANT_BOOL SetOpen(long bIsOpen);

**Description**
Opens or closes a text annotation.

**Parameters**

<table>
<thead>
<tr>
<th>bIsOpen</th>
<th>If a positive number, the annotation is open. If 0, the annotation is closed.</th>
</tr>
</thead>
</table>

**Return Value**
Always returns -1.

**Related Methods**
- PDAnnot.IsOpen
- PDAnnot.SetColor
- PDAnnot.SetContents
- PDAnnot.SetDate
- PDAnnot.SetRect
- PDAnnot.SetTitle
SetRect

VARIANT_BOOL SetRect(LPDISPATCH iAcroRect);

Description
Sets an annotation's bounding rectangle.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iAcroRect</td>
<td>The LPDISPATCH for the bounding rectangle (AcroExch.Rect) to set. iAcroRect contains the instance variable m_lpDispatch, which contains the LPDISPATCH.</td>
</tr>
</tbody>
</table>

Return Value
-1 if a rectangle was supplied, 0 otherwise.

Related Methods
PDAnnot.GetRect
PDAnnot.SetColor
PDAnnot.SetContents
PDAnnot.SetDate
PDAnnot.SetOpen
PDAnnot.SetTitle
SetTitle

VARIANT_BOOL SetTitle(BSTR szTitle);

Description
Sets a text annotation's title.

Parameters

| szTitle       | The title to use. |

Return Value
-1 if the title was set, 0 if the Acrobat application does not support editing.

Related Methods
PDAnnot.GetByTitle
PDAnnot.SetColor
PDAnnot.SetContents
PDAnnot.SetDate
PDAnnot.SetOpen
PDAnnot.SetRect
**AcroExch.PDBookmark**

The methods in this section relate to Acrobat bookmarks.

---

**Destroy**

```c
VARIANT_BOOL Destroy();
```

**Description**

Destroys a bookmark. (You can create bookmarks in OLE. See *Programming Acrobat JavaScript Using Visual Basic*).

**Parameters**

None

**Return Value**

0 if the Acrobat application does not support editing (making it impossible to delete the bookmark), -1 otherwise.

**Related Methods**

*PDBookmark.IsValid*
**GetByTitle**

```cpp
VARIANT_BOOL GetByTitle(LPDISPATCH iAcroPDDoc,
BSTR bookmarkTitle);
```

**Description**

Gets the bookmark that has the specified title. The `AcroExch.PDBookmark` object is set to the specified bookmark as a side effect of the method; it is not the method's return value. You cannot enumerate bookmark titles with this method.

**Parameters**

| iAcroPDDoc | The LPDISPATCH for the document (AcroExch.PDDoc object) containing the bookmark. iAcroPDDoc contains the instance variable `m_lpDispatch`, which contains the LPDISPATCH. |
| bookmarkTitle | The title of the bookmark to get. The capitalization of the title must match that in the bookmark. |

**Return Value**

-1 if the specified bookmark exists (the method determines this using the `PDBookmark.IsValid` method), 0 otherwise.

**Related Methods**

- `PDBookmark.GetTitle`
- `PDBookmark.SetTitle`

**Example**

```cpp
long b;
CAcroPDBookmark;
bookmark = new AcroPDBookmark;
COleException e;

if (!bookmark->CreateDispatch(
    "AcroExch.PDBookmark", &e))
    AfxMessageBox("Failed to create PDBookmark object.");

b = bookmark->GetByTitle(pddoc, "Name of bookmark");
if (b)
    bookmark->Perform();
```
**GetTitle**

```c
BSTR GetTitle();
```

**Description**

Gets a bookmark's title.

**Parameters**

None

**Return Value**

The title.

**Related Methods**

- `PDBookmark.GetByTitle`
- `PDBookmark.SetTitle`
**IsValid**

```c
VARIANT_BOOL IsValid();
```

**Description**
Determines whether the bookmark is valid. This method only checks whether the bookmark has been deleted; it does not thoroughly check the bookmark’s data structures.

**Parameters**
None

**Return Value**
-1 if the bookmark is valid, 0 otherwise.

**Related Methods**
- `PDBookmark.Destroy`
Perform

VARIANT_BOOL Perform(LPDISPATCH iAcroAVDoc);

Description
Performs a bookmark's action.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iAcroAVDoc</td>
<td>The LPDISPATCH for the AcroExch.AVDoc in which the bookmark is located. iAcroAVDoc contains the instance variable m_lpDispatch, which contains the LPDISPATCH.</td>
</tr>
</tbody>
</table>

Return Value
-1 if the action was executed successfully, 0 otherwise.

Related Methods
PDBBookmark.IsValid
**SetTitle**

```c
VARIANT_BOOL SetTitle(BSTR szNewTitle);
```

**Description**
Sets a bookmark's title.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>szNewTitle</td>
<td>The title to set.</td>
</tr>
</tbody>
</table>

**Return Value**

0 if the Acrobat application does not support editing, -1 otherwise.

**Related Methods**

- PDBookmark.GetByTitle
- PDBookmark.GetTitle
AcroExch.PDDoc

The methods in this section work with documents.

AcquirePage

 LPDISPATCH AcquirePage(long nPage);

Description

Acquires the specified page.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nPage</td>
<td>The number of the page to acquire. The first page is page 0.</td>
</tr>
</tbody>
</table>

Return Value

The LPDISPATCH for the AcroExch.PDPage object for the acquired page. Returns NULL if the page could not be acquired.

Related Methods

AVPageView.GetPage
AVPageView.GetPageNum
PDDoc.GetNumPages
PDPage.GetDoc
PDPage.GetNumber
PDPage.GetRotate
PDPage.GetSize
PDTextSelect.GetPage
## ClearFlags

```c
VARIANT_BOOL ClearFlags(long nFlags);
```

### Description

Clears a document’s flags. The flags indicate whether the document has been modified, whether the document is a temporary document and should be deleted when closed, and the version of PDF used in the file. This method can only be used to clear, not to set, the flag bits.

### Parameters

<table>
<thead>
<tr>
<th>nFlags</th>
<th>Flags to be cleared. See <code>PDDoc.GetFlags</code> for a description of the flags. The flags <code>PDDocWasRepaired</code>, <code>PDDocNewMajorVersion</code>, <code>PDDocNewMinorVersion</code>, and <code>PDDocOldVersion</code> are read-only and cannot be cleared.</th>
</tr>
</thead>
</table>

### Return Value

Always returns `-1`.

### Related Methods

- `PDDoc.GetFlags`
- `PDDoc.SetFlags`
Close

VARIANT_BOOL Close();

Description

Closes a file.

NOTE: If PDDoc and AVDoc are constructed with the same file, PDDoc.Close will destroy both objects (which closes the document in the viewer).

Parameters

None

Return Value

-1 if the document was closed successfully, 0 otherwise.

Related Methods

App.CloseAllDocs
AVDoc.Close
AVDoc.Open
AVDoc.OpenInWindow
AVDoc.OpenInWindowEx
PDDoc.Open
PDDoc.OpenAVDoc
Create

VARIANT_BOOL Create();

Description

Creates a new AcroExch.PDDoc.

Parameters

None

Return Value

-1 if the document is created successfully, 0 if it is not or if the Acrobat application does not support editing.

Related Methods

None
CreateTextSelect

LPDISPATCH CreateTextSelect(long nPage, LPDISPATCH iAcroRect);

Description

Creates a text selection from the specified rectangle on the specified page. After creating the text selection, use the AVDoc.SetTextSelection method to use it as the document's selection, and use AVDoc.ShowTextSelect to show the selection.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nPage</td>
<td>The page on which the selection is created. The first page in a PDDoc object is page 0.</td>
</tr>
<tr>
<td>iAcroRect</td>
<td>The LPDISPATCH for the AcroExch.Rect enclosing the region to select. iAcroRect contains the instance variable m_lpDispatch, which contains the LPDISPATCH.</td>
</tr>
</tbody>
</table>

Return Value

The LPDISPATCH for an AcroExch.PDTextSelect containing the text selection. Returns NULL if the text selection was not created successfully.

Related Methods

AVDoc.ClearSelection
AVDoc.SetTextSelection
AVDoc.ShowTextSelect
PDPPage.CreatePageHilite
PDPPage.CreateWordHilite
PDTextSelect.Destroy
PDTextSelect.GetBoundingRect
PDTextSelect.GetNumText
PDTextSelect.GetPage
PDTextSelect.GetText
CreateThumbs

VARIANT_BOOL CreateThumbs(long nFirstPage, long nLastPage);

Description

Creates thumbnail images for the specified page range in a document.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nFirstPage</td>
<td>First page for which thumbnail images are created. The first page in a PDDoc object is page 0.</td>
</tr>
<tr>
<td>nLastPage</td>
<td>Last page for which thumbnail images are created.</td>
</tr>
</tbody>
</table>

Return Value

-1 if thumbnail images were created successfully, 0 if they were not or if the Acrobat application does not support editing.

Related Methods

PDDoc.DeleteThumbs
**CropPages**

```cpp
VARIANT_BOOL CropPages(long nStartPage, long nEndPage,
                        short nEvenOrOddPagesOnly,
                        LPDISPATCH iAcroRect);
```

**Description**

Crops the pages in a specified range in a document. This method ignores the request if either the width or height of the crop box is less than 72 points (one inch).

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nStartPage</td>
<td>First page that is cropped. The first page in a <code>PDDoc</code> object is page 0.</td>
</tr>
<tr>
<td>nEndPage</td>
<td>Last page that is cropped.</td>
</tr>
</tbody>
</table>
| nEvenOrOddPagesOnly     | Value indicating which pages in the range are cropped. Must be one of the following:  
                          ● 0 means crop all pages in the range  
                          ● 1 means crop only odd pages in the range  
                          ● 2 means crop only even pages in the range  |
| iAcroRect               | An `LPDISPATCH` for a `CAcroRect` specifying the cropping rectangle, which is specified in user space. |

**Return Value**

-1 if the pages were cropped successfully, 0 otherwise.

**Related Methods**

`PDPage.CropPages`
DeletePages

VARIANT_BOOL DeletePages(long nStartPage, long nEndPage);

Description

Deletes pages from a file.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nStartPage</td>
<td>The first page to be deleted. The first page in a PDDoc object is page 0.</td>
</tr>
<tr>
<td>nEndPage</td>
<td>The last page to be deleted.</td>
</tr>
</tbody>
</table>

Return Value

-1 if the pages were successfully deleted. Returns 0 if they were not or if the Acrobat application does not support editing.

Related Methods

- PDDoc.AcquirePage
- PDDoc.DeletePages
- PDDoc.GetNumPages
- PDDoc.InsertPages
- PDDoc.MovePage
- PDDoc.ReplacePages
DeleteThumbs

VARIANT_BOOL DeleteThumbs(long nStartPage, long nEndPage);

**Description**

Deletes thumbnail images from the specified pages in a document.

**Parameters**

- **nStartPage**: First page whose thumbnail image is deleted. The first page in a PDDoc object is page 0.
- **nEndPage**: Last page whose thumbnail image is deleted.

**Return Value**

-1 if the thumbnails were deleted, 0 if they were not deleted or if the Acrobat application does not support editing.

**Related Methods**

PDDoc.CreateThumbs
GetFileName

BSTR GetFileName();

Description

Gets the name of the file associated with this AcroExch.PDDoc.

Parameters

None

Return Value

The file name, which can currently contain up to 256 characters.

Related Methods

PDDoc.Save
GetFlags

long GetFlags();

Description
Gets a document’s flags. The flags indicate whether the document has been modified, whether the document is a temporary document and should be deleted when closed, and the version of PDF used in the file.

Parameters
None

Return Value
The document’s flags, containing an OR of the following:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDDocNeedsSave</td>
<td>Document has been modified and needs to be saved.</td>
</tr>
<tr>
<td>PDDocRequiresFullSave</td>
<td>Document cannot be saved incrementally; it must be written using PDSaveFull.</td>
</tr>
<tr>
<td>PDDocIsModified</td>
<td>Document has been modified slightly (such as bookmarks or text annotations have been opened or closed), but not in a way that warrants saving.</td>
</tr>
<tr>
<td>PDDocDeleteOnClose</td>
<td>Document is based on a temporary file that must be deleted when the document is closed or saved.</td>
</tr>
<tr>
<td>PDDocWasRepaired</td>
<td>Document was repaired when it was opened.</td>
</tr>
<tr>
<td>PDDocNewMajorVersion</td>
<td>Document’s major version is newer than current.</td>
</tr>
<tr>
<td>PDDocNewMinorVersion</td>
<td>Document’s minor version is newer than current.</td>
</tr>
<tr>
<td>PDDocOldVersion</td>
<td>Document’s version is older than current.</td>
</tr>
<tr>
<td>PDDocSuppressErrors</td>
<td>Don’t display errors.</td>
</tr>
</tbody>
</table>

Related Methods
PDDoc.ClearFlags
PDDoc.SetFlags
GetInfo

BSTR GetInfo(BSTR szInfoKey);

Description

Gets the value of a specified key in the document’s Info dictionary. A maximum of 512 bytes are returned.

Parameters

| szInfoKey | The key whose value is obtained. |

Return Value

The string if the value was read successfully. Returns an empty string if the key does not exist or its value cannot be read.

Related Methods

PDDoc.SetInfo
GetInstanceID

BSTR GetInstanceID();

**Description**

Gets the instance ID (the second element) from the ID array in the document’s trailer.

**Parameters**

None

**Return Value**

A string whose maximum length is 32 characters, containing the document’s instance ID.

**Related Methods**

PDDoc.GetPermanentID
GetJSObject

LDispatch* GetJSObject();

Description

Gets a dual interface to the JavaScript object associated with the PDDoc. This allows Automation clients full access to both built-in and user-defined JavaScript methods available in the document. For detailed information on this method, see Programming Acrobat JavaScript Using Visual Basic.

Parameters

None

Return Value

The interface to the JavaScript object if the call succeeded, NULL otherwise.

Related Methods

None.
GetNumPages

```c
long GetNumPages();
```

**Description**

Gets the number of pages in a file.

**Parameters**

None

**Return Value**

The number of pages, or -1 if the number of pages cannot be determined.

**Related Methods**

- AVPageView.GetPage
- AVPageView.GetPageNum
- PDDoc.AcquirePage
- PDPage.GetNumber
- PDTextSelect.GetPage
GetPageMode

long GetPageMode();

Description
Gets a value indicating whether the Acrobat application is currently displaying only pages, pages and thumbnails, or pages and bookmarks.

Parameters
None

Return Value
The current page mode. Will be one of the values listed in View Mode (see document in Chapter 1, “Apple Event Objects,” for an explanation of the View Mode property).

Related Methods
PDDoc.SetPageMode
GetPermanentID

```c
BSTR GetPermanentID();
```

**Description**

Gets the permanent ID (the first element) from the ID array in the document’s trailer.

**Parameters**

None

**Return Value**

A string whose maximum length is 32 characters, containing the document’s permanent ID.

**Related Methods**

PDDoc.GetInstanceID
**InsertPages**

```c
VARIANT_BOOL InsertPages(long nInsertPageAfter,
                          LPDISPATCH iPDDocSource,long nStartPage,
                          long nNumPages, long bBookmarks);
```

**Description**

Inserts the specified pages from the source document after the indicated page within the current document.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nInsertPageAfter</td>
<td>The page in the current document after which pages from the source document are inserted. The first page in a PDDoc object is page 0.</td>
</tr>
<tr>
<td>iPDDocSource</td>
<td>The LPDISPATCH for the AcroExch.PDDoc containing the pages to insert. iPDDocSource contains the instance variable m_lpDispatch, which contains the LPDISPATCH.</td>
</tr>
<tr>
<td>nStartPage</td>
<td>The first page in iPDDocSource to be inserted into the current document.</td>
</tr>
<tr>
<td>nNumPages</td>
<td>The number of pages to be inserted.</td>
</tr>
<tr>
<td>bBookmarks</td>
<td>If a positive number, bookmarks are copied from the source document. If 0, they are not.</td>
</tr>
</tbody>
</table>

**Return Value**

-1 if the pages were successfully inserted. Returns 0 if they were not or if the Acrobat application does not support editing.

**Related Methods**

- PDDoc.AcquirePage
- PDDoc.DeletePages
- PDDoc.GetNumPages
- PDDoc.MovePage
- PDDoc.ReplacePages
OLE Automation Methods

AcroExch.PDDoc

MovePage

VARIANT_BOOL MovePage(long nMoveAfterThisPage,
long nPageToMove);

Description

Moves a page to another location within the same document.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nMoveAfterThisPage</td>
<td>The page being moved is placed after this page number. The first page in a PDDoc object is page 0.</td>
</tr>
<tr>
<td>nPageToMove</td>
<td>Page number of the page to be moved.</td>
</tr>
</tbody>
</table>

Return Value

0 if the Acrobat application does not support editing, -1 otherwise.

Related Methods

PDDoc.AcquirePage
PDDoc.DeletePages
PDDoc.GetNumPages
PDDoc.InsertPages
PDDoc.ReplacePages
Open

VARIANT_BOOL Open(BSTR szFullPath);

Description

Opens a file. A new instance of AcroExch.PDDoc must be created for each open PDF file.

Parameters

| szFullPath          | Full pathname of the file to be opened. |

Return Value

-1 if the document was opened successfully, 0 otherwise.

Related Methods

App.CloseAllDocs
AVDoc.Close
AVDoc.Open
AVDoc.OpenInWindow
AVDoc.OpenInWindowEx
PDDoc.Close
PDDoc.OpenAVDoc
OpenAVDoc

LPDISPATCH OpenAVDoc(BSTR szTitle);

Description
Opens a window and displays the document in it.

Parameters

| szTitle | The title to be used for the window. A default title is used if szTitle is NULL or an empty string. |

Return Value
The LPDISPATCH for the AcroExch.AVDoc that was opened, or NULL if the open fails.

Related Methods
- App.CloseAllDocs
- AVDoc.Close
- AVDoc.GetTitle
- AVDoc.Open
- AVDoc.OpenInWindow
- AVDoc.OpenInWindowEx
- AVDoc.SetTitle
- PDDoc.Close
- PDDoc.Open
ReplacePages

VARIANT_BOOL ReplacePages(long nStartPage,
LPDISPATCH iPDDocSource,
long nStartSourcePage, long nNumPages,
long bMergeTextAnnotations);

Description
Replaces the indicated pages in the current document with those specified from the source document. No links or bookmarks are copied from iPDDocSource, but text annotations may optionally be copied.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nStartPage</td>
<td>The first page within the source file to be replaced. The first page in a PDDoc object is page 0.</td>
</tr>
<tr>
<td>iPDDocSource</td>
<td>The LPDISPATCH for the AcroExch.PDDoc containing the new copies of pages that are replaced. iPDDocSource contains the instance variable m_lpDispatch, which contains the LPDISPATCH.</td>
</tr>
<tr>
<td>nStartSourcePage</td>
<td>The first page in iPDDocSource to use as a replacement page.</td>
</tr>
<tr>
<td>nNumPages</td>
<td>The number of pages to be replaced.</td>
</tr>
<tr>
<td>bMergeTextAnnotations</td>
<td>If a positive number, text annotations from iPDDocSource are copied. If 0, they are not.</td>
</tr>
</tbody>
</table>

Return Value
-1 if the pages were successfully replaced. Returns 0 if they were not or if the Acrobat application does not support editing.

Related Methods
- PDDoc.AcquirePage
- PDDoc.DeletePages
- PDDoc.GetNumPages
- PDDoc.InsertPages
- PDDoc.MovePage
Save

VARIANT_BOOL Save(short nType, BSTR szFullPath);

Description

Saves a document.

Parameters

<table>
<thead>
<tr>
<th>nType</th>
<th>Specifies the way in which the file should be saved.</th>
</tr>
</thead>
<tbody>
<tr>
<td>nType</td>
<td>A file may be saved as a linearized file using the</td>
</tr>
<tr>
<td></td>
<td>PDSaveLinearized flag, but the following sequence</td>
</tr>
<tr>
<td></td>
<td>MUST be observed:</td>
</tr>
<tr>
<td>1.</td>
<td>Open the PDF file with PDDoc.Open.</td>
</tr>
<tr>
<td>2.</td>
<td>Call PDDoc.Save using the PDSaveLinearized flag.</td>
</tr>
<tr>
<td>nType</td>
<td>This allows batch linearizing of files.</td>
</tr>
</tbody>
</table>

nType is a logical OR of one or more of the following flags:

- **PDSaveIncremental**: Write changes only, not the complete file. This will always result in a larger file, even if objects have been deleted.
- **PDSaveFull**: Write the entire file to the filename specified by szFullPath.
- **PDSaveCopy**: Write a copy of the file into the file specified by szFullPath, but keep using the old file. This flag can only be specified if PDSaveFull is also used.
- **PDSaveCollectGarbage**: Remove unreferenced objects; this often reduces the file size, and its usage is encouraged. This flag can only be specified if PDSaveFull is also used.
- **PDSaveLinearized**: Save the file in a linearized fashion, providing hint tables. This allows the PDF file to be byte-served. This flag can only be specified if PDSaveFull is also used.

<table>
<thead>
<tr>
<th>szFullPath</th>
<th>The new pathname to the file, if any.</th>
</tr>
</thead>
</table>

Return Value

-1 if the document was successfully saved. Returns 0 if it was not or if the Acrobat application does not support editing.

Related Methods

PDDoc.GetFileName
SetFlags

VARIANT_BOOL SetFlags(long nFlags);

Description
Sets a document's flags indicating whether the document has been modified, whether the document is a temporary document and should be deleted when closed, and the version of PDF used in the file. This method can only be used to set, not to clear, the flag bits.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nFlags</td>
<td>Flags to be set. See PDDoc.GetFlags for a description of the flags. The flags PDDocWasRepaired, PDDocNewMajorVersion, PDDocNewMinorVersion, and PDDocOldVersion are read-only and cannot be set.</td>
</tr>
</tbody>
</table>

Return Value
Always returns -1.

Related Methods
- PDDoc.ClearFlags
- PDDoc.GetFlags
SetInfo

VARIANT_BOOL SetInfo(BSTR szInfoKey, BSTR szBuffer);

Description

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

Parameters

<table>
<thead>
<tr>
<th>szInfoKey</th>
<th>The key whose value is set.</th>
</tr>
</thead>
<tbody>
<tr>
<td>szBuffer</td>
<td>The value to be assigned to the key.</td>
</tr>
</tbody>
</table>

Return Value

-1 if the value was added successfully, 0 if it was not or if the Acrobat application does not support editing.

Related Methods

PDDoc.GetInfo
SetPageMode

VARIANT_BOOL SetPageMode(long nPageMode);

Description

Sets the page mode in which a document is to be opened: display only pages, pages and thumbnails, or pages and bookmarks.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nPageMode</td>
<td>The page mode to be set. Must be one of the values listed in View Mode (see document in Chapter 1, &quot;Apple Event Objects&quot; for an explanation of the View Mode property).</td>
</tr>
</tbody>
</table>

Return Value

Always returns -1.

Related Methods

PDDoc.GetPageMode
PDDoc.SetPageMode
AcroExch.PDPage

The methods in this section work with pages.

AddAnnot

VARIANT_BOOL AddAnnot(long nIndexAddAfter,
                        LPDISPATCH iPDAnnot);

Description

Adds a specified annotation at a specified location in the page's annotation array.

Parameters

<table>
<thead>
<tr>
<th>nIndexAddAfter</th>
<th>Location in the page's annotation array to add the annotation. The first annotation on a page has an index of zero.</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPDAnnot</td>
<td>The LPDISPATCH for the AcroExch.PDAnnot to add. iPDAnnot contains the instance variable m_lpDispatch, which contains the LPDISPATCH.</td>
</tr>
</tbody>
</table>

Return Value

0 if the Acrobat application does not support editing, -1 otherwise.

Related Methods

PDPage.AddNewAnnot
PDPage.RemoveAnnot
AddNewAnnot

LPDISPATCH AddNewAnnot(long nIndexAddAfter, BSTR szSubType,
                          LPDISPATCH iAcroRect);

Description

Creates a new text annotation and adds it to the page.

The newly-created text annotation is not complete until PDAnnot.SetContents has been called in order to fill in the /Contents key.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nIndexAddAfter</td>
<td>Location in the page's annotation array after which to add the annotation. The first annotation on a page has an index of zero.</td>
</tr>
<tr>
<td>szSubType</td>
<td>Subtype of the annotation to be created. Must be text.</td>
</tr>
<tr>
<td>iAcroRect</td>
<td>The LPDISPATCH for the AcroExch.Rect bounding the annotation's location on the page. iAcroRect contains the instance variable m_lpDispatch, which contains the LPDISPATCH.</td>
</tr>
</tbody>
</table>

Return Value

The LPDISPATCH for an AcroExch.PDAnnot object, or NULL if the annotation could not be added.

Related Methods

PDAnnot.SetContents
PDPage.AddAnnot
PDPage.RemoveAnnot
CopyToClipboard

```c
VARIANT_BOOL CopyToClipboard(LPDISPATCH boundRect,
    short nXOrigin, short nYOrigin,
    short nZoom);
```

**Description**

Copies a PDF image to the clipboard without requiring an hWnd or hdc from the client.

*NOTE:* This method is only available on 32-bit systems.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boundRect</td>
<td>The LPDISPATCH for the AcroExch.Rect bounding rectangle in device space coordinates. boundRect contains the instance variable m_lpDispatch, which contains the LPDISPATCH.</td>
</tr>
<tr>
<td>nXOrigin</td>
<td>x–coordinate of the portion of the page to be copied.</td>
</tr>
<tr>
<td>nYOrigin</td>
<td>y–coordinate of the portion of the page to be copied.</td>
</tr>
<tr>
<td>nZoom</td>
<td>Zoom factor at which the page is copied, specified as a percent (for example, 100 corresponds to a magnification of 1.0).</td>
</tr>
</tbody>
</table>

**Return Value**

-1 if the page is successfully copied, 0 otherwise.

**Related Methods**

PDPage.DrawEx
CreatePageHilite

LPDISPATCH CreatePageHilite(LPDISPATCH iAcroHiliteList);

Description

Creates a text selection from a list of character offsets and character counts on a single page. The text selection can then be set as the current selection using AVDoc.SetTextSelection, and the view can be set to show the selection using AVDoc.ShowTextSelect.

NOTE: As in the Acrobat application, the text selection always consists of whole words.

Parameters

| iAcroHiliteList | The LPDISPATCH for the highlight list for which a text selection is created. iAcroHiliteList contains the instance variable m_lDispatch, which contains the LPDISPATCH. Use HiliteList.Add to create a highlight list. |

Return Value

The LPDISPATCH for the AcroExch.PDTextSelect containing the text selection, or NULL if the selection could not be created.

Related Methods

AVDoc.ClearSelection
AVDoc.SetTextSelection
AVDoc.ShowTextSelect
HiliteList.Add
PDDoc.CreateTextSelect
PDPage.CreateWordHilite
PDTextSelect.Destroy
PDTextSelect.GetBoundingRect
PDTextSelect.GetNumText
PDTextSelect.GetPage
PDTextSelect.GetText
CreateWordHilite

LPDISPATCH CreateWordHilite(LPDISPATCH iAcroHiliteList);

Description
Creates a text selection from a list of word offsets and word counts on a single page. The text selection can then be set as the current selection using AVDoc.SetTextSelection, and the view can be set to show the selection using AVDoc.ShowTextSelect.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iAcroHiliteList</td>
<td>The LPDISPATCH for the highlight list for which a text selection is created. iAcroHiliteList contains the instance variable m_lpDispatch, which contains the LPDISPATCH. Use HiliteList.Add to create a highlight list.</td>
</tr>
</tbody>
</table>

Return Value
The LPDISPATCH for the AcroExch.PDTextSelect, or NULL if the selection could not be created.

Related Methods
AVDoc.ClearSelection
AVDoc.SetTextSelection
AVDoc.ShowTextSelect
HiliteList.Add
PDDoc.CreateTextSelect
PDPage.CreatePageHilite
PDTextSelect.Destroy
PDTextSelect.GetBoundingRect
PDTextSelect.GetNumText
PDTextSelect.GetPage
PDTextSelect.GetText
CropPage

VARIANT_BOOL CropPage(LPDISPATCH iAcroRect);

Description
Crops the page. This method ignores the request if either the width or height of the crop box is less than 72 points (one inch).

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iAcroRect</td>
<td>An LPDISPATCH for a CAcroRect specifying the cropping rectangle, which is specified in user space.</td>
</tr>
</tbody>
</table>

Return Value
-1 is the page was cropped successfully, 0 otherwise.

Related Methods
PDDoc.CropPages
**Draw**

`VARIANT_BOOL Draw(short window, short displayContext, short XOrigin, short YOrigin, short zoom);`

**Description**

**NOTE:** Deprecated: as of Acrobat 3.0, this method simply returns `false`. Use the method `AVDoc.DrawEx` instead.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>window</code></td>
<td>HWND into which the page is to be drawn.</td>
</tr>
<tr>
<td><code>displayContext</code></td>
<td>HDC to use for drawing. If <code>NULL</code>, the HDC for <code>window</code> is used.</td>
</tr>
<tr>
<td><code>XOrigin</code></td>
<td>x-coordinate of the portion of the page to be drawn.</td>
</tr>
<tr>
<td><code>YOrigin</code></td>
<td>y-coordinate of the portion of the page to be drawn.</td>
</tr>
<tr>
<td><code>zoom</code></td>
<td>Zoom factor at which the page is to be drawn, specified as a percent (for example, 100 corresponds to a magnification of 1.0).</td>
</tr>
</tbody>
</table>

**Return Value**

-1 if the page is successfully drawn, 0 otherwise.

**Related Methods**

- `PDPage.CopyToClipboard`
- `PDPage.DrawEx`
**DrawEx**

```cpp
VARIANT_BOOL DrawEx(long window, long displayContext,
                     LPDISPATCH updateRect, short xOrigin,
                     short yOrigin, short zoom);
```

**Description**

Draws page contents into a specified window.

You can use `PDPage.CopyToClipboard` to copy page contents to the clipboard without an `hWnd` or `hDC` from the client.
OLE Automation Methods

AcroExch.PDPage

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>window</td>
<td>Handle for the window (HWND) into which the page is drawn.</td>
</tr>
<tr>
<td>displayContext</td>
<td>This parameter is invalid and should not be used. It should be assigned a NULL value. If it is not assigned a NULL value, an exception will be thrown. Note: displayContext cannot be reliably used as the hdc for a printer device. In particular, Visual Basic applications cannot use DrawEx to print.</td>
</tr>
<tr>
<td>updateRect</td>
<td>LPDISPATCH for an AcroExch.Rect to be drawn with user space coordinates. updateRect contains the instance variable m_lpDispatch, which contains the LPDISPATCH. Any objects outside of updateRect are not drawn. All objects are drawn if updateRect is NULL. Note: In previous specifications, this rectangle was in device space coordinates. Use methods in the CACroRect class to set the size of the rectangle. For example: CACroRect* rect = new CACroRect; rect-&gt;CreateDispatch(&quot;AcroExch.Rect&quot;, &amp;e); if (rect) { /* Set values for rect - increases from right to left and bottom to top */ rect-&gt;SetLeft(100); rect-&gt;SetTop(400); rect-&gt;SetRight(400); rect-&gt;SetBottom(100); }</td>
</tr>
<tr>
<td>xOrigin</td>
<td>x-coordinate of the portion of the page to be drawn.</td>
</tr>
<tr>
<td>yOrigin</td>
<td>y-coordinate of the portion of the page to be drawn.</td>
</tr>
<tr>
<td>zoom</td>
<td>Zoom factor at which the page is drawn, specified as a percent (for example, 100 corresponds to a magnification of 1.0).</td>
</tr>
</tbody>
</table>

Return Value

A positive number if the page is successfully drawn, 0 otherwise.

Related Methods

PDPage.CopyToClipboard
GetAnnot

LPDISPATCH GetAnnot(long nIndex);

Description

Gets the specified annotation from the page's array of annotations.

Parameters

| nIndex | Index (in the page's annotation array) of the annotation to be retrieved. The first annotation in the array has an index of zero. |

Return Value

The LPDISPATCH for the AcroExch.PDAnnot object.

Related Methods

- PDPage.GetAnnotIndex
- PDPage.GetNumAnnots
GetAnnotIndex

long GetAnnotIndex(LPDISPATCH iPDAnnot);

Description

Gets the index (within the page’s annotation array) of the specified annotation.

Parameters

| iPDAnnot | LPDISPATCH for the AcroExch.PDAnnot whose index is obtained. iPDAnnot contains the instance variable m_lpDispatch, which contains the LPDISPATCH. |

Return Value

The annotation’s index.

Related Methods

PDPage.GetAnnot
PDPage.GetNumAnnots
GetDoc

LPDISPATCH GetDoc();

Description

Gets the AcroExch.PDDoc associated with the page.

Parameters

None

Return Value

The LPDISPATCH for the page's AcroExch.PDDoc.

Related Methods

AVPageView.GetPage
AVPageView.GetPageNum
PDDoc.AcquirePage
PDDoc.GetNumPages
PDPage.GetNumber
PDPage.GetRotate
PDPage.GetSize
PDTextSelect.GetPage
GetNumAnnots

```c
long GetNumAnnots();
```

**Description**
Gets the number of annotations on the page.

**NOTE:** Annotations that have associated pop-up windows (e.g. strikeouts) are counted as two annotations. Also note that widget annotations (Acrobat form fields) are included.

**Parameters**
None

**Return Value**
The number of annotations on the page.

**Related Methods**
PDPAGE.GetAnnot
PDPAGE.GetAnnotIndex
GetNumber

```cpp
long GetNumber();
```

**Description**

Gets the page number of the current page. The first page in a document is page zero.

**Parameters**

None

**Return Value**

The page number of the current page. The first page in a `PDDoc` object is page 0.

**Related Methods**

- `AVPageView.GetPage`
- `AVPageView.GetPageNum`
- `PDDoc.AcquirePage`
- `PDDoc.GetNumPages`
- `PDPage.GetDoc`
- `PDPage.GetRotate`
- `PDPage.GetSize`
- `PDTextSelect.GetPage`
GetRotate

short GetRotate();

Description
Gets the rotation value, in degrees, for the current page.

Parameters

Return Value
Rotation value. See rotation in page (located in Chapter 1, “Apple Event Objects”) for a list of rotation values.

Related Methods
AVPageView.GetPage
AVPageView.GetPageNum
PDDoc.AcquirePage
PDPage.GetNumber
PDPage.GetSize
PDPage.SetRotate
PDTextSelect.GetPage
GetSize

LPDISPATCH GetSize();

Description

Gets a page's width and height in points.

Parameters

None

Return Value

The LPDISPATCH for an AcroExch.Point containing the width and height, measured in points. Point x contains the width, point y the height.

Related Methods

AVPageView.GetPage
AVPageView.GetPageNum
PDDoc.AcquirePage
PDPage.GetNumber
PDPage.GetRotate
PDTextSelect.GetPage
RemoveAnnot

    VARIANT_BOOL RemoveAnnot(long nIndex);

**Description**

Removes the specified annotation from the page's annotation array.

**Parameters**

| nIndex | Index within the page's annotation array of the annotation to be deleted. The first annotation on a page has an index of zero. |

**Return Value**

0 if the Acrobat application does not support editing, a positive number otherwise.

**Related Methods**

PDPage.AddAnnot
PDPage.AddNewAnnot
PDPage.GetAnnotIndex
SetRotate

VARIANT_BOOL SetRotate(short nRotate);

Description
Sets the rotation, in degrees, for the current page.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nRotate</td>
<td>Rotation value. See rotation in page (in Chapter 1, “Apple Event Objects”) for a list of rotation values.</td>
</tr>
</tbody>
</table>

Return Value

0 if the Acrobat application does not support editing, -1 otherwise.

Related Methods

PDPage.GetRotate
**AcroExch.PDTextSelect**

The methods in this section work with text in a document.

---

**Destroy**

```c
VARIANT_BOOL Destroy();
```

**Description**

Destroys a text selection object.

**Parameters**

None

**Return Value**

Always returns -1.

**Related Methods**

- AVDoc.ClearSelection
- AVDoc.SetTextSelection
- AVDoc.ShowTextSelect
- PDDoc.CreateTextSelect
- PDPAGE.CreatePageHilite
- PDPAGE.CreateWordHilite
- PDTextSelect.GetBoundingRect
- PDTextSelect.GetNumText
- PDTextSelect.GetPage
- PDTextSelect.GetText
GetBoundingRect

LPDISPATCH GetBoundingRect();

Description

Gets a text selection's bounding rectangle.

Parameters

None

Return Value

The LPDISPATCH for an AcroExch.Rect corresponding to the text selection's bounding rectangle

Related Methods

AVDoc.ClearSelection
AVDoc.SetTextSelection
AVDoc.ShowTextSelect
PDDoc.CreateTextSelect
PDPage.CreatePageHilite
PDPage.CreateWordHilite
PDTextSelect_Destroy
PDTextSelect_GetNumText
PDTextSelect_GetPage
PDTextSelect_GetText
GetNumText

long GetNumText();

Description

Gets the number of text elements in a text selection. Use this method to determine how many times to call the PDTextSelect.GetText method to obtain all of a text selection's text.

NOTE: A text element is not necessarily a word. A text element consists of characters of the same font, size and style; therefore, there may be more than one text element in a word.

Parameters

None

Return Value

The number of elements in the text selection.

Related Methods

AVDoc.ClearSelection
AVDoc.SetTextSelection
AVDoc.ShowTextSelect
PDDoc.CreateTextSelect
PDPage.CreatePageHilite
PDPage.CreateWordHilite
PDTextSelect.Destroy
PDTextSelect.GetBoundingRect
PDTextSelect.GetPage
PDTextSelect.GetText
GetPage

long GetPage();

Description
Gets the page number on which the text selection is located.

Parameters
None

Return Value
The text selection's page number. The first page in a PDDoc object is page 0.

Related Methods
AVDoc.ClearSelection
AVDoc.SetTextSelection
AVDoc.ShowTextSelect
AVPageView.GetPage
AVPageView.GetPageNum
PDDoc.CreateTextSelect
PDDoc.GetNumPages
PDPage.CreatePageHilite
PDPage.CreateWordHilite
PDPage.GetNumber
PDTextSelect.Destroy
PDTextSelect.GetBoundingRect
PDTextSelect.GetNumText
PDTextSelect.GetText
GetText

BSTR GetText(long nTextIndex);

Description
Gets the text from the specified element of a text selection. To obtain all the text within the text selection, use PDTextSelect.GetNumText to determine the number of elements in the text selection, then call this method in a loop to obtain each of the elements.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nTextIndex</td>
<td>The element of the text selection to get.</td>
</tr>
</tbody>
</table>

Return Value
The text, or an empty string if nTextIndex is greater than the number of elements in the text selection.

Related Methods
AVDoc.ClearSelection
AVDoc.SetTextSelection
AVDoc.ShowTextSelect
PDPage.CreatePageHilite
PDDoc.CreateTextSelect
PDPage.CreateWordHilite
PDTextSelect.Destroy
PDTextSelect.GetBoundingRect
PDTextSelect.GetNumText
PDTextSelect.GetPage
**AxAcroPDFLib.AxAcroPDF**

The methods in this section work with PDF browser controls.

---

**GetVersions**

```c
VARIANT GetVersions();
```

**NOTE:** Deprecated: No longer available - do not use.
GoBackwardStack

void GoBackwardStack();

Description
Goes to the previous view on the view stack, if the previous view exists. The previous view may be in a different document.

Parameters
None

Return Value
None.

Related Methods
AcroPDF.GoForwardStack
GoForwardStack

    void GoForwardStack();

Description

Goes to the next view on the view stack, if the next view exists. The next view may be in a different document.

Parameters

None

Return Value

None.

Related Methods

AcroPDF.GoBackwardStack
**GotoFirstPage**

```csharp
gotoFirstPage();
```

**Description**
Goes to the first page in the document, maintaining the current location within the page and zoom level.

**Parameters**
None

**Return Value**
None.

**Related Methods**
- `AcroPDF.GotoLastPage`
- `AcroPDF.GotoNextPage`
- `AcroPDF.GotoPreviousPage`
- `AcroPDF.SetCurrentPage`
**GotoLastPage**

    void gotoLastPage();

**Description**

Goes to the last page in the document, maintaining the current location within the page and zoom level.

**Parameters**

| None |

**Return Value**

None.

**Related Methods**

- `AcroPDF.GotoFirstPage`
- `AcroPDF.GotoNextPage`
- `AcroPDF.GotoPreviousPage`
- `AcroPDF.SetCurrentPage`
**GotoNextPage**

```c
void gotoNextPage();
```

**Description**

Goes to the next page in the document, if it exists. Maintains the current location within the page and zoom level.

**Parameters**

None

**Return Value**

None.

**Related Methods**

- `AcroPDF.GotoFirstPage`
- `AcroPDF.GotoLastPage`
- `AcroPDF.GotoPreviousPage`
- `AcroPDF.SetCurrentPage`
GotoPreviousPage

    void gotoPreviousPage();

Description

Goes to the previous page in the document, if it exists. Maintains the current location within the page and zoom level.

Parameters

None

Return Value

None.

Related Methods

    AcroPDF.GotoFirstPage
    AcroPDF.GotoLastPage
    AcroPDF.GotoNextPage
    AcroPDF.SetCurrentPage
LoadFile

VARIANT_BOOL LoadFile(BSTR fileName);

Description
Opens and displays the specified document within the browser.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>The pathname of the file to be opened.</td>
</tr>
</tbody>
</table>

Return Value

0 if the file could not be opened, -1 otherwise.

Related Methods
None
Print

void Print();

Description

Prints the document according to the options selected in a user dialog box. The options include embedded printing (printing within a bounding rectangle on a given page), as well as interactive printing to a specified printer. This method returns immediately, even if the printing has not completed.

**Note:** If security settings do not allow printing, this method will be ignored.

Parameters

None

Return Value

None.

Related Methods

AcroPDF.PrintAll
AcroPDF.PrintAllFit
AcroPDF.PrintPages
AcroPDF.PrintPagesFit
AcroPDF.PrintWithDialog
PrintAll

    void printAll();

Description

Prints the entire document without displaying a user dialog box. The current printer, page settings, and job settings are used. This method returns immediately, even if the printing has not completed.

**NOTE:** If security settings do not allow printing, this method will be ignored.

Parameters

None

Return Value

None.

Related Methods

* AcroPDF.Print
* AcroPDF.PrintAllFit
* AcroPDF.PrintPages
* AcroPDF.PrintPagesFit
* AcroPDF.PrintWithDialog
PrintAllFit

```csharp
void printAllFit(VARIANT_BOOL bOn);
```

**Description**

Prints the entire document without displaying a user dialog box, and the pages are shrunk, if necessary, to fit into the imageable area of a page in the printer. The current printer, page settings, and job settings are used. This method returns immediately, even if the printing has not completed.

**NOTE:** If security settings do not allow printing, this method will be ignored.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bOn</td>
<td>Determines whether to scale the imageable area when printing the document. A value of 0 indicates that no scaling should be used, and a positive value indicates that the pages are shrunk, if necessary, to fit into the imageable area of a page in the printer.</td>
</tr>
</tbody>
</table>

**Return Value**

None.

**Related Methods**

- `AcroPDF.Print`
- `AcroPDF.PrintAll`
- `AcroPDF.PrintPages`
- `AcroPDF.PrintPagesFit`
- `AcroPDF.PrintWithDialog`
PrintPages

void printPages( Long nFrom, Long nTo);

Description

Prints the specified pages without displaying a user dialog box. The current printer, page settings, and job settings are used. This method returns immediately, even if the printing has not completed.

NOTE: If security settings do not allow printing, this method will be ignored.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nFrom</td>
<td>The page number of the first page to be printed. The first page in a document is page 0.</td>
</tr>
<tr>
<td>nTo</td>
<td>The page number of the last page to be printed.</td>
</tr>
</tbody>
</table>

Return Value

None.

Related Methods

- AcroPDF.Print
- AcroPDF.PrintAll
- AcroPDF.PrintAllFit
- AcroPDF.PrintPagesFit
- AcroPDF.PrintWithDialog
PrintPagesFit

void printPagesFit( Long nFrom, Long nTo,
                    VARIANT_BOOL bShrinkToFit);

Description
Prints the specified pages without displaying a user dialog box. The current printer, page
settings, and job settings are used. This method returns immediately, even if the printing
has not completed.

**NOTE:** If security settings do not allow printing, this method will be ignored.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nFrom</td>
<td>The page number of the first page to be printed. The first page in a document is page 0.</td>
</tr>
<tr>
<td>nTo</td>
<td>The page number of the last page to be printed.</td>
</tr>
<tr>
<td>bShrinkToFit</td>
<td>Specifies whether the pages will be shrunk, if necessary, to fit into the imageable area of a page in the printer.</td>
</tr>
</tbody>
</table>

Return Value
None.

Related Methods
- AcroPDF.Print
- AcroPDF.PrintAll
- AcroPDF.PrintAllFit
- AcroPDF.PrintPages
- AcroPDF.PrintWithDialog
PrintWithDialog

    void printWithDialog();

Description

Prints the document according to the options selected in a user dialog box. The options include embedded printing (printing within a bounding rectangle on a given page), as well as interactive printing to a specified printer. This method returns immediately, even if the printing has not completed.

**NOTE:** If security settings do not allow printing, this method will be ignored.

Parameters

None

Return Value

None.

Related Methods

AcroPDF.Print
AcroPDF.PrintAll
AcroPDF.PrintAllFit
AcroPDF.PrintPages
AcroPDF.PrintPagesFit
SetCurrentHighlight

```c
void setCurrentHighlight(LONG nLeft, LONG nTop,
                         LONG nRight, LONG nBottom);
```

**Description**

Highlights the text selection within the specified bounding rectangle on the current page.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nLeft</td>
<td>The distance in points from the left side of the page.</td>
</tr>
<tr>
<td>nTop</td>
<td>The distance in points from the top of the page.</td>
</tr>
<tr>
<td>nRight</td>
<td>The width of the bounding rectangle.</td>
</tr>
<tr>
<td>nBottom</td>
<td>The height of the bounding rectangle.</td>
</tr>
</tbody>
</table>

**Return Value**

None.

**Related Methods**

None
SetCurrentPage

```c
void setCurrentPage(LONG nPage);
```

**Description**

Goes to the specified page in the document. Maintains the current location within the page and zoom level.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nPage</td>
<td>The page number of the destination page. The first page in a document is page 0.</td>
</tr>
</tbody>
</table>

**Return Value**

None.

**Related Methods**

- `AcroPDF.GotoFirstPage`
- `AcroPDF.GotoLastPage`
- `AcroPDF.GotoNextPage`
- `AcroPDF.GotoPreviousPage`
SetLayoutMode

void setLayoutMode(BSTR szLayoutMode);

Description
Sets the layout mode for a page view according to the specified string.

Parameters

<table>
<thead>
<tr>
<th>szLayoutMode</th>
<th>Possible values:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;DontCare&quot;: use the current user preference</td>
</tr>
<tr>
<td></td>
<td>&quot;SinglePage&quot;: use single page mode (as it would have</td>
</tr>
<tr>
<td></td>
<td>appeared in pre-Acrobat 3.0 viewers)</td>
</tr>
<tr>
<td></td>
<td>&quot;OneColumn&quot;: use one-column continuous mode</td>
</tr>
<tr>
<td></td>
<td>&quot;TwoColumnLeft&quot;: use two-column continuous mode with</td>
</tr>
<tr>
<td></td>
<td>the first page on the left</td>
</tr>
<tr>
<td></td>
<td>&quot;TwoColumnRight&quot;: use two-column continuous mode with</td>
</tr>
<tr>
<td></td>
<td>the first page on the right</td>
</tr>
</tbody>
</table>

Return Value
None.

Related Methods

AcroPDF.SetNamedDest
AcroPDF.SetView
AcroPDF.SetViewRect
AcroPDF.SetViewScroll
SetNamedDest

void SetNamedDest(BSTR szNamedDest);

Description
Changes the page view to the named destination in the specified string.

Parameters

| szNamedDest | The named destination to which the viewer will go. |

Return Value
None.

Related Methods
AcroPDF.SetLayoutMode
AcroPDF.SetView
AcroPDF.SetViewRect
AcroPDF.SetViewScroll
**SetPageMode**

```c
void setPageMode(BSTR szPageMode);
```

**Description**

Sets the page mode according to the specified string.

**Parameters**

<table>
<thead>
<tr>
<th>szPageMode</th>
<th>Possible values:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;none&quot;: displays the document, but does not display bookmarks or thumbnails (default)</td>
</tr>
<tr>
<td></td>
<td>&quot;bookmarks&quot;: displays the document and bookmarks</td>
</tr>
<tr>
<td></td>
<td>&quot;thumbs&quot;: displays the document and thumbnails</td>
</tr>
</tbody>
</table>

**Return Value**

None.

**Related Methods**

- `AcroPDF::SetShowScrollbars`
- `AcroPDF::SetShowToolbar`
**SetShowScrollbars**

```cpp
void setShowScrollbars(VARIANT_BOOL bOn);
```

**Description**

Determines whether scrollbars will appear in the document view.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bOn</td>
<td>A positive value indicates that scrollbars will appear, 0 indicates that they will not.</td>
</tr>
</tbody>
</table>

**Return Value**

None.

**Related Methods**

- `AcroPDF.SetPageMode`
- `AcroPDF.SetShowToolbar`
### SetShowToolbar

```c
void setShowToolbar(VARIANT_BOOL bOn);
```

**Description**

Determines whether a toolbar will appear in the viewer.

**Parameters**

<table>
<thead>
<tr>
<th>bOn</th>
<th>A positive value indicates that the toolbar will appear, 0 indicates that it will not.</th>
</tr>
</thead>
</table>

**Return Value**

None.

**Related Methods**

- `AcroPDF.SetPageMode`
- `AcroPDF.SetShowScrollbars`
SetView

```csharp
void setView(BSTR szViewMode);
```

**Description**
Sets the view of a page according to the specified string.

**Parameters**

<table>
<thead>
<tr>
<th>szViewMode</th>
<th>Possible values:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;Fit&quot;: fits the entire page within the window both vertically and horizontally.</td>
</tr>
<tr>
<td></td>
<td>&quot;FitH&quot;: fits the entire width of the page within the window.</td>
</tr>
<tr>
<td></td>
<td>&quot;FitV&quot;: fits the entire height of the page within the window.</td>
</tr>
<tr>
<td></td>
<td>&quot;FitB&quot;: fits the bounding box within the window both vertically and horizontally.</td>
</tr>
<tr>
<td></td>
<td>&quot;FitBH&quot;: fits the entire width of the bounding box within the window.</td>
</tr>
<tr>
<td></td>
<td>&quot;FitBV&quot;: fits the entire height of the bounding box within the window.</td>
</tr>
</tbody>
</table>

**Return Value**
None.

**Related Methods**

- `AcroPDF.SetLayoutMode`
- `AcroPDF.SetNamedDest`
- `AcroPDF.SetViewRect`
- `AcroPDF.SetViewScroll`
SetViewRect

    void setViewRect(FLOAT left, FLOAT top,
                    FLOAT width, FLOAT height);

Description

Sets the view rectangle according to the specified coordinates.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>left</td>
<td>The upper left horizontal coordinate.</td>
</tr>
<tr>
<td>top</td>
<td>The vertical coordinate in the upper left corner.</td>
</tr>
<tr>
<td>width</td>
<td>The horizontal width of the rectangle.</td>
</tr>
<tr>
<td>height</td>
<td>The vertical height of the rectangle.</td>
</tr>
</tbody>
</table>

Return Value

None.

Related Methods

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcroPDF.SetLayoutMode</td>
</tr>
<tr>
<td>AcroPDF.SetNamedDest</td>
</tr>
<tr>
<td>AcroPDF.SetView</td>
</tr>
<tr>
<td>AcroPDF.SetViewScroll</td>
</tr>
</tbody>
</table>
SetViewScroll

void setViewRect(BSTR szViewMode, FLOAT offset);

Description
Sets the view of a page according to the specified string. Depending on the view mode, the page is either scrolled to the right or scrolled down by the amount specified in offset.

Parameters

<table>
<thead>
<tr>
<th>szViewMode</th>
<th>Possible values:</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Fit&quot;</td>
<td>fits the entire page within the window both vertically and horizontally.</td>
</tr>
<tr>
<td>&quot;FitH&quot;</td>
<td>fits the entire width of the page within the window.</td>
</tr>
<tr>
<td>&quot;FitV&quot;</td>
<td>fits the entire height of the page within the window.</td>
</tr>
<tr>
<td>&quot;FitB&quot;</td>
<td>fits the bounding box within the window both vertically and horizontally.</td>
</tr>
<tr>
<td>&quot;FitBH&quot;</td>
<td>fits the entire width of the bounding box within the window.</td>
</tr>
<tr>
<td>&quot;FitBV&quot;</td>
<td>fits the entire height of the bounding box within the window.</td>
</tr>
</tbody>
</table>

| offset | The horizontal or vertical coordinate positioned either at the left or top edge. |

Return Value
None.

Related Methods

AcroPDF.SetLayoutMode
AcroPDF.SetNamedDest
AcroPDF.SetView
AcroPDF.SetViewRect
SetZoom

```c
void setZoom(FLOAT percent);
```

**Description**
Sets the magnification according to the specified value.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>percent</td>
<td>The desired zoom factor, expressed as a percentage (for example, 1.0 represents a magnification of 100%).</td>
</tr>
</tbody>
</table>

**Return Value**
None.

**Related Methods**

`AcroPDF.SetZoomScroll`
SetZoomScroll

void setZoomScroll(FLOAT percent, FLOAT left, FLOAT top);

Description
Sets the magnification according to the specified value, and scrolls the page view both horizontally and vertically according to the specified amounts.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>percent</td>
<td>The desired zoom factor, expressed as a percentage (for example, 1.0 represents a magnification of 100%).</td>
</tr>
<tr>
<td>left</td>
<td>The horizontal coordinate positioned at the left edge.</td>
</tr>
<tr>
<td>top</td>
<td>The vertical coordinate positioned at the top edge.</td>
</tr>
</tbody>
</table>

Return Value
None.

Related Methods
AcroPDF.SetZoom
AcroExch.Point

X

[get/set] Short

Description
Gets or sets the x-coordinate of an AcroPoint.

Return Value
The x-coordinate of the AcroPoint.
Y

[get/set] Short

Description

Gets or sets the y-coordinate of an AcroPoint.

Return Value

The y-coordinate of the AcroPoint.
**AcroExch.Rect**

**Bottom**

[get/set] Short

**Description**

Gets or sets the bottom y-coordinate of an AcroRect.

**Return Value**

The y-coordinate of the bottom of the AcroRect.
Left

[get/set] Short

Description

Gets or sets left x-coordinate of an AcroRect.

Return Value

The x-coordinate of the left side of the AcroRect.
Right

[get/set] Short

Description

Gets or sets the right x-coordinate of an AcroRect.

Return Value

The x-coordinate of the right side of the AcroRect.
**Top**

[get/set] Short

**Description**

Gets or sets the top y-coordinate of an *AcroRect*.

**Return Value**

The y-coordinate of the top of the *AcroRect*.
**AcroExch.Time**

**Date**

[get/set] Short

**Description**

Gets or sets the date from an `AcroTime`.

**Return Value**

The date from the `AcroTime`. The date runs from 1 to 31.
**Hour**

[get/set] Short

**Description**

Gets or sets the hour from an AcroTime.

**Return Value**

The hour from the AcroTime. The hour runs from 0 to 23.
Millisecond

[get/set] Short

**Description**

Gets or sets the milliseconds from an AcroTime.

**Return Value**

The milliseconds from the AcroTime. Milliseconds run from 0 to 999.
**Minute**

[get/set] Short

**Description**

Gets or sets the minutes from an `AcroTime`.

**Return Value**

The minutes from the `AcroTime`. Minutes run from 0 to 59.
Month

[get/set] Short

Description

Gets or sets the month from an AcroTime.

Return Value

The month from the AcroTime. The month runs from 1 to 12, where 1 is January, ..., and 12 is December.
**Second**

[get/set] Short

**Description**

Gets or sets the seconds from an `AcroTime`.

**Return Value**

The seconds from the `AcroTime`. Seconds run from 0 to 59.
Year

[get/set] Short

Description

Gets or sets the year from an AcroTime.

Return Value

The year from the AcroTime. The Year runs from 1 to 32767.
AxAcroPDFLib.AxAcroPDF

Src

[get/set] src

Description

Gets or sets the URL for the document.

Return Value

The URL for the document, formatted as a string.
This reference contains the following section:

- **DDE Messages.** Description of each message, its arguments, return values, and related methods.

In the DDE message descriptions, the square bracket characters [ and ] in DDE messages are significant, and must be included as part of the message.
AppExit

[AppExit()]

Description

Exits the Acrobat application.

AppExit is also supported in Adobe Reader.

Parameters

None

Return Value

true if the Acrobat application exited successfully, false otherwise.

Related Methods

AppHide
AppShow
**AppHide**

[AppHide()]

**Description**
Iconifies or hides the Acrobat application.

**Parameters**
None

**Return Value**
true if the Acrobat application was hidden successfully, false otherwise.

**Related Methods**
- AppExit
- AppShow
AppShow

[AppShow()]

Description
Shows the Acrobat application.

Parameters
None

Return Value
true if the Acrobat application was shown successfully, false otherwise.

Related Methods
AppExit
AppHide
**CloseAllDocs**

**Description**

Closes all open documents.

*CloseAllDocs* is also supported in Adobe Reader.

**Parameters**

None

**Return Value**

*true* if the documents were closed successfully, *false* otherwise.

**Related Methods**

- *DocClose*
- *DocOpen*
- *FileOpen*
**DocClose**

[DocClose(char* fullPath)]

**Description**

Closes the specified document without saving it, and without prompting the user to save the document if it has been modified.

**Parameters**

| fullPath | The full pathname of the file to be closed. |

**Return Value**

true if the document was closed successfully, false if the document does not exist or was not closed successfully.

**Related Methods**

CloseAllDocs
DocOpen
FileOpen
DocDeletePages

[DocDeletePages(char* fullPath, long fromPage, long toPage)]

Description

Deletes the specified pages in the document. Requests to delete all pages in a document are ignored, since a document must have at least one page.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full pathname of the document.</td>
</tr>
<tr>
<td>fromPage</td>
<td>The page number of the first page to be deleted.</td>
</tr>
<tr>
<td>toPage</td>
<td>The page number of the last page to be deleted.</td>
</tr>
</tbody>
</table>

Return Value

true if the pages were deleted successfully. Returns false if the document specified by fullPath does not exist, if the request was to delete all the document’s pages, or if the pages were not deleted successfully.

Related Methods

DocInsertPages
DocReplacePages
DocFind

[DocFind(char* fullPath, char* string, boolean caseSensitive, boolean wholeWords, boolean bReset)]

Description

Finds a string in a specified file. This does not use the cross-document search present in versions 2.0 and later of Acrobat, but performs a page-by-page search of the specified file.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full pathname of the file to be searched.</td>
</tr>
<tr>
<td>string</td>
<td>The string to be found.</td>
</tr>
<tr>
<td>caseSensitive</td>
<td>true if the search is case-sensitive, false otherwise.</td>
</tr>
<tr>
<td>wholeWords</td>
<td>true if the search will only match whole words, false otherwise.</td>
</tr>
<tr>
<td>bReset</td>
<td>true if the search begins on the first page of the document, false otherwise.</td>
</tr>
</tbody>
</table>

Return Value

false if the document specified by fullPath does not exist or if the text is not found, true otherwise.
**DocGoTo**

```
[DocGoTo(char* fullPath, long pageNum)]
```

**Description**

Goes to the specified page.

**DocGoTo** is also supported in Adobe Reader.

**Parameters**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full pathname of the file.</td>
</tr>
<tr>
<td>pageNum</td>
<td>The page number of the destination page.</td>
</tr>
</tbody>
</table>

**Return Value**

`false` if the document specified by `fullPath` does not exist, `true` otherwise.
**DocGoToNameDest**

```c
[DocGoToNameDest(char* fullPath, char* nameDest)]
```

**Description**

Goes to the specified named destination.

*DocGoToNameDest* is also supported in Adobe Reader.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fullPath</code></td>
<td>The full pathname of the file.</td>
</tr>
<tr>
<td><code>nameDest</code></td>
<td>The named destination.</td>
</tr>
</tbody>
</table>

**Return Value**

- `false` if the document specified by `fullPath` does not exist, `true` otherwise.
**DocInsertPages**

```c
[DocInsertPages(char* fullPath, long insertAfterPage, char* sourcePath)]
```

**Description**

Inserts pages from one file into another.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>fullPath</strong></td>
<td>The full pathname of the target document, which must already be open in the Acrobat application.</td>
</tr>
</tbody>
</table>
| **insertAfterPage** | The page number after which pages are being inserted. Possible values may be a page number, or one of the following enumerations:  
  - **PDBeforeFirstPage** — Pages are inserted at the beginning of the document.  
  - **PDLastPage** — Pages are inserted at the end of the document. |
| **sourcePath** | The full pathname of the source document. This file need not be open in the Acrobat application. |

**Return Value**

*true* if the pages were inserted successfully, *false* if the document does not exist or the pages were not inserted successfully.

**Related Methods**

- DocDeletePages
- DocReplacePages
DocOpen

[DocOpen(char* fullPath)]

Description
Opens a document and adds it to the list of documents known to DDE, allowing it to be manipulated by other DDE messages (see FileOpen).
DocOpen is also supported in Adobe Reader.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full pathname of the file to be opened.</td>
</tr>
</tbody>
</table>

Return Value
true if the file was opened successfully, false otherwise.

Related Methods
CloseAllDocs
DocClose
FileOpen
**DocPageDown**

```
[DocPageDown(char* fullPath)]
```

**Description**

Scrolls forward through the document by one screen area.

**Parameters**

<table>
<thead>
<tr>
<th>fullPath</th>
<th>The full pathname of the document.</th>
</tr>
</thead>
</table>

**Return Value**

*false* if the document specified by *fullPath* does not exist, *true* otherwise.

**Related Methods**

- DocPageLeft
- DocPageRight
- DocPageUp
- DocScrollTo
**DocPageLeft**

[DocPageLeft(char* fullPath)]

**Description**

Scrolls to the left by a small amount.

**Parameters**

| fullPath | The full pathname of the document. |

**Return Value**

*false* if the document specified by *fullPath* does not exist, *true* otherwise.

**Related Methods**

- [DocPageDown](#)
- [DocPageRight](#)
- [DocPageUp](#)
- [DocScrollTo](#)
**DocPageRight**

[DocPageRight(char* fullPath)]

**Description**

Scrolls to the right by a small amount.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full pathname of the document.</td>
</tr>
</tbody>
</table>

**Return Value**

*false* if the document specified by *fullPath* does not exist, *true* otherwise.

**Related Methods**

- DocPageDown
- DocPageLeft
- DocPageUp
- DocScrollTo
**DocPageUp**

```
[DocPageUp(char* fullPath)]
```

**Description**

Scrolls backward through the document by one screen area.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fullPath</code></td>
<td>The full pathname of the document.</td>
</tr>
</tbody>
</table>

**Return Value**

`false` if the document specified by `fullPath` does not exist, `true` otherwise.

**Related Methods**

- `DocPageDown`
- `DocPageLeft`
- `DocPageRight`
- `DocScrollTo`
DocPrint

[DocPrint(char* fullPath, long startPage, long endPage)]

Description
Prints a specified range of pages from a document, without displaying any modal Print
dialog box to the user. For PostScript printing, only Level 1 operators are used, only ASCII
data is generated, and the document's pages are not shrunk to fit into the imageable area
of the printed page.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full pathname of document.</td>
</tr>
<tr>
<td>startPage</td>
<td>The page number of the first page to be printed.</td>
</tr>
<tr>
<td>endPage</td>
<td>The page number of the last page to be printed.</td>
</tr>
</tbody>
</table>

Return Value
false if the document specified by fullPath does not exist, true otherwise.

Related Methods
FilePrint
FilePrintSilent
FilePrintTo
**DocReplacePages**

```c
[DocReplacePages(char* fullPath, long startDestPage,
                 char* sourcePath, long startSourcePage,
                 long endSourcePage)]
```

**Description**

Replaces pages in the target document using the specified pages from the source document.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>fullPath</strong></td>
<td>The full pathname of the target document. This file must already be open in the Acrobat application.</td>
</tr>
<tr>
<td><strong>startDestPage</strong></td>
<td>The page number of the first page in the target document to be replaced.</td>
</tr>
<tr>
<td><strong>sourcePath</strong></td>
<td>The full pathname of the source document. This file does not have to be already open in the Acrobat application.</td>
</tr>
<tr>
<td><strong>startSourcePage</strong></td>
<td>The page number of the first page in the source document to use as a replacement page.</td>
</tr>
<tr>
<td><strong>endSourcePage</strong></td>
<td>The page number of the last page in the source document to use as a replacement page.</td>
</tr>
</tbody>
</table>

**Return Value**

*true* if the pages were replaced successfully. Returns *false* if the document does not exist or the pages were not replaced successfully.

**Related Methods**

- [DocDeletePages](#)
- [DocInsertPages](#)
DocSave

[DocSave(char* fullPath)]

Description
Saves the specified file. The user is not warned if there are any problems saving the file.

Parameters

| fullPath       | The full pathname of the file to be saved. |

Return Value
true if the document was saved successfully, false if the document does not exist or was not saved successfully.

Related Methods
DocSaveAs
**DocSaveAs**

[DocSaveAs(char* fullPath, char* newPath)]

**Description**

Saves an open file to a new pathname. The user is not warned if there are any problems saving the file.

**Parameters**

<table>
<thead>
<tr>
<th>fullPath</th>
<th>The full pathname of the existing file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>newPath</td>
<td>The full pathname of the new file.</td>
</tr>
</tbody>
</table>

**Return Value**

true if the document was saved successfully, false if the document does not exist or was not saved successfully.

**Related Methods**

DocSave
DocScrollTo

[DocScrollTo(char* fullPath, int x, int y)]

Description

Scrolls the view of the current page to the specified location.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full pathname of the document.</td>
</tr>
<tr>
<td>x</td>
<td>The destination's x-coordinate.</td>
</tr>
<tr>
<td>y</td>
<td>The destination's y-coordinate.</td>
</tr>
</tbody>
</table>

Return Value

false if the document specified by fullPath does not exist, true otherwise.

Related Methods

DocPageDown
DocPageLeft
DocPageRight
DocPageUp
**DocSetViewMode**

[DocSetViewMode(char* fullPath, char* viewType)]

**Description**
Determines whether bookmarks, thumbnail images, or neither are shown in addition to the document.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>fullPath</strong></td>
<td>The full pathname of the document.</td>
</tr>
<tr>
<td><strong>viewType</strong></td>
<td>The view mode to be used. Must be one of the following:</td>
</tr>
<tr>
<td></td>
<td><strong>PDUseThumbs</strong> — Displays pages and thumbnail images.</td>
</tr>
<tr>
<td></td>
<td><strong>PDUseNone</strong> — Displays only pages.</td>
</tr>
<tr>
<td></td>
<td><strong>PDUseBookmarks</strong> — Displays pages and bookmarks.</td>
</tr>
</tbody>
</table>

**Return Value**

*true* if the view mode was set successfully, *false* if the document specified by **fullPath** does not exist or an unknown view mode is specified.

**Related Methods**

- FullMenus
- ShortMenus
DocZoomTo

[DocZoomTo(char* fullPath, char* zoomType, int scale)]

Description
Sets the zoom for a specified document.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full pathname of the file whose zoom to set.</td>
</tr>
</tbody>
</table>
| zoomType   | The zoom strategy to use. Must be one of the following:  
  AVZoomNoVary — A fixed zoom, such as 100%.  
  AVZoomFitPage — Fits the page in the window.  
  AVZoomFitWidth — Fits the page's width into the window.  
  AVZoomFitVisibleWidth — Fits the page's visible content into the window.  |
| scale      | The magnification specified as a percent (for example, 100 corresponds to a magnification of 1.0). scale is used only when zoomType is AVZoomNoVary. |

Return Value

false if the document specified by fullPath does not exist, or if zoomType has an unknown value. Returns true otherwise.
**FileOpen**

```c
[FileOpen(char* fullPath)]
```

**Description**

Opens and displays the specified document. If the file is already open, it becomes the active document and appears in the front. This DDE message does not add the document to the list that can be manipulated using DDE messages; use `DocOpen` to do that.

`FileOpen` is also supported in Adobe Reader.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fullPath</code></td>
<td>The full pathname of the file to be opened.</td>
</tr>
</tbody>
</table>

**Return Value**

`true` if the file was opened successfully, `false` otherwise.

**Related Methods**

- `CloseAllDocs`
- `DocClose`
- `DocOpen`
**FileOpenEx**

[FileOpenEx(char* fullPath)]

**Description**

Opens and displays a file. If the file is already open, it becomes the active document and appears in the front. This DDE message does not add the document to the list that can be manipulated using DDE messages; use **DocOpen** to do that.

This method allows documents that either take a long time to open or are password-protected to open without stopping the flow of DDE messages. Documents opened with **FileOpenEx** are opened during an idle period. This is useful in situations in which several DDE messages are sent at once, such as a multiple file select from Windows Explorer. **FileOpenEx** is also supported in Adobe Reader.

**Parameters**

| fullPath       | The full pathname of the file to be opened. |

**Return Value**

true is always returned. The specified file may not actually open.

**Related Methods**

- FileOpen
- CloseAllDocs
- DocClose
- DocOpen
FilePrint

[FilePrint(char* fullPath)]

Description
Prints all pages in a document, displaying a modal print dialog box to the user. For PostScript printing, only Level 1 operators are used, only ASCII data is generated, and the document's pages are not shrunk to fit into the imageable area of the printed page.

FilePrint is also supported in Adobe Reader.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full pathname of the file to be printed.</td>
</tr>
</tbody>
</table>

Return Value
false if the document specified by fullPath does not exist, true otherwise.

Related Methods
DocPrint
FilePrintSilent
FilePrintTo
FilePrintEx

[FilePrintEx(char* fullPath)]

Description

Prints all pages in a document, displaying a modal print dialog box to the user. For PostScript printing, only Level 1 operators are used, only ASCII data is generated, and the document’s pages are not shrunk to fit into the imageable area of the printed page.

Similar to FileOpenEx, this is a special DDE command that returns true right away and performs the action during idle periods. This ensures that no DDE commands are dropped when printing a large number of files simultaneously.

FilePrintEx is also supported in Adobe Reader.

Parameters

| fullPath                       | The full pathname of the file to print.
|-------------------------------|---------------------------------------------|

Return Value

true is always returned.

Related Methods

DocPrint
FileOpenEx
FilePrint
FilePrintSilent
FilePrintSilentEx
FilePrintTo
FilePrintToEx
**FilePrintSilent**

`FilePrintSilent(char* fullPath)`

**Description**

Prints all pages in a document, without displaying a print dialog box to the user. For PostScript printing, only Level 1 operators are used, only ASCII data is generated, and the document's pages are not shrunk to fit into the imageable area of the printed page. **FilePrintSilent** is also supported in Adobe Reader.

**Parameters**

- **fullPath** The full pathname of the file to be printed.

**Return Value**

- *false* if the document specified by **fullPath** does not exist, *true* otherwise.

**Related Methods**

- DocPrint
- FilePrint
- FilePrintTo
FilePrintSilentEx

[FilePrintSilentEx(char* fullPath)]

Description

Prints all pages in a document, without displaying a print dialog box to the user. For PostScript printing, only Level 1 operators are used, only ASCII data is generated, and the document’s pages are not shrunk to fit into the imageable area of the printed page.

Similar to FileOpenEx, this is a special DDE command that returns true right away and does the action during idle periods. This is to ensure that no DDE commands are dropped when printing a large number of files simultaneously.

FilePrintSilentEx is also supported in Adobe Reader.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full pathname of the file to be printed.</td>
</tr>
</tbody>
</table>

Return Value

true is always returned.

Related Methods

DocPrint
FileOpenEx
FilePrintEx
FilePrintSilent
FilePrintTo
FilePrintToEx
**FilePrintTo**

```c
[FilePrintTo(char* fullPath, char* printName,
        char* driverName, char* portName)]
```

**Description**

Prints all pages in a document to a specified printer, using a specified driver and port, displaying a modal print dialog box to the user. For PostScript printing, only ASCII data is generated, and the document’s pages are not shrunk to fit into the imageable area of the printed page. 

*FilePrintTo* is also supported in Adobe Reader.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fullPath</code></td>
<td>The full pathname of the file to be printed.</td>
</tr>
<tr>
<td><code>printName</code></td>
<td><em>(Required for Windows 95 and higher)</em> The name of the printer.</td>
</tr>
<tr>
<td><code>driverName</code></td>
<td>Printer driver name.</td>
</tr>
<tr>
<td><code>portName</code></td>
<td><em>(Required for Windows NT)</em> Port name.</td>
</tr>
</tbody>
</table>

**Return Value**

*false* if the document specified by `fullPath` does not exist, *true* otherwise.

**Related Methods**

- [DocPrint](#)
- [FilePrint](#)
- [FilePrintSilent](#)
FilePrintToEx

[FilePrintToEx(char* fullPath, char* printName,
               char* driverName, char* portName)]

**Description**

Prints all pages in a document to a specified printer, using a specified driver and port, displaying a modal print dialog box to the user. For PostScript printing, only ASCII data is generated, and the document’s pages are not shrunk to fit into the imageable area of the printed page.

Similar to FileOpenEx, this is a special DDE command that returns true right away and does the action during idle periods. This is to ensure that no DDE commands are dropped when printing a large number of files simultaneously.

FilePrintToEx is also supported in Adobe Reader.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full pathname of the file to be printed.</td>
</tr>
<tr>
<td>printName</td>
<td><em>(Required for Windows 95 and higher)</em> The name of the printer.</td>
</tr>
<tr>
<td>driverName</td>
<td>Printer driver name.</td>
</tr>
<tr>
<td>portName</td>
<td><em>(Required for Windows NT)</em> Port name.</td>
</tr>
</tbody>
</table>

**Return Value**

true is always returned.

**Related Methods**

- DocPrint
- FileOpenEx
- FilePrintEx
- FilePrintSilentEx
- FilePrintTo
- FilePrintToEx
FullMenus

[FullMenus()]

Description
Displays full menus, and sets this option in the Acrobat application’s preferences file.
With Acrobat 3.0 or later, all menus are displayed, and this function is ignored.

Parameters

None

Return Value

true if full menus were set successfully, false otherwise.

Related Methods

DocSetViewMode
ShortMenus
**HideToolbar**

```
[HideToolbar()]
```

**Description**
Hides the toolbar.

**Parameters**
None

**Return Value**
*true* if the toolbar was hidden successfully, *false* otherwise.

**Related Methods**
*ShowToolbar*
**MenuItemExecute**

```
MenuItemExecute(char* menuItemName)
```

**Description**

Executes the menu item specified by its language-independent name.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>menuItemName</td>
<td>The language-independent name of the menu item to execute. See the Acrobat And PDF Library API Reference for a list of menu item names.</td>
</tr>
</tbody>
</table>

**Return Value**

None
**ShortMenus**

```python
[ShortMenus()]
```

**Description**
Displays short menus, and sets this option in the Acrobat application’s preferences file. With Acrobat 3.0 or later, all menus are displayed, and this function is ignored.

**Parameters**
None

**Return Value**
true if short menus were set successfully, false otherwise.

**Related Methods**
- `DocSetViewMode`
- `FullMenus`
ShowToolbar

[ShowToolbar()]

Description
Shows the toolbar.

Parameters
None

Return Value
true if the toolbar was shown successfully, false otherwise.

Related Methods
HideToolbar
This portion of the reference contains descriptions of the *Catalog, AcroForm,* and *Search* plug-ins.
Acrobat Catalog is a plug-in that allows you to create a full-text index of a set of PDF documents. A full-text index is a searchable database of all the text in the documents. After building an index, you can use the **Edit > Search** command to search the entire library quickly. Searches of full-text indexes created using Catalog are faster and more convenient than using the **Edit > Find** command.

For more information, see the *Acrobat and PDF Library API Overview*.

---

**Contents**

This part of the document describes the methods available when using DDE.

---

**Catalog Windows Messages**

Catalog broadcasts a set of Windows Messages when certain operations occur. These messages are broadcast whether the operations are initiated from the user interface, HFT methods, or DDE methods.

- **AcrobatCatalogBuildSuccess**: On every successful build.
- **AcrobatCatalogBuildFail**: On every failed build.
- **AcrobatCatalogBuildStopped**: When a build has stopped.
Clients can connect to the Catalog plug-in via DDE using the service name "Acrobat" and the topic name "Control." This section lists the available DDE methods.

---

**AppExit**

```appexit()
```

**Description**

Exits Acrobat Catalog.

**Parameters**

None.

**Return Value**

If `true`, Catalog exited successfully, otherwise `false`. 
AppFront

[AppExit()]

Description
Brings Catalog to the front.

Parameters
None.

Return Value
None.
FileBuild

[FileBuild(char* fullPath)]

**Description**

Builds an index based on the specified index definition file.

**Parameters**

| fullPath | The full pathname of the file to be opened (including the .pdx extension). |

**Return Value**

If *true*, the file opened successfully, otherwise *false*. 
**FileOpen**

```
[FileOpen(char* fullPath)]
```

**Description**

Opens an index definition file and displays the **Edit Index Definition** dialog box.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full pathname of the file to be opened (including the <code>.pdx</code> extension).</td>
</tr>
</tbody>
</table>

**Return Value**

*true* if the file opened successfully, otherwise *false*. 
FilePurge

[FilePurge(char* fullPath)]

Description
Purges an index definition file.

Parameters

| fullPath | The full pathname of the file to be purged (including the .pdx extension). |

Return Value
true if the file was successfully purged, otherwise false.
The Acrobat Forms plug-in allows a Portable Document Format (PDF) document to act as a form; that is, the Acrobat equivalent of a paper form with fields. It is now faster and easier to exchange information either in familiar paper or electronic forms converted to PDF files with Acrobat, or as dynamic interactive database templates.

**NOTE:** Forms as used here do not refer to XObject forms as defined in the *PDF Reference.*

The Forms plug-in for Acrobat (versions 4.0 and above) allows users to author form fields. For Adobe Reader, the Forms plug-in does not allow form authoring, but allows users to fill in data and print Acrobat forms. The Reader Forms plug-in also does not allow users to save data to the local hard disk. Both Acrobat and Reader allow Web designers to send data from the form back to a Web server.

---

**Contents**

This section of the document is a reference for developers who want to take advantage of the Forms API. It describes the OLE Automation methods exported by the Acrobat AcroForm plug-in.

---

**Other Useful Documentation**

You should be familiar with the Acrobat core API. For more information, see the following documents:

- Acrobat and PDF Library API Overview
- Acrobat and PDF Library API Reference
- Acrobat SDK Plug-in Guide
- PDF Reference
Description

The Acrobat Forms plug-in has been enhanced to work as an Automation server in the Win32 environment. Since the automation capabilities have been added to a plug-in, rather than an executable that can be directly launched, the following steps are necessary to access them from an Automation controller:

First instantiate the Acrobat application by using the VB `CreateObject` method. For example:

```
CreateObject("AcroExch.App")
```

This causes the Acrobat Forms plug-in to run, at which time it registers its class object with OLE. Then its main exposed object can be instantiated, that is:

```
CreateObject("AFormAut.App")
```

Presently, registration in the Windows registry (which is different from the class object registration described above) happens every time Acrobat loads the plug-in. Therefore, you must run Acrobat at least once with the AForm32.api file in the plug-ins folder before its type library can be found for object browsing within the Microsoft Visual Studio environment. This is also necessary in order to allow early binding. Declare the program variables as objects of the corresponding classes in `AFORMAUTLib`, and not simply as Object.

The object model was designed in accordance with the applicable standards and guidelines for document-centric applications from the OLE Programmer's Reference. That manual uses the term document to describe whatever an application uses as a file or an individual entity of data (in our case a field).

**NOTE:** Neither Acrobat, nor the Acrobat Forms plug-in, are thread-safe, and therefore Acrobat Forms OLE Automation uses the single-threading model.
Contents

This portion of the document contains the following sections:

- **Acroform OLE Automation Objects.** This section describes the Acrobat objects represented as OLE objects.
- **Acroform OLE Automation Methods.** This section describes each OLE method, including its parameters, return value, and related methods.
- **Acroform Automation Properties.** This section details the properties that can be set in the various objects. Each property describes the key, the property type (for example, read-only), and the semantics.

Conventions

The syntax used in this reference follows that used in Microsoft Visual Basic references.

Exceptions

All methods and properties may return an exception. These may include standard OLE exceptions, such as:

- **E_OUTOFMEMORY** (0x8007000E)
- **E_INVALIDARG** (0x80070057)

These exceptions are not specifically listed in the descriptions of the methods and properties that appear below. Others are AcroForm-specific, and are listed in the following table.

The actual numeric value of the returned exception is assembled as an `HRESULT`, uses the `FACILITY_ITF`, and starts with decimal 512 (hex 0x0200), as recommended by Microsoft. For example, the numeric value of the exception `AutErcNoForm` is 0x80040201. The important part is the right-most (0x201), which is the first error in the enumeration below.

<table>
<thead>
<tr>
<th>Exception Name</th>
<th>Numeric Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutErcNoDoc</td>
<td>1</td>
<td>No document is currently open in the Acrobat application.</td>
</tr>
<tr>
<td>AutErcNotTerminal</td>
<td>2</td>
<td>This property or method applies to terminal fields or their annotations.</td>
</tr>
<tr>
<td>AutErcNotToThisFieldType</td>
<td>3</td>
<td>This property or method is not applicable to this type of field.</td>
</tr>
</tbody>
</table>
**AFormApp**

AFormApp is the only object the controller can externally instantiate (that is, using CreateObject). All other objects must be created by navigating down the hierarchy with the methods and properties described in this section.

---

**Field**

A field in the document that is currently active in Acrobat.

---

**Fields**

A collection of all the fields in the document that are currently active in Acrobat at the time Fields is instantiated.

The Fields collection includes both terminal and non-terminal fields. A terminal field is one that either does not have children, or if it does, they are simply multiple appearances (that is, child annotations) of the field in question.

**NOTE:** If you instantiate a Fields object, and subsequently fields are manually added or removed using the Forms tool in Acrobat, the Fields object will no longer be in sync with the document. You must re-instantiate the Fields object.
Field

PopulateListOrComboBox

void PopulateListOrComboBox ( const VARIANT& arrItems,
                           const VARIANT& arrExportVal);

Description
Specifies the item names and optionally exports values for a field of type listbox or combobox.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrItems</td>
<td>An array of strings, with each element representing an item name.</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE:</strong> There is a limit of 64K for string data in a combo or list box</td>
</tr>
<tr>
<td></td>
<td>control on Windows platforms. For Mac OS systems, the limit is 200 entries</td>
</tr>
<tr>
<td></td>
<td>for the combo or list box control. Using more than these limits degrades</td>
</tr>
<tr>
<td></td>
<td>performance and makes the control unusable.</td>
</tr>
<tr>
<td>arrExportVal</td>
<td><em>(Optional)</em> An array of strings, the same size as the first parameter,</td>
</tr>
<tr>
<td></td>
<td>with each element representing an export value.</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE:</strong> Some of the elements in <code>exportString</code> may be empty strings.</td>
</tr>
</tbody>
</table>

Return Value
None

Exceptions

* Raises AutErcNotToThisFieldType if the field is not of type listbox or combobox.

Related Methods

* Add
SetBackgroundColor

```c
void SetBackgroundColor (LPCTSTR bstrColorSpace, float GorRorC, float GorM, float BorY, float K);
```

**Description**

Specifies the background color for a field. The background color is used to fill the field’s rectangle.

**Parameters**

<table>
<thead>
<tr>
<th>bstrColorSpace</th>
<th>Values are defined by using a <strong>transparent</strong>, <strong>gray</strong>, <strong>RGB</strong> or <strong>CMYK</strong> color space. Valid strings include:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>● &quot;T&quot;</td>
</tr>
<tr>
<td></td>
<td>● &quot;G&quot;</td>
</tr>
<tr>
<td></td>
<td>● &quot;RGB&quot;</td>
</tr>
<tr>
<td></td>
<td>● &quot;CMYK&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GorRorC</th>
<th>Used if bstrColorSpace is set to T, G, or RGB. A float range between zero and one inclusive.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>GorM</th>
<th>Used if bstrColorSpace is set to G. A float range between zero and one inclusive.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>BorY</th>
<th>Used if bstrColorSpace is set to RGB. A float range between zero and one inclusive.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>K</th>
<th>Used if bstrColorSpace is set to CMYK. A float range between zero and one inclusive.</th>
</tr>
</thead>
</table>

**Return Value**

None

**Related Methods**

- SetBorderColor
- SetForegroundColor

**Example**

```c
Field.SetBackgroundColor "RGB", 0.7, 0.3, 0.6, 0
```
SetBorderColor

```c
void SetBorderColor (LPCTSTR bstrColorSpace, float GorRorC, float GorM, float BorY, float K);
```

**Description**

Specifies the border color for a field. The border color is used to stroke the field's rectangle with a line as large as the border width. The new border color is propagated to any child annotations underneath, so the field may be non-terminal.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `bstrColorSpace` | A string specifying the color space. Valid strings include:  
  - "T"  
  - "G"  
  - "RGB"  
  - "CMYK"  |
| `GorRorC`       | Used if `bstrColorSpace` is set to T, G, or RGB. A float range between zero and one inclusive. |
| `GorM`          | Used if `bstrColorSpace` is set to G. A float range between zero and one inclusive. |
| `BorY`          | Used if `bstrColorSpace` is set to RGB. A float range between zero and one inclusive. |
| `K`             | Used if `bstrColorSpace` is set to CMYK. A float range between zero and one inclusive. |

**Return Value**

None

**Related Methods**

- SetBackgroundColor
- SetForegroundColor

**Example**

```c
Field.SetBorderColor "RGB", 0.7, 0.3, 0.6, 0
```
### SetButtonCaption

```c
void SetButtonCaption (LPCTSTR bstrFace, LPCTSTR bstrCaption);
```

**Description**

The caption to be used for the appearance of a field of type `button`.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `bstrFace` | A string that specifies the face for which the caption will be used. Valid strings include:  
  - `N` is the Normal appearance  
  - `D` is the Down appearance  
  - `R` is the appearance for Rollover |
| `bstrCaption` | The caption for the button. |

**NOTE:** If a button’s layout is of type `icon` only, the caption is not used in generating its appearance. In addition, only the `Normal` face is displayed, unless the `Highlight` is of type `push`.

**Return Value**

None

**Exceptions**

Raises `AutErcNotToThisFieldType` if the field is not of type `button`. The new appearance is propagated to any child annotations underneath; the field may be non-terminal.

**Related Methods**

- `SetButtonIcon`

**Example**

```c
Field.SetButtonCaption "D", "Submit Form"
```
SetButtonIcon

void SetButtonIcon (LPCTSTR bstrFace, LPCTSTR bstrFullPath, short pageNum);

Description

Specifies the icon to be used for the appearance of a field of type button.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrFace</td>
<td>A string that specifies the face for which the icon will be used. Valid strings include:</td>
</tr>
<tr>
<td></td>
<td>● N is the Normal appearance</td>
</tr>
<tr>
<td></td>
<td>● D is the Down appearance</td>
</tr>
<tr>
<td></td>
<td>● R is the appearance for Rollover</td>
</tr>
<tr>
<td>bstrFullPath</td>
<td>The full pathname of the PDF file to be used as the source of the appearance.</td>
</tr>
<tr>
<td>pageNum</td>
<td>Used to select the page inside that PDF file (zero-based).</td>
</tr>
</tbody>
</table>

NOTE: If a button's layout is of type icon only, the caption is not used in generating its appearance. In addition, only the Normal face is displayed, unless the Highlight is of type push.

Return Value

None

Exceptions

Raises AutErcNotToThisFieldType if the field is not of type button. The new appearance is propagated to any child annotations underneath, so it is OK if the field is non-terminal.

Related Methods

SetButtonCaption

Example

Field.SetButtonIcon "N", "c:\Clipart.pdf", 0
### SetExportValues

```c
void SetExportValues (const VARIANT& arrExportVal);
```

**Description**
Sets the export values for each of the annotations of a field of type radio button and checkbox.

For radio button fields, this is necessary to make the field work properly as a group. One button is checked at any given time, giving its value to the field as a whole.

For checkbox fields, unless an export value is specified, the default is used when the field checked is **Yes**. When it is unchecked, its value is **Off** (this is also true for a radio button field when none of its buttons are checked).

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrExportVal</td>
<td>An array of strings, which is expected to have as many elements as there are annotations in the field. The elements of the array are distributed among the individual annotations comprising the field, using their tab order.</td>
</tr>
</tbody>
</table>

**Return Value**

None

**Exceptions**

Raises **AutErcNotToThisFieldType** if the field is not of type radio button or checkbox.

**Related Methods**

Add

**Example**

```vbnet
Dim arrExp(1) As String
arrExp(0) = "Visa"
arrExp(1) = "Mastercard"
Field.SetExportValues arrExp
```
SetForegroundColor

void SetForegroundColor (LPCTSTR bstrColorSpace, float GorRorC, float GorM, float BorY, float K);

Description

Specifies the foreground color for a field. It represents the text color for text, button, combobox, or listbox fields and the check color for checkbox or radio button fields.

NOTE: Parameters are similar to SetBorderColor and SetBackgroundColor, except that the transparent color space is not allowed.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrColorSpace</td>
<td>A string specifying the color space. Valid strings include:</td>
</tr>
<tr>
<td></td>
<td>● &quot;T&quot;</td>
</tr>
<tr>
<td></td>
<td>● &quot;G&quot;</td>
</tr>
<tr>
<td></td>
<td>● &quot;RGB&quot;</td>
</tr>
<tr>
<td></td>
<td>● &quot;CMYK&quot;</td>
</tr>
<tr>
<td>GorRorC</td>
<td>Used if bstrColorSpace is set to T, G, or RGB. A float range between zero and one inclusive.</td>
</tr>
<tr>
<td>GorM</td>
<td>Used if bstrColorSpace is set to G. A float range between zero and one inclusive.</td>
</tr>
<tr>
<td>BorY</td>
<td>Used if bstrColorSpace is set to RGB. A float range between zero and one inclusive.</td>
</tr>
<tr>
<td>K</td>
<td>Used if bstrColorSpace is set to CMYK. A float range between zero and one inclusive.</td>
</tr>
</tbody>
</table>

Return Value

None

Related Methods

SetBackgroundColor
SetBorderColor

Example

Field.SetForegroundColor "CMYK", 0.25, 0.25, 0.25, 0.1
Acroform OLE Automation Methods

Field

---

**SetJavaScriptAction**

```c
void SetJavaScriptAction (LPCTSTR bstrTrigger,
                       LPCTSTR bstrTheScript);
```

**Description**

Sets the action of the field to be of type JavaScript. When using `SetJavaScriptAction` within Visual Basic, you may use Chr(13) to add a <CR>, and Chr(9) for tabs, so that the function is nicely formatted.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrTrigger</td>
<td>A string that specifies the trigger for the action. Valid strings include: up, down, enter, exit, calculate, validate, format, keystroke</td>
</tr>
<tr>
<td>bstrTheScript</td>
<td>The script itself.</td>
</tr>
</tbody>
</table>

**NOTE:** If the trigger is calculate, an entry is added at the end of the calculation order array (see the `CalcOrderIndex` property).

**Return Value**

None

**Related Methods**

None

**Calculate Notes**

There is a simple calculate script supplied with Acrobat.

```c
AFSimple_Calculate(cFunction, cFields)
```

- `cFunction` is one of "AVG", "SUM", "PRD", "MIN", "MAX"
- `cFields` is the list of the fields to use in the calculation.
Formatting Notes

The following scripts and formats can be used for the **format** and **keystroke** triggers:

```
AFDate_KeystrokeEx(cFormat)
AFDate_Format(cFormat)
```

- **cFormat** is one of:
  
  - "m/d", "m/d/yy", "mm/dd/yy", "mm/yy", "d-mmm", "d-mmm-yy", "dd-mmm-yy", "yy-mm-dd", "mmm-yy", "mmmmm-yy", "mm d, yyyy", "mmm d, yyyy", "m/d/yy h:MM tt", "m/d/yy HH:MM"

```
AFTime_Keystroke(ptf)
AFTime_Format(ptf)
```

- **ptf** is the time format:
  
  - 0 = 24HR_MM [ 14:30 ]
  - 1 = 12HR_MM [ 2:30 PM ]
  - 2 = 24HR_MM_SS [ 14:30:15 ]
  - 3 = 12HR_MM_SS [ 2:30:15 PM ]

```
AFPercent_Keystroke(nDec, sepStyle)
AFPercent_Format(nDec, sepStyle)
```

- **nDec** is the number of places after the decimal point.
- **sepStyle** is an integer denoting whether to use a separator. If **sepStyle** is 0, use commas. If **sepStyle** is 1, do not separate.

```
AFSpecial_Keystroke(psf)
AFSpecial_Format(psf)
```

- **psf** is the type of formatting to use:
  
  - 0 = zip code
  - 1 = zip + 4
  - 2 = phone
  - 3 = SSN
AFNumber_Format(nDec, sepStyle, negStyle, currStyle, strCurrency, bCurrencyPrepend)
AFNumber_Keystroke(nDec, sepStyle, negStyle, currStyle, strCurrency, bCurrencyPrepend)

- \( n\text{Dec} \) is the number of places after the decimal point.
- \( \text{sepStyle} \) is an integer denoting whether to use a separator. If \( \text{sepStyle} \) is 0, use commas. If \( \text{sepStyle} \) is 1, do not separate.
- \( \text{sepStyle} \) is the formatting used for negative numbers:
  0 = MinusBlack
  1 = Red
  2 = ParensBlack
  3 = ParensRed
- \( \text{currStyle} \) is the currency style - not used.
- \( \text{strCurrency} \) is the currency symbol.
- \( \text{bCurrencyPrepend} \) is true to prepend the currency symbol; false to display on the end of the number.
SetResetFormAction

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

Description
Sets the action of the field to be of type ResetForm.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrTrigger</td>
<td>A string that specifies which trigger is used for the action. Valid strings include:</td>
</tr>
<tr>
<td></td>
<td>● &quot;up&quot; (Mouse Up)</td>
</tr>
<tr>
<td></td>
<td>● &quot;down&quot; (Mouse Down)</td>
</tr>
<tr>
<td></td>
<td>● &quot;enter&quot; (Mouse Enter)</td>
</tr>
<tr>
<td></td>
<td>● &quot;exit&quot; (Mouse Exit)</td>
</tr>
<tr>
<td>theFlags</td>
<td>When 0 (Include), arrFields specifies which fields to include in the reset operation. When non-zero (Exclude), arrFields specifies which fields to exclude from the reset operation.</td>
</tr>
<tr>
<td>arrFields</td>
<td>(Optional) An array of strings for the fully-qualified names of the fields. Depending on the value of theFlags, these fields are included in or excluded from the reset operation. When the fields are included, the set can include the names of non-terminal fields, which is a fast and easy way to cause all their children to be included in the action. When not supplied, all fields are reset.</td>
</tr>
</tbody>
</table>

Return Value
None

Related Methods
None
SetSubmitFormAction

void SetSubmitFormAction (LPCTSTR bstrTrigger,
LPCTSTR bstrTheURL, long theFlags, const VARIANT& arrFields);

Description
Sets the action of the field to be of type SubmitForm.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| bstrTrigger  | A string that specifies which trigger is used for the action. Valid strings include:
  - "up" (Mouse Up)
  - "down" (Mouse Down)
  - "enter" (Mouse Enter)
  - "exit" (Mouse Exit) |
| bstrTheURL   | A string containing the URL. |
| theFlags     | A collection of flags that define various characteristics of the action. Note: See the PDF Reference to learn how the binary value of this long is interpreted. |
| arrFields    | (Optional) If passed, represents an array of strings for the fully-qualified names of the fields to submit when the action is executed. If the array is interpreted as fields to submit (as opposed to fields excluded from the submission, depending on the least-significant bit in the flags), then it may include the names of non-terminal fields, which is a fast and easy way to cause all their children to be included in the submission. If not passed, then the created action will not include a /Fields key. |

Return Value
None

Related Methods
None
**Fields**

**Add**

```cpp
LPDISPATCH Add (LPCTSTR bstrFieldName, LPCTSTR bstrFieldType, short pageNum, float left, float top, float right, float bottom);
```

**Description**

Dynamically adds a new field to the Acrobat form and to the **Fields** collection.

Returns the newly-created **Field** object. You can pass the name of an existing field as a parameter, as long as that field is of the same type as the one being created.

This is useful in the following circumstances:

- For radio buttons to use the **SetExportValues** method to make the radio buttons mutually exclusive.
- For fields that should have multiple appearances (that is, child annotations) in the document.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bstrFieldName</strong></td>
<td>The fully-qualified name of the field.</td>
</tr>
<tr>
<td><strong>bstrFieldType</strong></td>
<td>Field type for the newly created field. Valid types are:</td>
</tr>
<tr>
<td></td>
<td>● &quot;text&quot;</td>
</tr>
<tr>
<td></td>
<td>● &quot;button&quot;</td>
</tr>
<tr>
<td></td>
<td>● &quot;combobox&quot;</td>
</tr>
<tr>
<td></td>
<td>● &quot;listbox&quot;</td>
</tr>
<tr>
<td></td>
<td>● &quot;checkbox&quot;</td>
</tr>
<tr>
<td></td>
<td>● &quot;radio button&quot;</td>
</tr>
<tr>
<td></td>
<td>● &quot;signature&quot;</td>
</tr>
<tr>
<td><strong>pageNum</strong></td>
<td>The page number (zero-based).</td>
</tr>
</tbody>
</table>

**NOTE:** When creating list or combo boxes, there is a limit of 64K for string data on Windows platforms. MacOS systems have a limit of 200 entries for the list or combo boxes. Using more than the limit degrades performance. You populate the fields of the list and combo boxes using the **PopulateListOrComboBox** method.
Acroform OLE Automation Methods

Fields

Return Value

The newly-created Field object.

Related Methods

- PopulateListOrComboBox
- Remove

Example

Set Field = Fields.Add("payment", "radiobutton", 0, 100, 600, 130, 570)
AddDocJavascript

```c
void AddDocJavascript (LPCTSTR bstrScriptName,
                      LPCTSTR bstrTheScript);
```

**Description**

Adds a document-level JavaScript function to the PDF file. When using `AddDocJavascript`, within Visual Basic, you can use `Chr(13)` to add a `<CR>`, and `Chr(9)` for tabs, so that the function is nicely formatted.

**Parameters**

<table>
<thead>
<tr>
<th>bstrScriptName</th>
<th>The name of the function to be added to the document.</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrTheScript</td>
<td>The definition to be added to the document.</td>
</tr>
</tbody>
</table>

**Return Value**

None

**Related Methods**

`ExecuteThisJavascript`

**Example**

```vba
'Adding a document-level JavaScript function, to compute factorials:
Fields.AddDocJavaScript "Fact", 
"function Fact(n)" & Chr(13) & _
{" & Chr(13) & _
Chr(9) & "if (n <= 0)" & Chr(13) & _
Chr(9) & Chr(9) & "return 1;" & Chr(13) & _
Chr(9) & "else" & Chr(13) & _
Chr(9) & Chr(9) & "return n * Fact(n - 1);" & Chr(13) & _
"}"
```
**ExecuteThisJavascript**

```cpp
CString ExecuteThisJavascript (LPCTSTR bstrTheScript);
```

**Description**

Executes the specified JavaScript script.

**Parameters**

- **bstrTheScript**: A string containing a JavaScript script, which is executed by Acrobat in the context of the currently active document.

**NOTE:** See the *Acrobat JavaScript Scripting Reference* for information on event level values.

**Return Value**

Returns a result by assigning it to event value.

**Related Methods**

- **AddDocJavascript**

**Example**

```javascript
Fields.ExecuteThisJavascript "var f = _this.getField("myButton");
f.delay = _false;"
```

To get the return value in Visual Basic:

```vbscript
Dim cSubmitName As String
cSubmitName = Fields.ExecuteThisJavaScript
"event.value = this.getField("myField").submitName;"
```
ExportAsFDF

void ExportAsFDF (LPCTSTR bstrFullPath,
LPCTSTR bstrSubmitButton, BOOL bEmptyFields,
const VARIANT& arrFields);

Description

Exports the data as FDF from an Acrobat form.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrFullPath</td>
<td>A full pathname of the file to which the produced FDF file will be saved.</td>
</tr>
<tr>
<td>bstrSubmitButton</td>
<td>The name of an existing form field of type button (in case you want to include it in the FDF file, as if it had been used to trigger a SubmitForm action). You may pass an empty string.</td>
</tr>
<tr>
<td>bEmptyFields</td>
<td>A boolean to indicate whether fields with no value should be included in the produced FDF file.</td>
</tr>
<tr>
<td>arrFields</td>
<td>(Optional) An array of strings representing the fully-qualified names of the fields to include in the FDF file. This array may include the names of non-terminal fields, which is a fast and easy way to cause all their children to be included in the FDF file.</td>
</tr>
</tbody>
</table>

Return Value

None

Related Methods

ImportAnFDF
ExportAsHtml

Example

Dim arrFields(1) As String
arrFields(0) = "name"
arrFields(1) = "address"
' This will create an FDF that includes 'name.last, name.first, address.street, 'etc., but only if they have a value ' (since we are passing False for the ' "bEmptyFields" parameter.
Fields.ExportAsFDF "C:\Temp\out.fdf", "", False, arrFields
ExportAsHtml

void ExportAsHtml (LPCTSTR bstrFullPath,
LPCTSTR bstrSubmitButton, BOOL bEmptyFields,
const VARIANT& arrFields);

Description

Exports the data as HTML from an Acrobat form. This method is similar to ExportAsFDF. The only difference is that the form data is exported in URL-encoded format.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrFullPath</td>
<td>A full pathname of the file to which the produced FDF file will be saved.</td>
</tr>
<tr>
<td>bstrSubmitButton</td>
<td>The name of an existing form field of type button (in case you want to include it in the FDF file, as if it had been used to trigger a SubmitForm action). You may pass an empty string.</td>
</tr>
<tr>
<td>bEmptyFields</td>
<td>A boolean to indicate whether fields with no value should be included in the produced FDF file.</td>
</tr>
<tr>
<td>arrFields</td>
<td>(Optional) An array of strings representing the fully-qualified names of the fields to include in the FDF file. This array may include the names of non-terminal fields, which is a fast and easy way to cause all their children to be included in the FDF file.</td>
</tr>
</tbody>
</table>

Return Value

None

Related Methods

ExportAsFDF
ImportAnFDF

void ImportAnFDF (LPCTSTR bstrFullPath);

**Description**
Imports the FDF file into an Acrobat form.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrFullPath</td>
<td>The full pathname of the file containing the FDF file to be imported.</td>
</tr>
</tbody>
</table>

**Return Value**
None

**Related Methods**

ExportAsFDF
**Remove**

```cpp
void Remove (LPCTSTR bstrFieldName);
```

**Description**

Removes a field from the Acrobat Form and from the *Fields* collection.

**Parameters**

| `bstrFieldName` | The fully-qualified name of the field to be removed from the Acrobat form. If the field has multiple child annotations, all of them are removed. If multiple fields have the same name, all are removed. |

**Return Value**

None

**Related Methods**

*Add*

**Example**

```
'Remove fields you no longer used.
Fields.Remove("MyOldField")
```
Field

Alignment

[get/set] String

Description

The text alignment of a text field. Valid alignments are (must be spelled exactly as shown):

left
center	right

Return Value

If the field is terminal and has multiple child annotations, a get returns the alignment for the first child, whichever annotation that happens to be.

On a set, the property is propagated to any child annotations underneath, so the field may be non-terminal.

Exceptions

If the field is not of type text, an exception AutErcNotToThisFieldType is returned.

On a get, if the field is non-terminal, an exception AutErcNotTerminal is returned.

Example

Field.Alignment = left
**BorderStyle**

[get/set] String

**Description**

The border style for a field. Valid border styles include **solid**, **dashed**, **beveled**, **inset**, and **underline**.

**Return Value**

If it is terminal and has multiple child annotations, a get returns the value of the border style for the first child, whichever annotation that happens to be.

On a set, the property is propagated to any child annotations underneath, so the field may be non-terminal.

**Exceptions**

On a get, raises **AutErcNotTerminal** if the field is non-terminal, an exception is returned.

**Example**

```
Field.BorderStyle = "beveled"
```
**BorderWidth**

[get/set] short

**Description**

The thickness of the border when stroking the perimeter of a field’s rectangle. If the border color is transparent, this property has no effect except in the case of a beveled border. The value **0** represents no border, and the value **3** represents a thick border.

**Return Value**

If it is terminal and has multiple child annotations, a get returns the value of the border width for the first child, whichever annotation that happens to be.

On a set, the property is propagated to any child annotations underneath, so the field may be non-terminal.

**Exceptions**

On a get, if the field is non-terminal, an exception **AutErcNotTerminal** is returned.

**Example**

Field.BorderWidth = 1
**ButtonLayout**

[get/set] short

**Description**

The layout appearance of a button. Valid values include:

- 0 - text only; the button has a caption but no icon.
- 1 - icon only; the button has an icon but no caption.
- 2 - icon over text; the icon should appear on top of the caption.
- 3 - text over icon; the text should appear on top of the icon.
- 4 - icon then text; the icon should appear to the left of the caption.
- 5 - text then icon; the icon should appear to the right of the caption.
- 6 - text over icon; the text should be overlaid on top of the icon.

If it is terminal and has multiple child annotations, a get returns the layout for the first child, whichever annotation that happens to be.

On a set, the property is propagated to any child annotations underneath, so it is OK if the field is non-terminal.

**Exceptions**

If the field is not of type `button`, an exception `AutErcNotToThisFieldType` is returned.

On a get, if the field is non-terminal, an exception `AutErcNotTerminal` is returned.

**Example**

Field.ButtonLayout = 2
**CalcOrderIndex**

[get/set] short

**Description**

The zero-based calculation order of fields in the document. If you want the calculation for a field $f_2$ to be performed after that for field $f_1$, you need only set the `CalcOrderIndex` for $f_2$ to $f_1$'s `CalcOrderIndex` + 1. The elements in the calculation order array are shifted to make room for the insertion (but the first calculation is still at index 0).

For more information, see the Acrobat JavaScript Scripting Reference.

**Example**

```javascript
Set F1 = Fields("SubTotal")
Set F2 = Fields("Total")
F2.CalcOrderIndex = F1.CalcOrderIndex + 1
```
CharLimit

[get/set] short

Description
The limit on the number of characters that a user can type into a text field.
On a set, the property is propagated to any child annotations underneath, if any.

Exceptions
If the field is not of type text, an exception AutErcNotToThisFieldType is returned.
DefaultValue

[get/set] String

Description
The default value of the field. It returns the empty string if the field has no default value. If the field is non-terminal, an exception `AutErcNotTerminal` is returned.

See also `Value`.
**Editable**

[get/set] Boolean

**Description**
Determined whether the user can type in a selection or must choose one of the provided selections. Comboboxes can be editable; that is, the user can type in a selection.

On a set, the property is propagated to any child annotations underneath, if any.

**Exceptions**
Returns an exception of `AutErcNotToThisFieldType` if the field is not of type combobox.

**Example**

```
Field.Ediable = False
```
Highlight

[get/set] String

Description

Defines how a button reacts when a user clicks it. The four highlight modes supported are:

- none
- invert
- push
- outline

If it is terminal and has multiple child annotations, a get returns the highlight for the first child, whichever annotation that happens to be.

On a set, the property is propagated to any child annotations underneath, so the field may be non-terminal.

Exceptions

If the field is not of type button, an exception AutErcNotToThisFieldType is returned.

On a get, if the field is non-terminal, an exception AutErcNotTerminal is returned.

Example

Field.Highlight = "invert"
IsHidden

[get/set] Boolean

Description
Determines whether the field is hidden or visible to the user. If the value is \texttt{true} the field is invisible, and \texttt{false} indicates that the field is visible.

During get operations, if the field is non-terminal, an exception \texttt{AutErcNotTerminal} is returned. If it is terminal, and has multiple child annotations, a get returns the value of the hidden flag for the first child, whichever annotation that happens to be.

During set operations, the property is propagated to any child annotations underneath, so it is OK if the field is non-terminal.

Example

```csharp
'Hide "name.last"
Set Field = Fields("name.last")
Field.IsHidden = True
```
IsMultiline

[get/set] Boolean

Description
Determines whether the text field is multi-line or single-line.
On a set, the property is propagated to any child annotations underneath, if any.

Exceptions
If the field is not of type text, an exception AutErcNotToThisFieldType is returned.

Example
Field.IsMultiline = True
IsPassword

[get/set] Boolean

Description
Determines whether the field will display asterisks for the data entered. Upon submission, the actual data entered is sent. Fields that have the password attribute set will not have the data in the field saved when the document is saved to disk.

On a set, the property is propagated to any child annotations underneath, if any.

Exceptions
If the field is not of type text, an exception AutErcNotToThisFieldType is returned.

Example
Field.IsPassword = True
IsReadOnly

[get/set] Boolean

Description
The read-only characteristic of a field. When a field is read-only, the user can see the field but cannot change it. If a button is read-only, the user cannot press it to execute an action.

Because this is a field flag and not an annotation flag, both a get and a set of this property are allowed regardless of whether the field is terminal or non-terminal.

- A get on a non-terminal field retrieves that field's flag.
- A set changes the flag on all its terminal children.
**IsRequired**

[get/set] Boolean

**Description**

The required characteristic of a field. When a field is required, its value must be non-`NULL` when the user clicks a submit button that causes the value of the field to be sent to the web. If the field value is `NULL`, the user receives a warning message and the submit does not occur.

Since this is a field flag and not an annotation flag, both a get and a set of this property are allowed, regardless of whether the field is terminal or non-terminal.

A get on a non-terminal field retrieves that field’s flag.

A set changes the flag on all its terminal children.
IsTerminal

[read-only] Boolean

**Description**

true if the field is terminal, otherwise false.

**Example**

```vbnet
Dim Field As AFORMAUTLib.Field
Dim bTerminal As Boolean

' bTerminal should be True
bTerminal = Field.IsTerminal
```
Name

[read-only] String

Description

The fully qualified name of the field. It is the default member of the Field interface.
NoViewFlag

[get/set] Boolean

Description
Determines whether a given field prints but doesn’t display on the screen.

Set the NoViewFlag property to true to allow the field to appear when the user prints the document but not when it displays on the screen; set it to false to allow both printing and displaying.

On a get, if the field is non-terminal, an exception AutErcNotTerminal is returned. If it is terminal, and has multiple child annotations, a get returns the value of the no-view flag for the first child, whichever annotation that happens to be.

On a set, the property is propagated to any child annotations underneath, so the field may be non-terminal.
**PrintFlag**

[get/set] Boolean

**Description**

Determines whether a field prints. Set the `PrintFlag` property to `true` to allow the field to appear when the user prints the document, set it to `false` to prevent printing.

On a get, if the field is non-terminal, an exception `AutErcNotTerminal` is returned. If it is terminal, and has multiple child annotations, a get returns the value of the print flag for the first child, whichever annotation that happens to be.

On a set, the property is propagated to any child annotations underneath, so the field may be non-terminal.
Style

[get/set] String

Description

The style of a checkbox or a radio button (the glyph used to indicate that the check box or radio button has been selected).

Valid styles include:

- check
- cross
- diamond
- circle
- star
- square

If it is terminal and has multiple child annotations, a get returns the style for the first child, whichever annotation that happens to be.

On a set, the property is propagated to any child annotations underneath, so it is OK if the field is non-terminal.

Exceptions

- During set, if the field is not of type checkbox or radio button, an exception AutErcNotToThisFieldType is returned.
- On a get, if the field is non-terminal, an exception AutErcNotTerminal is returned.

Example

Field.Style = "star"
**TextField**

[get/set] String

**Description**

The text font used when laying out the field. Valid fonts include:

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

On a set, the property is propagated to any child annotations underneath, if any.

**Example**

```csharp
Field.TextField = "Times-BoldItalic"
```
**TextSize**

[get/set] short

**Description**

The text points size used in the field. In combobox and radio button fields, the text size determines the size of the check. Valid text sizes include zero and the range from 4 to 144 inclusive.

A text size of zero means that the largest point size that can still fit in the field’s rectangle should be used (in multi-line text fields and buttons this is always 12 points).

On a set, the property is propagated to any child annotations underneath, if any.

**Example**

Field.TextSize = 18
Acroform Automation Properties

Field

Type

[read-only] String

Description
The type of the field as a string. Valid types that are returned:

text
button
combobox
listbox
checkbox
radiobutton
signature

Example
Set Field = Fields("name.last")
'Should print "name.last"
print Field
' Should print the type of field. Example,
' "text"
print Field.Type
Value

[get/set] String

Description
A string that represents the value of the field. Returns the empty string if the field has no value. If the field is non-terminal, an exception AutErcNotTerminal is returned.

For fields of type checkbox, the value Off represents the unchecked state. The checked state is represented using the export value. This is also true for radio buttons (where each individual button in a group should have a different export value; see SetExportValues). For fields of type listbox or combobox, if an export value is defined, then that represents the value, otherwise the item name is used.

These remarks apply also to DefaultValue.

Example

Dim arrExp(1) As String
arrExp(0) = "Visa"
arrExp(1) = "Mastercard"
Field.SetExportValues arrExp
Field.Value = arrExp(0)
Fields

Count

[read-only] long

Description
The number of items in the collection.

Example

Dim Field As AFORMAUTLib.Field
Dim nFields As Long

nFields = Fields.Count

For Each Field In Fields
    If Field.IsTerminal Then
        print Field.Value
    End If
Next Field
**Item**

```plaintext
[read-only] IDispatch*
```

**Description**

Takes the fully qualified name of the field (for example, "name.last") as a parameter, and returns the Field object for it. It is the default member of the Fields interface (that is, the property invoked if the object name is specified by itself without a property or a method in the controller script).

**Example**

```plaintext
Dim Field As AFORMAUTLib.Field
Dim nFields As Long

Set Field = Fields.Item("name.last")
'Since Item is the default property:
Set Field = Fields("name.last")
```
_NewEnum

[read-only] IUnknown*

Description

The IEnumVariant enumerator for the collection.

**NOTE:** You do not need to call this property directly. Visual Basic calls it in the background whenever the code contains a “For Each Field In Fields” loop. For example:

```
For Each Field in Fields
    If Field.IsTerminal
        print Field.Value
    End If
Next Field
```
The Acrobat Search plug-in allows users to perform text searches in PDF documents. It adds menus, menu items, toolbar buttons, and a Search panel to the Acrobat application. The Search plug-in exports a Host Function Table (HFT) containing several methods that can be used by other plug-ins.

Search supports interapplication communication (IAC) in the form of Apple events on the Macintosh and DDE messages under the Windows operating system. These Apple events and DDE messages allow remote clients to submit search queries and manipulate a list of indexes (the list of indexes is referred to as the shelf).

This section describes the HFT and IAC APIs supported by the Acrobat Search plug-in. It also contains the names of items added to the Acrobat application user interface by the Search plug-in.

For more information, see the *Acrobat and PDF Library API Overview*. 
Contents

This portion of the document contains the following sections:

- **Search Apple Events** describes in detail each Apple event, including parameters and return value.
- **DDE Messages** describes in each DDE message and its arguments.
- **Search Lists** describes Search dialog boxes, menu item names, and toolbar button names.
A client can connect to the Search plug-in via DDE using the service name "Acrobat Search" and the topic name "Acrobat Search".

```c
DdeInitialize(&id, &DDE_ProcessMessage, APPCMD_CLIENTONLY, 0);
hszServerName = DdeCreateStringHandle(id, "Acrobat Search", 0);
hszTopicName = DdeCreateStringHandle(id, "Acrobat Search", 0);
hConv = DdeConnect(id, hszServerName, hszTopicName, NULL);
```

After a connection has been made, a single poke transaction will submit a search query. Two types of queries are supported: simple query and query.

---

**Simple Query Item**

A simple query has the item name “SimpleQuery”. When using a simple query, pass only a string that contains the query, using the ASQL query parser’s format (see `QLangType_CQL` in Table 1). It is not possible to choose another parser or to set word options using the simple query item.
Query Item

Query has the item name “Query”. When using query, a **QueryData** structure is passed. This structure contains the query, as well as specifying the query parser to use and additional options.

```c
hszItemName = DdeCreateStringHandle(id, "Query", 0);
DdeClientTransaction(qd, nLen, hConv, hszItemName, CF_TEXT, XTYP_POKE, 1000, &dwResult);
DdeDisconnect(hConv)
```

The global data handle (**qd**) passed to the server must be in the following format:

```c
typedef struct _QueryData {
    eQLangType qlt;
    boolean bOverrideWordOptions;
    uns32 nWordOptions;
    uns16 nMaxDocs;
    uns16 nQueryOffset;
    uns16 nNumSorts; //deprecated in Acrobat 6.0
    uns16 nSortOffset[QP_MAX_SORT_FIELDS]; //deprecated in Acrobat 6.0
    boolean bSortWays[QP_MAX_SORT_FIELDS]; //deprecated in Acrobat 6.0
    unsigned char cData[1];
} QueryData;
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>qlt</strong></td>
<td>The query language type. Must be one of the values shown in Table 1.</td>
</tr>
<tr>
<td><strong>bOverrideWordOptions</strong></td>
<td>Indicates that the client wishes to use different word options than those currently set by the user.</td>
</tr>
<tr>
<td><strong>nWordOptions</strong></td>
<td>The word options. Must be an <strong>OR</strong> of the values shown in Table 2.</td>
</tr>
<tr>
<td><strong>nMaxDocs</strong></td>
<td>If non-zero, the client wishes to use a different limit for the maximum number of documents than the limit currently set by the user.</td>
</tr>
<tr>
<td><strong>nSortOffsets</strong></td>
<td>A list of offsets into the <strong>cData</strong> chunk. Each offset points to a <strong>NULL</strong>-terminated string containing the field name.</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong>: This value has no effect in Acrobat 6.0 or later, because sort options are not valid.</td>
</tr>
<tr>
<td><strong>nQueryOffset</strong></td>
<td>An offset into the <strong>cData</strong> chunk that points to a <strong>NULL</strong>-terminated string containing the query to execute.</td>
</tr>
</tbody>
</table>
nNumSorts  The number of fields in the sort spec. If this number is 0, the plug-in uses the current sort spec set by the user.

**NOTE:** This value has no effect in Acrobat 6.0 or later, because sort options are not valid.

bSortWays  A list of sort order flags, one for each sort field. true indicates an ascending sort, and false indicates a descending sort.

**NOTE:** This value has no effect in Acrobat 6.0 or later, because sort options are not valid.

### TABLE 1  Query language type constants

<table>
<thead>
<tr>
<th>QLangType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QLangType_Simple</td>
<td>Allows only simple phrase searches; does not allow boolean searching.</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE:</strong> This query type does not work in the DDE interface of the Search plug-in shipped with version 2.0 of Acrobat.</td>
</tr>
<tr>
<td>QLangType_CQL</td>
<td>Allows boolean searches using AND, OR, and NOT, as described in the Acrobat Search plug-in’s online help file.</td>
</tr>
<tr>
<td>QLangType_Passthrough</td>
<td>The Verity BooleanPlus® query language. Contact Verity for further information on this language.</td>
</tr>
</tbody>
</table>

### TABLE 2  Word option bit-flag constants

<table>
<thead>
<tr>
<th>QPON</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case</td>
<td>The search is case-sensitive.</td>
</tr>
<tr>
<td>Stemming</td>
<td>Find not only the specified word, but other words that have the same stem (for example, run and ran have the same stem).</td>
</tr>
<tr>
<td>SoundsLike</td>
<td>Find not only the specified word, but other words that sound like it.</td>
</tr>
<tr>
<td>Thesaurus</td>
<td>Find not only the specified word, but other words that have the same meaning.</td>
</tr>
</tbody>
</table>
To create and populate this structure correctly, the client must know the sum of the lengths of each sort field (sls), the length of the query (lq), and the size of the `QueryData` structure. The client then allocates memory as follows:

```c
nSize = sizeof(QueryData) + sls + lq;
qd = (QueryData *)malloc(nSize);
```

For example, if the query was “Adobe” and the sort spec was “Title” ascending and “Score” descending then the structure would be packed as follows:

```c
memset(qd, 0, nSize);
qd->nQueryOffset = 0;
strcpy(&cData[0], "Adobe");
qd->nNumSort = 2;
qd->nSortOffset[0] = strlen("Adobe") + 1;
qd->bSortWays[0] = TRUE;
strcpy(&cData[qd->nSortOffset[0]], "Title");
qd->bSortWays[1] = FALSE;
qd->nSortOffset[1] = qd->nSortOffset[0] + strlen("Title") + 1;
strcpy(&cData[qd->nSortOffset[1]], "Score");
```

**TABLE 2  Word option bit-flag constants**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QPON_Proximity</td>
<td>Consider the proximity of results when using the <strong>AND</strong> operator to look for more than one word in a document. Without this option, <strong>AND</strong> terms can be anywhere in a document. Searching for “red” and “blue,” for example, finds a document where “red” is the first word on the first page and where “blue” is the last word on the last page. With this option, however, <strong>AND</strong> terms must be within two or three pages of each other to be found. Also, the closer <strong>AND</strong> terms appear together, the higher the relevance ranking of the document that contains them.</td>
</tr>
<tr>
<td>QPON_Refine</td>
<td>Do not search the entire list of indices, but only the documents that matched the previous search. This is used to refine the results of the previous search.</td>
</tr>
</tbody>
</table>
Manipulating Indices Through DDE

After a connection has been made, a single poke transaction can add, delete, add, or remove indices. The item name to use is “Index”.

```c
hszItemName = DdeCreateStringHandle(id, "Index", 0);
DdeClientTransaction(qd, nLen, hConv, hszItemName, CF_TEXT, XTYP_POKE, 1000, &dwResult);
DdeDisconnect(hConv);
```

The global data handle (gd) passed to the server must be in the following format:

```c
typedef struct _IndexData {
    IndexActionType eAction;
    int16 nIndexOffset;
    int16 nTempNameOffset;
    unsigned char cData[1];
} IndexData;
```

<table>
<thead>
<tr>
<th>eAction</th>
<th>The operation to be performed on the index, and must be one of values listed in Table 3.</th>
</tr>
</thead>
<tbody>
<tr>
<td>nIndexOffset</td>
<td>An offset into the cData chunk that points to a NULL-terminated string containing the .PDX file representing the index.</td>
</tr>
<tr>
<td>nTempNameOffset</td>
<td>An offset into cData. It points to a temporary name that is displayed by the Search plug-in when the index is unavailable. This field must specify an offset either to an empty string (\0) or to a non-empty C string.</td>
</tr>
</tbody>
</table>

**Table 3**  
*Search plug-in index operation selectors for DDE messages*

<table>
<thead>
<tr>
<th>IndexAction_Add</th>
<th>Adds an index to the shelf.</th>
</tr>
</thead>
<tbody>
<tr>
<td>IndexAction_Remove</td>
<td>Removes an index from the shelf.</td>
</tr>
<tr>
<td>IndexAction_Enable</td>
<td>Enables an index on the shelf.</td>
</tr>
<tr>
<td>IndexAction_Disable</td>
<td>Disables an index on the shelf.</td>
</tr>
</tbody>
</table>
To create and populate this structure correctly, the client must know the sum of the lengths of the Index (\( li \)) and Temp names (\( lt \)) (including NULL-terminating characters), and the size of the IndexData structure.

The client then allocates memory as follows:

\[
\text{nSize} = \text{sizeof(IndexData)} + li + lt; \\
\text{id} = (\text{IndexData} *)\text{malloc(nSize)};
\]

For example, to add the index C:\FOO.PDX to the Search plug-in's shelf:

```c
memset(id, 0, nSize);
id->eAction = IndexAction_Add;
id->nIndexOffset = 0;
strcpy(&id->cData[0], "C:\FOO.PDX");
id->nTempNameOffset = strlen("C:\FOO.PDX") + 1;
strcpy(&id->cData[id->nTempNameOffset], "My Favorite Index");
```
SearchAddIndex

Description

Adds a specified index to the shelf.

Apple Event ID

kSearchAddIndex (‘addx’)

Apple Event Parameters

<table>
<thead>
<tr>
<th>Tag</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kIndexListTag (‘SilP’), typeLongInteger</td>
<td>An opaque void* representing the shelf, obtained from SearchGetIndexList.</td>
<td></td>
</tr>
<tr>
<td>kPathTag (‘Path’), typeChar</td>
<td>Macintosh full path representing an index, of the form:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MyDisk:TopFolder:BottomFolder:Strange.pdx</td>
</tr>
<tr>
<td>kFlagTag (‘Flag’), typeLongInteger</td>
<td>Index flags. See SearchGetIndexFlags for a description of them. The kIndexAvailable flag should always be set.</td>
<td></td>
</tr>
</tbody>
</table>

Return Value

kIndexTag (‘SixP’), typeLongInteger
An opaque void* representing an index. Returns NULL if failure. Returns

#define kIndexExists ((SearchIndexPtr)-1)
if the index already exists in the index list. If the index already exists, you can retrieve it using SearchGetIndexByPath.
SearchCountIndexList

Description

Gets the number of indices currently on the shelf.

Apple Event ID

kSearchCountIndexList (‘cidx’)

Apple Event Parameters

kIndexListTag (‘SilP’), typeLongInteger
An opaque void* representing the shelf, obtained from SearchGetIndexList.

Return Value

kIndexListTag (‘SilP’), typeLongInteger
Number of indices on the shelf (kIndexListTag here is not semantically correct, but works).
SearchDoQuery

Description
Executes a specified query, using the set of indices currently on the shelf. The search results are displayed in the Acrobat Search plug-in's Results window.

Apple Event ID
kSearchDoQuery (\"kwry\")

Apple Event Parameters

- **kqueryStringTag** (\"Qury\"), typeChar
  The query string, a NULL-terminated block of text. Its format is the same as what a user would type into the search Query window, and depends on the search language specified by **kParserTag**.

- **kParserTag** (\"Prsr\"), typeShortInteger
  The query parser to use; may be one of (see **SrchType.h**):
  - **kParserSimple** 0 — Allows only simple phrase searches; does not allow boolean searching.
  - **kParserCQL** 1 — Allows boolean searches using **AND**, **OR**, and **NOT**, as described in the Acrobat Search plug-in's online help file.
  - **kParserBPlus** 2 — The Verity BooleanPlus query language. Contact Verity for further information on this language.

- **kSortSpecTag** (\"Sort\"), typeAEList
  A list of C strings representing fields to sort by. The first element is the first level sort, the second is the second level sort, and so forth.
  Each string may be any field that appears in the index, plus **Score** (which sorts results by relevance ranking).
  Some common fields are Title, ModificationDate, CreationDate, and Keywords.

- **kWordOptionsTag** (\"WOpt\"), typeLongInteger
  A bit field of word options. Must be a logical OR of the values listed below in Search Plug-in Word Options For Apple Events.
  The manner in which the options are used depends on the value associated with **kOptionsOverrideTag**.

- **kOptionsOverrideTag** (\"WOer\"), typeShortInteger
  Flag that indicates whether the word options are OR'ed with the search options set in the user interface, or used instead of them. If 0, the word options are OR'ed with the user interface search options, and the resulting value is used. If non-zero, the word options are used instead of the user interface search options.
**kMaxDocsTag**
('MaxD'),
**typeShortInteger**
The maximum number of documents to display in the Results window. If more documents than this have hits, only the first \texttt{maxDocs} are displayed. \texttt{maxDocs} cannot be greater than 999.

**Return Value**
None

**Search Plug-in Word Options For Apple Events**

<table>
<thead>
<tr>
<th>kWordOptionCase</th>
<th>The search is case-sensitive.</th>
</tr>
</thead>
<tbody>
<tr>
<td>kWordOptionStemming</td>
<td>Find not only the specified word, but other words that have the same stem (for example, run and ran have the same stem).</td>
</tr>
<tr>
<td>kWordOptionSoundsLike</td>
<td>Find not only the specified word, but other words that sound like it.</td>
</tr>
<tr>
<td>kWordOptionThesaurus</td>
<td>Find not only the specified word, but other words that have the same meaning.</td>
</tr>
<tr>
<td>kWordOptionProximity</td>
<td>Consider the proximity of results when using the \texttt{AND} operator to look for more than one word in a document. Without \texttt{kWordOptionProximity}, \texttt{AND} terms can be anywhere in a document. Searching for “red” and “blue,” for example, finds a document where “red” is the first word on the first page and where “blue” is the last word on the last page. With \texttt{kWordOptionProximity}, however, \texttt{AND} terms must be within two or three pages of each other to be found. Also, with \texttt{kWordOptionProximity}, the closer \texttt{AND} terms appear together, the higher the relevance ranking of the document that contains them.</td>
</tr>
<tr>
<td>kWordOptionRefine</td>
<td>Do not search the entire list of indices, but only the documents that matched the previous search. This is used to refine the results of the previous search.</td>
</tr>
</tbody>
</table>
SearchGetIndexByPath

Description
Gets the index that has the specified path. The index must already be on the shelf. The index can be passed to other Search Apple events to remove it from the shelf, obtain its title, and so forth.

Apple Event ID
kSearchGetIndexByPath (‘fpdx’)

Apple Event Parameters

<table>
<thead>
<tr>
<th>Tag</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kIndexListTag</td>
<td>void*</td>
<td>An opaque void* representing the shelf, obtained from SearchGetIndexList.</td>
</tr>
<tr>
<td>(‘SilP’) typeLongInteger</td>
<td></td>
<td></td>
</tr>
<tr>
<td>kPathTag</td>
<td>Char</td>
<td>Macintosh full path representing an index, of the form: MyDisk:TopFolder:BottomFolder:Strange.pdx</td>
</tr>
<tr>
<td>(‘Path’) typeChar</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Return Value

<table>
<thead>
<tr>
<th>Tag</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kIndexTag</td>
<td>void*</td>
<td>An opaque void* representing an index. Returns NULL if the specified index is gone.</td>
</tr>
<tr>
<td>(‘SixP’), typeLongInteger</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**SearchGetIndexFlags**

**Description**

Get the flags for an index.

**Apple Event ID**

kSearchGetIndexFlags (‘gfdx’)

**Apple Event Parameters**

kIndexTag (‘SixP’), typeLongInteger

An opaque void* representing an index.

**Return Value**

kFlagTag (‘Flag’), typeLongInteger

A logical OR of the following:

kIndexAvailableFlag (1L << 0) — Set if the index is available for searching.

kIndexSelectedFlag (1L << 1) — Set if the index appears with a check mark in the Search plug-in’s user interface.

kIndexPtrInvalidFlag (1L << 31) — Set if the index is not valid or is no longer valid.
**SearchGetIndexList**

**Description**

Gets a list of the indices currently on the shelf.

**Apple Event ID**

kSearchGetIndexList ('gidx')

**Apple Event Parameters**

None

**Return Value**

kIndexListTag ('SilP'), typeLongInteger

An opaque void* representing the list of indices currently on the shelf. This value can subsequently be used by other search Apple events to obtain information about a specific index, the number of indices on the shelf, and so forth.
SearchGetIndexPath

Description

Gets the full path to an index.

Apple Event ID

kSearchGetIndexPath (‘gpdx’)

Apple Event Parameters

kIndexTag (‘SixP’), typeLongInteger

An opaque void* representing the index whose path is to be obtained. The index may be obtained using SearchGetIndexByPath, SearchGetNthIndex, or SearchAddIndex.

Return Value

kPathTag (‘Path’), typeChar

A NULL-terminated character string representing the full path of the index. Returns an empty string if the requested index is not valid.
SearchGetIndexTitle

Description

Gets the title of an index.

Apple Event ID

kSearchGetIndexTitle ('gtdx')

Apple Event Parameters

| kIndexTag ('SixP'), typeLongInteger | An opaque void* representing the index whose title is to be obtained. The index may be obtained using SearchGetIndexByPath, SearchGetNthIndex, or SearchAddIndex. |

Return Value

| kTitleTag ('Titl'), typeChar | A NULL-terminated character string representing the title of the index. If there is no title, it will return the index’s path. Returns an empty string if the requested index is not valid. |
SearchGetNthIndex

Description

Gets the n\textsuperscript{th} index on the shelf. The index can be passed to other Search Apple events to remove it from the shelf, obtain its title, and so forth.

Apple Event ID

kSearchGetNthIndex (‘fndx’)

Apple Event Parameters

- **kIndexListTag** (‘SilP’), typeLongInteger: An opaque \texttt{void*} representing the shelf, obtained from \texttt{SearchGetIndexList}.
- **kNthIndexTag** (‘Enth’), typeLongInteger: The index to get. The first index on the shelf is index zero.

Return Value

- **kIndexTag** (‘SixP’), typeLongInteger: An opaque \texttt{void*} representing an index. Returns \texttt{NULL} if the n\textsuperscript{th} index is gone.
SearchRemoveIndex

Description
Removes the specified index from the shelf.

Apple Event ID
kSearchRemoveIndex (‘rm dx’)

Apple Event Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kIndexListTag</td>
<td>An opaque void* representing the shelf, obtained from SearchGetIndexList.</td>
</tr>
<tr>
<td>(‘SilP’), typeLongInteger</td>
<td></td>
</tr>
<tr>
<td>kIndexTag</td>
<td>An opaque void* representing the index to be removed. The index may be obtained using SearchGetIndexByPath, SearchGetNthIndex, or SearchAddIndex.</td>
</tr>
<tr>
<td>(‘SixP’), typeLongInteger</td>
<td></td>
</tr>
</tbody>
</table>

Return Value
None
SearchSetIndexFlags

Description
Sets the flags for an index.

Apple Event ID
kSearchSetIndexFlags (‘sfdx’)

Apple Event Parameters

kIndexTag (‘SixP’), typeLongInteger
An opaque void* representing an index.

kFlagTag (‘Flag’), typeLongInteger
Index flags. See the description in SearchGetIndexFlags. In practice, kIndexAvailableFlag should always be set.

Return Value

kFlagTag (‘Flag’), typeLongInteger
Index flags. See the description in SearchGetIndexFlags. This value is returned because it is possible for a request to set a flag to fail.
The Search plug-in adds a new menu, menu items, and toolbar buttons to the Acrobat application.

**Menu Names**

The Search plug-in adds the following menu to Acrobat.

<table>
<thead>
<tr>
<th>Menu name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcroSrch:ToolsSubMenu</td>
<td>Acrobat Search submenu of Edit menu.</td>
</tr>
</tbody>
</table>

**Menu Item Names**

The Search plug-in adds the following menu items to Acrobat.

<table>
<thead>
<tr>
<th>Menu item name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcroSrch:Query</td>
<td>Displays the Search dialog.</td>
</tr>
<tr>
<td>AcroSrch:Indexes</td>
<td>Displays the Index dialog.</td>
</tr>
<tr>
<td>AcroSrch:Results</td>
<td>Displays the Results dialog.</td>
</tr>
<tr>
<td>AcroSrch:Assist</td>
<td>Displays the Word Assistant dialog.</td>
</tr>
<tr>
<td>AcroSrch:Separator</td>
<td>A separator item in the Search tools menu.</td>
</tr>
<tr>
<td>AcroSrch:PrevDoc</td>
<td>Goes to the previous document in the hit list.</td>
</tr>
<tr>
<td>AcroSrch:PrevHit</td>
<td>Goes to the previous hit in the hit list.</td>
</tr>
<tr>
<td>AcroSrch:NextHit</td>
<td>Goes to the next hit in the hit list.</td>
</tr>
<tr>
<td>AcroSrch:NextDoc</td>
<td>Goes to the next document in the hit list.</td>
</tr>
</tbody>
</table>
# Toolbar Button Names

The Search plug-in adds the following buttons to the Acrobat toolbar

<table>
<thead>
<tr>
<th>Button name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcroSrch:Separator</td>
<td>Separator (not visible).</td>
</tr>
<tr>
<td>AcroSrch:Query</td>
<td>Displays the Acrobat Search plug-in's query dialog.</td>
</tr>
<tr>
<td>AcroSrch:Results</td>
<td>Displays the Acrobat Search plug-in's search results dialog.</td>
</tr>
<tr>
<td>AcroSrch:Prev</td>
<td>Goes to the previous hit in the Acrobat Search plug-in's results list.</td>
</tr>
<tr>
<td>AcroSrch:Next</td>
<td>Goes to the next hit in the Acrobat Search plug-in's results list.</td>
</tr>
</tbody>
</table>