# Contents

**Preface** ............................................................................................................................................... 12
- What’s in this guide? ............................................................................................................................ 12
- Who should read this guide? ................................................................................................................ 12
- Related documentation ......................................................................................................................... 12

## 1 OLE Automation ................................................................................................................................. 14

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcroExch.App</td>
<td>14</td>
</tr>
<tr>
<td>CloseAllDocs</td>
<td>16</td>
</tr>
<tr>
<td>Exit</td>
<td>16</td>
</tr>
<tr>
<td>GetActiveDoc</td>
<td>17</td>
</tr>
<tr>
<td>GetActiveTool</td>
<td>17</td>
</tr>
<tr>
<td>GetAVDoc</td>
<td>18</td>
</tr>
<tr>
<td>GetFrame</td>
<td>18</td>
</tr>
<tr>
<td>GetInterface</td>
<td>19</td>
</tr>
<tr>
<td>GetLanguage</td>
<td>19</td>
</tr>
<tr>
<td>GetNumAVDocs</td>
<td>20</td>
</tr>
<tr>
<td>GetPreference</td>
<td>20</td>
</tr>
<tr>
<td>GetPreferenceEx</td>
<td>21</td>
</tr>
<tr>
<td>Hide</td>
<td>21</td>
</tr>
<tr>
<td>Lock</td>
<td>21</td>
</tr>
<tr>
<td>Minimize</td>
<td>22</td>
</tr>
<tr>
<td>Maximize</td>
<td>23</td>
</tr>
<tr>
<td>MenuItemExecute</td>
<td>23</td>
</tr>
<tr>
<td>MenuItemIsEnabled</td>
<td>24</td>
</tr>
<tr>
<td>MenuItemIsMarked</td>
<td>24</td>
</tr>
<tr>
<td>MenuItemRemove</td>
<td>25</td>
</tr>
<tr>
<td>Restore</td>
<td>25</td>
</tr>
<tr>
<td>SetActiveTool</td>
<td>26</td>
</tr>
<tr>
<td>setFrame</td>
<td>26</td>
</tr>
<tr>
<td>SetPreference</td>
<td>27</td>
</tr>
<tr>
<td>SetPreferenceEx</td>
<td>27</td>
</tr>
<tr>
<td>Show</td>
<td>28</td>
</tr>
<tr>
<td>ToolButtonIsEnabled</td>
<td>28</td>
</tr>
<tr>
<td>ToolButtonRemove</td>
<td>29</td>
</tr>
<tr>
<td>Unlock</td>
<td>29</td>
</tr>
<tr>
<td>UnlockEx</td>
<td>30</td>
</tr>
<tr>
<td>AcroExch.AVDoc</td>
<td>30</td>
</tr>
<tr>
<td>BringToFront</td>
<td>31</td>
</tr>
<tr>
<td>ClearSelection</td>
<td>32</td>
</tr>
<tr>
<td>Close</td>
<td>32</td>
</tr>
<tr>
<td>FindText</td>
<td>33</td>
</tr>
<tr>
<td>GetAVPageView</td>
<td>33</td>
</tr>
<tr>
<td>GetFrame</td>
<td>34</td>
</tr>
<tr>
<td>GetPDDoc</td>
<td>34</td>
</tr>
<tr>
<td>getTitle</td>
<td>35</td>
</tr>
</tbody>
</table>
1 OLE Automation (Continued)

AcroExch.AVDoc (Continued)
- GetViewMode ........................................35
- IsValid ..................................................35
- Maximize ...............................................36
- Open .......................................................36
- OpenInWindow .......................................37
- OpenInWindowEx ...................................38
- PrintPages ..............................................40
- PrintPagesEx ..........................................40
- PrintPagesSilent ......................................41
- PrintPagesSilentEx .................................42
- SetFrame ...............................................43
- SetTextSelection .....................................44
- SetTitle ..................................................45
- SetViewMode ..........................................45
- ShowTextSelect .......................................46

AcroExch.AVPPageView ................................47
- DevicePointToPage ................................47
- DoGoBack ..............................................48
- DoGoForward ........................................48
- GetAperture ..........................................49
- GetAVDoc ..............................................49
- GetDoc ..................................................49
- GetPage ................................................49
- GetPageNum ..........................................50
- GetZoom ...............................................50
- GetZoomType .........................................51
- Goto ......................................................51
- PointToDevice .......................................52
- ReadPageDown .......................................53
- ReadPageUp ..........................................53
- ScrollTo ................................................54
- ZoomTo .................................................55

AcroExch.HiliteList ..................................56
- Add .......................................................56

AcroExch.PDAnnot ....................................56
- GetColor ..............................................57
- GetContents .........................................58
- GetDate ................................................58
- GetRect ...............................................58
- GetSubtype ..........................................59
- GetTitle ................................................59
- IsEqual ................................................60
- IsOpen ..................................................60
- IsValid ..................................................61
- Perform .................................................62
- SetColor ..............................................62
- SetContents .........................................63
1 OLE Automation (Continued)

AcroExch.PDAnnot (Continued)
- SetDate .............................................................................................................................................................................63
- SetOpen ..............................................................................................................................................................................64
- SetRect ..............................................................................................................................................................................64
- SetTitle ............................................................................................................................................................................65

AcroExch.PDBookmark ..........................................................................................................................................................66
- Destroy ..................................................................................................................................................................................66
- GetByTitle ...........................................................................................................................................................................67
- getTitle ..................................................................................................................................................................................67
- IsValid ................................................................................................................................................................................68
- Perform ................................................................................................................................................................................68
- SetTitle ............................................................................................................................................................................69

AcroExch.PDPage ..................................................................................................................................................................69
- AcquirePage .........................................................................................................................................................................71
- ClearFlags ...........................................................................................................................................................................71
- Close ...................................................................................................................................................................................72
- Create ..................................................................................................................................................................................72
- CreateTextSelect ..................................................................................................................................................................73
- CreateThumbs ........................................................................................................................................................................74
- CropPages ...........................................................................................................................................................................74
- DeletePages ..........................................................................................................................................................................75
- DeleteThumbs .........................................................................................................................................................................75
- GetFileName .........................................................................................................................................................................76
- GetFlags ...............................................................................................................................................................................76
- GetInfo ................................................................................................................................................................................77
- GetInstanceID ......................................................................................................................................................................77
- GetXObject ...........................................................................................................................................................................78
- GetNumPages .......................................................................................................................................................................78
- GetPageMode .........................................................................................................................................................................79
- GetPermanentID ....................................................................................................................................................................79
- InsertPages ...........................................................................................................................................................................79
- MovePage ...............................................................................................................................................................................80
- Open ....................................................................................................................................................................................81
- OpenAVDoc ...........................................................................................................................................................................81
- ReplacePages .........................................................................................................................................................................82
- Save .....................................................................................................................................................................................83
- SetFlags ................................................................................................................................................................................84
- SetInfo ..................................................................................................................................................................................85
- SetPageMode .........................................................................................................................................................................85

AcroExch.PDDoc .....................................................................................................................................................................69
- AcquirePage .........................................................................................................................................................................71
- ClearFlags ...........................................................................................................................................................................71
- Close ...................................................................................................................................................................................72
- Create ..................................................................................................................................................................................72
- CreateTextSelect ..................................................................................................................................................................73
- CreateThumbs ........................................................................................................................................................................74
- CropPages ...........................................................................................................................................................................74
- DeletePages ..........................................................................................................................................................................75
- DeleteThumbs .........................................................................................................................................................................75
- GetFileName .........................................................................................................................................................................76
- GetFlags ...............................................................................................................................................................................76
- GetInfo ................................................................................................................................................................................77
- GetInstanceID ......................................................................................................................................................................77
- GetXObject ...........................................................................................................................................................................78
- GetNumPages .......................................................................................................................................................................78
- GetPageMode .........................................................................................................................................................................79
- GetPermanentID ....................................................................................................................................................................79
- InsertPages ...........................................................................................................................................................................79
- MovePage ...............................................................................................................................................................................80
- Open ....................................................................................................................................................................................81
- OpenAVDoc ...........................................................................................................................................................................81
- ReplacePages .........................................................................................................................................................................82
- Save .....................................................................................................................................................................................83
- SetFlags ................................................................................................................................................................................84
- SetInfo ..................................................................................................................................................................................85
- SetPageMode .........................................................................................................................................................................85

AcroExch.PDPage ..................................................................................................................................................................86
- AddAnnot ..............................................................................................................................................................................87
- AddNewAnnot .......................................................................................................................................................................88
- CopyToClipboard ..................................................................................................................................................................88
- CreatePageHilite .................................................................................................................................................................89
- CreateWordHilite ...............................................................................................................................................................90
- CropPage .............................................................................................................................................................................91
- Draw ...................................................................................................................................................................................91
- DrawEx ...............................................................................................................................................................................92
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetAnnot</td>
<td>93</td>
</tr>
<tr>
<td>GetAnnotIndex</td>
<td>93</td>
</tr>
<tr>
<td>GetDoc</td>
<td>94</td>
</tr>
<tr>
<td>GetNumAnnots</td>
<td>94</td>
</tr>
<tr>
<td>GetNumber</td>
<td>95</td>
</tr>
<tr>
<td>GetRotate</td>
<td>95</td>
</tr>
<tr>
<td>GetSize</td>
<td>96</td>
</tr>
<tr>
<td>RemoveAnnot</td>
<td>96</td>
</tr>
<tr>
<td>SetRotate</td>
<td>97</td>
</tr>
<tr>
<td>AcroExch.PDTextSelect</td>
<td>98</td>
</tr>
<tr>
<td>Destroy</td>
<td>98</td>
</tr>
<tr>
<td>GetBoundingRect</td>
<td>99</td>
</tr>
<tr>
<td>GetNumText</td>
<td>99</td>
</tr>
<tr>
<td>GetPage</td>
<td>100</td>
</tr>
<tr>
<td>GetText</td>
<td>101</td>
</tr>
<tr>
<td>AcroExch.Point</td>
<td>102</td>
</tr>
<tr>
<td>X</td>
<td>102</td>
</tr>
<tr>
<td>Y</td>
<td>102</td>
</tr>
<tr>
<td>AcroExch.Rect</td>
<td>102</td>
</tr>
<tr>
<td>Bottom</td>
<td>103</td>
</tr>
<tr>
<td>Left</td>
<td>103</td>
</tr>
<tr>
<td>Right</td>
<td>103</td>
</tr>
<tr>
<td>Top</td>
<td>104</td>
</tr>
<tr>
<td>AcroExch.Time</td>
<td>104</td>
</tr>
<tr>
<td>Date</td>
<td>104</td>
</tr>
<tr>
<td>Hour</td>
<td>105</td>
</tr>
<tr>
<td>Millisecond</td>
<td>105</td>
</tr>
<tr>
<td>Minute</td>
<td>105</td>
</tr>
<tr>
<td>Month</td>
<td>105</td>
</tr>
<tr>
<td>Second</td>
<td>106</td>
</tr>
<tr>
<td>Year</td>
<td>106</td>
</tr>
<tr>
<td>AxAcroPDFLib.AxAcroPDF</td>
<td>106</td>
</tr>
<tr>
<td>GetVersions</td>
<td>108</td>
</tr>
<tr>
<td>GoBackwardalStack</td>
<td>108</td>
</tr>
<tr>
<td>GoForwardStack</td>
<td>108</td>
</tr>
<tr>
<td>GotoFirstPage</td>
<td>108</td>
</tr>
<tr>
<td>GotoLastPage</td>
<td>108</td>
</tr>
<tr>
<td>GotoNextPage</td>
<td>109</td>
</tr>
<tr>
<td>GotoPreviousPage</td>
<td>109</td>
</tr>
<tr>
<td>LoadFile</td>
<td>109</td>
</tr>
<tr>
<td>Print</td>
<td>110</td>
</tr>
<tr>
<td>PrintAll</td>
<td>110</td>
</tr>
<tr>
<td>PrintAllFit</td>
<td>111</td>
</tr>
<tr>
<td>PrintPages</td>
<td>111</td>
</tr>
<tr>
<td>PrintPagesFit</td>
<td>112</td>
</tr>
<tr>
<td>PrintWithDialog</td>
<td>113</td>
</tr>
<tr>
<td>SetCurrentHighlight</td>
<td>113</td>
</tr>
</tbody>
</table>
2 DDE Messages ........................................................................................................................................................................... 120
AppExit .................................................................................................................................................................................. 121
AppHide ................................................................................................................................................................................. 122
AppShow ............................................................................................................................................................................. 122
CloseAllDocs .................................................................................................................................................................... 122
DocClose ........................................................................................................................................................................ 123
DocDeletePages .................................................................................................................................................................. 123
DocFind .............................................................................................................................................................................. 124
DocGoTo ........................................................................................................................................................................ 125
DocGoToNameDest ......................................................................................................................................................... 125
DocInsertPages ............................................................................................................................................................. 125
DocOpen ........................................................................................................................................................................ 126
DocPageDown .................................................................................................................................................................. 127
DocPageLeft ................................................................................................................................................................. 127
DocPageRight .............................................................................................................................................................. 127
DocPageUp .................................................................................................................................................................. 128
DocPrint ........................................................................................................................................................................ 128
DocPrint ........................................................................................................................................................................ 129
DocReplacePages ......................................................................................................................................................... 129
DocSave .......................................................................................................................................................................... 130
DocSaveAs ...................................................................................................................................................................... 130
DocScrollTo .................................................................................................................................................................. 131
DocSetViewMode ......................................................................................................................................................... 132
DocZoomTo .................................................................................................................................................................. 132
FileOpen ........................................................................................................................................................................ 133
FileOpenEx ...................................................................................................................................................................... 133
FilePrint ........................................................................................................................................................................ 134
FilePrintEx ..................................................................................................................................................................... 135
FilePrintSilent ............................................................................................................................................................... 135
FilePrintSilentEx ............................................................................................................................................................ 136
FilePrintTo .................................................................................................................................................................. 137
FilePrintToEx .................................................................................................................................................................. 137
FullMenus ..................................................................................................................................................................... 138
HideToolbar ..................................................................................................................................................................... 138
MenuItemExecute ............................................................................................................................................................ 139
ShortMenus ..................................................................................................................................................................... 139
ShowToolbar .................................................................................................................................................................. 140
3 Apple Event Objects and Apple Events................................................................. 141

Objects ...................................................................................................................... 141
  annotation .............................................................................................................. 141
  application .......................................................................................................... 143
  AVPageView ........................................................................................................ 145
  bookmark ............................................................................................................. 145
  conversion .......................................................................................................... 147
  document ............................................................................................................ 147
  EPS Conversion .................................................................................................. 149
  Link Annotation ................................................................................................ 149
  menu ..................................................................................................................... 149
  menu item ........................................................................................................... 150
  page ..................................................................................................................... 151
  PDAnnot ............................................................................................................. 151
  PDBookMark ..................................................................................................... 152
  PDLinkAnnot ..................................................................................................... 152
  PDPage ............................................................................................................... 152
  PDTextAnnot ..................................................................................................... 152
  PDF Window ....................................................................................................... 152
  PostScript Conversion ....................................................................................... 153
  Text Annotation ................................................................................................ 154

Required suite events ............................................................................................. 155
  open ...................................................................................................................... 155
  print ...................................................................................................................... 155
  quit ....................................................................................................................... 155
  run ......................................................................................................................... 156

Core suite events .................................................................................................... 156
  close ...................................................................................................................... 156
  count ..................................................................................................................... 157
  delete .................................................................................................................... 157
  exists .................................................................................................................... 158
  get ......................................................................................................................... 158
  make ..................................................................................................................... 159
  move ...................................................................................................................... 159
  open ....................................................................................................................... 160
  quit ....................................................................................................................... 160
  save ...................................................................................................................... 161
  set ......................................................................................................................... 161

Acrobat application events.................................................................................... 162
  bring to front .................................................. 163
  clear selection ........................................... 163
  close all docs ........................................... 163
  create thumbs .......................................... 164
  delete pages ............................................ 165
  delete thumbs .......................................... 165
  execute ..................................................... 166
  find next note ......................................... 166
  find text .................................................. 167
3 Apple Event Objects and Apple Events (Continued)
Acrobat application events (Continued)

- get info ............................................................................................................................................................................. 168
- go backward ..................................................................................................................................................................... 168
- go forward ......................................................................................................................................................................... 169
- goto .................................................................................................................................................................................. 170
- goto next ........................................................................................................................................................................... 170
- goto previous ................................................................................................................................................................. 171
- insert pages ...................................................................................................................................................................... 172
- is toolbutton enabled ......................................................................................................................................................... 172
- maximize .............................................................................................................................................................................. 173
- perform .................................................................................................................................................................................. 174
- print pages ........................................................................................................................................................................... 174
- read page down ................................................................................................................................................................. 175
- read page up ......................................................................................................................................................................... 175
- remove toolbutton .............................................................................................................................................................. 176
- replace pages ...................................................................................................................................................................... 176
- scroll .................................................................................................................................................................................... 177
- select text .............................................................................................................................................................................. 178
- set info ................................................................................................................................................................................ 179
- zoom ..................................................................................................................................................................................... 179

Miscellaneous events .............................................................................................................................................................. 180
- do script ................................................................................................................................................................................. 180

4 Acrobat Catalog Plug-In ................................................................................................................................................................. 181
Catalog Windows messages ......................................................................................................................................................... 181
Catalog DDE methods ................................................................................................................................................................... 181
- AppExit ................................................................................................................................................................................ 181
- AppFront ............................................................................................................................................................................... 181
- FileBuild .............................................................................................................................................................................. 182
- FileOpen ............................................................................................................................................................................... 182
- FilePurge .............................................................................................................................................................................. 182

5 Acrobat Forms Plug-In ................................................................................................................................................................. 183
Forms plug-in OLE automation ................................................................................................................................................... 183
Exceptions .................................................................................................................................................................................. 183
- AFormApp ........................................................................................................................................................................... 184
- Field ...................................................................................................................................................................................... 184
Methods ..................................................................................................................................................................................... 184
- PopulateListOrComboBox ......................................................................................................................................................... 185
- SetBackgroundColor ............................................................................................................................................................. 185
- SetBorderColor ................................................................................................................................................................... 186
- SetButtonCaption ................................................................................................................................................................. 187
- SetButtonIcon ..................................................................................................................................................................... 187
- SetExportValues ................................................................................................................................................................ 188
- SetForegroundColor ............................................................................................................................................................ 189
- SetJavaScriptAction ............................................................................................................................................................. 190
- SetResetFormAction ........................................................................................................................................................ 191
- SetSubmitFormAction ......................................................................................................................................................... 192
5 Acrobat Forms Plug-In (Continued)

Field (Continued)

Properties ....................................................................................................................................................... 192
   Alignment ...................................................................................................................................................... 193
   BorderStyle .................................................................................................................................................. 194
   BorderWidth ................................................................................................................................................. 194
   ButtonLayout ............................................................................................................................................... 194
   CalcOrderIndex .......................................................................................................................................... 195
   CharLimit ..................................................................................................................................................... 196
   DefaultValue ............................................................................................................................................... 196
   Editable ....................................................................................................................................................... 196
   Highlight ..................................................................................................................................................... 197
   IsHidden ..................................................................................................................................................... 197
   IsMultiline .................................................................................................................................................. 198
   IsPassword .................................................................................................................................................. 198
   IsReadOnly ................................................................................................................................................. 199
   IsRequired .................................................................................................................................................. 199
   IsTerminal ................................................................................................................................................ 199
   Name .......................................................................................................................................................... 199
   NoViewFlag .............................................................................................................................................. 199
   PrintFlag ................................................................................................................................................... 200
   Style ........................................................................................................................................................... 200
   TextFont .................................................................................................................................................... 201
   TextSize .................................................................................................................................................... 201
   Type ............................................................................................................................................................ 201
   Value .......................................................................................................................................................... 202

Fields ......................................................................................................................................................... 202

Methods ...................................................................................................................................................... 203
   Add ............................................................................................................................................................... 203
   AddDocJavascript ...................................................................................................................................... 204
   ExecuteThisJavascript .............................................................................................................................. 205
   ExportAsFDF ........................................................................................................................................... 205
   ExportAsHtml ........................................................................................................................................... 206
   ImportAnFDF ........................................................................................................................................... 207
   Remove ..................................................................................................................................................... 207

Properties ................................................................................................................................................ 207
   Count ......................................................................................................................................................... 207
   Item ............................................................................................................................................................ 207
   _NewEnum ............................................................................................................................................. 208

6 Acrobat Search Plug-in ............................................................................................................................. 209

Search plug-in using DDE .......................................................................................................................... 209

Simple query item ....................................................................................................................................... 209

Query item .................................................................................................................................................. 209

Query options .......................................................................................................................................... 210

Query language type constants ............................................................................................................ 211

Word option bit-flag constants ........................................................................................................... 211

Manipulating indexes through DDE ...................................................................................................... 212

Options .................................................................................................................................................... 212

Index operation selectors ..................................................................................................................... 212
6 Acrobat Search Plug-in (Continued)

Search plug-in using Apple events ................................................................. 213
  SearchAddIndex ......................................................................................... 213
  SearchCountIndexList ................................................................................. 213
  SearchDoQuery .......................................................................................... 214
  SearchGetIndexByPath .............................................................................. 215
  SearchGetIndexFlags ................................................................................ 216
  SearchGetIndexList ................................................................................... 216
  SearchGetIndexPath ................................................................................... 216
  SearchGetIndexTitle .................................................................................. 217
  SearchGetNthIndex .................................................................................... 217
  SearchRemoveIndex ................................................................................... 218
  SearchSetIndexFlags ................................................................................ 218

Search lists .................................................................................................. 219
  Menu names .............................................................................................. 219
  Menu item names ...................................................................................... 219
  Toolbar button names .............................................................................. 220

7 Coordinate Systems ................................................................................. 221

User space .................................................................................................. 221
Device space .............................................................................................. 222

Index ........................................................................................................... 223
Preface

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

What’s in this guide?

This document provides a detailed reference of all the calls needed for OLE, DDE, and Apple events.

There is no IAC support for the UNIX® versions of Acrobat. There is no IAC support in the Japanese version of Acrobat.

Who should read this guide?

This guide is for developers that want to communicate with Acrobat from another application or render Adobe PDF files in their own application, or who are writing plug-ins that need to communicate with or use multiple applications.

You should already be familiar with at least one of OLE, DDE, Apple events, or AppleScript. 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 documentation

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>A guide to the documentation in the Acrobat SDK</td>
<td>Acrobat SDK Documentation Roadmap</td>
</tr>
<tr>
<td>A guide to the sections of the Acrobat SDK that pertain to Adobe Reader®</td>
<td>Developing for Adobe Reader</td>
</tr>
<tr>
<td>A guide to the sample code included with the Acrobat SDK</td>
<td>Guide to SDK Samples</td>
</tr>
<tr>
<td>Prototyping code without the overhead of writing and verifying a complete plug-in or application</td>
<td>Snippet Runner Cookbook</td>
</tr>
<tr>
<td>Using DDE, OLE, Apple events, and AppleScript to control Acrobat and Adobe Reader and to render PDF documents</td>
<td>Developing Applications Using Interapplication Communication</td>
</tr>
<tr>
<td>Using JavaScript™ to develop and enhance standard workflows in Acrobat and Adobe Reader</td>
<td>Developing Acrobat Applications Using JavaScript</td>
</tr>
<tr>
<td>Detailed descriptions of JavaScript APIs for developing and enhancing workflows in Acrobat and Adobe Reader</td>
<td>JavaScript for Acrobat API Reference</td>
</tr>
<tr>
<td>For information about</td>
<td>See</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----</td>
</tr>
<tr>
<td>A detailed description of the PDF file format</td>
<td><em>PDF Reference</em></td>
</tr>
<tr>
<td>Developing plug-ins for Acrobat and Adobe Reader, as well as for PDF Library applications</td>
<td><em>Developing Plug-ins and Applications</em></td>
</tr>
<tr>
<td>Detailed descriptions of the APIs for Acrobat and Adobe Reader plug-ins, as well as for PDF Library applications</td>
<td><em>Acrobat and PDF Library API Reference</em></td>
</tr>
</tbody>
</table>
This chapter describes the objects, data types, and methods in the OLE automation interface.

The names `AcroExch.App` and `AxAcroPDFLib.AxAcroPDF` are the external strings OLE clients use to create objects of certain types. The Acrobat developer type libraries call them `CAcro.App` and `AcroPDFLib`, respectively.

Acrobat supports dual interfaces, so the methods all have a return type of `HResult`.

The following table summarizes the available objects and data types.

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AcroExch.App</code></td>
<td>The application itself.</td>
</tr>
<tr>
<td><code>AcroExch.AVDoc</code></td>
<td>A document as seen in the user interface.</td>
</tr>
<tr>
<td><code>AcroExch.PDDoc</code></td>
<td>The underlying PDF representation of a document.</td>
</tr>
<tr>
<td><code>AcroExch.HiliteList</code></td>
<td>An entry in a highlight list.</td>
</tr>
<tr>
<td><code>AcroExch.AVPageView</code></td>
<td>The area of the window that displays the contents of a page.</td>
</tr>
<tr>
<td><code>AcroExch.PDPage</code></td>
<td>A single page in the PDF representation of a document.</td>
</tr>
<tr>
<td><code>AcroExch.PDAnnot</code></td>
<td>An annotation on a page in the PDF file.</td>
</tr>
<tr>
<td><code>AcroExch.PDBookmark</code></td>
<td>A bookmark in a PDF file.</td>
</tr>
<tr>
<td><code>AcroExch.PDTextSelect</code></td>
<td>A selection of text on a single page.</td>
</tr>
<tr>
<td><code>AxAcroPDFLib.AxAcroPDF</code></td>
<td>An object containing PDF browser controls.</td>
</tr>
<tr>
<td><code>AcroExch.Point</code></td>
<td>A point, specified by its x–coordinate and y–coordinate.</td>
</tr>
<tr>
<td><code>AcroExch.Rect</code></td>
<td>A rectangle, specified by the top-left and bottom-right points.</td>
</tr>
<tr>
<td><code>AcroExch.Time</code></td>
<td>A specified time, accurate to the millisecond.</td>
</tr>
</tbody>
</table>

**AcroExch.App**

The Acrobat application itself. This is a creatable interface. From the application layer, you can 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 (through an `AVDoc` object).
Methods

The **App** object has the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CloseAllDocs</strong></td>
<td>Closes all open documents.</td>
</tr>
<tr>
<td><strong>Exit</strong></td>
<td>Exits Acrobat.</td>
</tr>
<tr>
<td><strong>GetActiveDoc</strong></td>
<td>Gets the frontmost document.</td>
</tr>
<tr>
<td><strong>GetActiveTool</strong></td>
<td>Gets the name of the currently active tool.</td>
</tr>
<tr>
<td><strong>GetAVDoc</strong></td>
<td>Gets an <code>AcroExch.AVDoc</code> object via its index within the list of open <code>AVDoc</code> objects.</td>
</tr>
<tr>
<td><strong>GetFrame</strong></td>
<td>Gets the window’s frame.</td>
</tr>
<tr>
<td><strong>GetInterface</strong></td>
<td>Gets an <code>IDispatch</code> interface for a named object, typically a third-party plug-in.</td>
</tr>
<tr>
<td><strong>GetLanguage</strong></td>
<td>Gets a code that specifies which language the Acrobat application’s user interface is using.</td>
</tr>
<tr>
<td><strong>GetNumAVDocs</strong></td>
<td>Gets the number of open <code>AcroExch.AVDoc</code> objects.</td>
</tr>
<tr>
<td><strong>GetPreference</strong></td>
<td>Gets a value from the preferences file.</td>
</tr>
<tr>
<td><strong>GetPreferenceEx</strong></td>
<td>Gets the specified application preference, using the VARIANT type to pass values.</td>
</tr>
<tr>
<td><strong>Hide</strong></td>
<td>Hides the Acrobat application.</td>
</tr>
<tr>
<td><strong>Lock</strong></td>
<td>Locks the Acrobat application.</td>
</tr>
<tr>
<td><strong>Minimize</strong></td>
<td>Minimizes the Acrobat application.</td>
</tr>
<tr>
<td><strong>Maximize</strong></td>
<td>Maximizes the Acrobat application.</td>
</tr>
<tr>
<td><strong>MenuItemExecute</strong></td>
<td>Executes the menu item whose language-independent menu item name is specified.</td>
</tr>
<tr>
<td><strong>MenuItemisEnabled</strong></td>
<td>Determines whether the specified menu item is enabled.</td>
</tr>
<tr>
<td><strong>MenuItemIsMarked</strong></td>
<td>Determines whether the specified menu item is marked.</td>
</tr>
<tr>
<td><strong>MenuItemRemove</strong></td>
<td>Removes the menu item whose language-independent menu item is specified.</td>
</tr>
<tr>
<td><strong>Restore</strong></td>
<td>Restores the main window of the Acrobat application.</td>
</tr>
<tr>
<td><strong>SetActiveTool</strong></td>
<td>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).</td>
</tr>
<tr>
<td><strong>SetFrame</strong></td>
<td>Sets the window’s frame to the specified rectangle.</td>
</tr>
</tbody>
</table>
CloseAllDocs

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

You must explicitly close all documents or call App.CloseAllDocs. Otherwise, the process never exits.

Syntax

VARIANT_BOOL CloseAllDocs();

Returns

-1 if successful, 0 if not.

Related methods

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

Exit

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

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

```c
VARIANT_BOOL Exit();
```

Returns

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 continues running. This method does 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 can call Hide and then Exit.

Related methods

App. `CloseAllDocs`

**GetActiveDoc**

Gets the frontmost document.

Syntax

```c
LPDISPATCH GetActiveDoc();
```

Returns

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

Related methods

App. `GetAVDoc`

**GetActiveTool**

Gets the name of the currently active tool.

Syntax

```c
BSTR GetActiveTool();
```

Returns

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

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

Syntax

LPDISPATCH GetAVDoc(long nIndex);

Parameters

<table>
<thead>
<tr>
<th>nIndex</th>
<th>The index of the document to get.</th>
</tr>
</thead>
</table>

Returns

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

Gets the window's frame.

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.

This method is also not useful if the Acrobat application is in single document interface (SDI) mode.

Syntax

LPDISPATCH GetFrame();

Returns

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

If the Acrobat application is in SDI mode, a [0,0,0,0] Rect is returned.

Related methods

App.Maximize
App.SetFrame
GetInterface

Gets an IDispatch interface for a named object, typically a third-party plug-in. This is an entry point to functionality that is undefined 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.

Syntax

LPDISPATCH GetInterface (BSTR szName);

Parameters

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

Returns

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

GetLanguage

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

Syntax

BSTR GetLanguage();

Returns

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

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.

Syntax

long GetNumAVDocs();

Returns

The number of open AcroExch.AVDoc objects.

Related methods

App.GetActiveDoc
App.GetAVDoc

GetPreference

**Note:** This method is 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.

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.

Syntax

long GetPreference(short nType);

Parameters

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

Returns

The value of the specified preference item.

Related methods

App.GetLanguage
App.SetPreference
GetPreferenceEx

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

Syntax

VARIANT GetPreferenceEx(short nType);

Parameters

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

Returns

The value of the specified preference item.

Related methods

App. GetLanguage
App. SetPreferenceEx

Hide

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.

Syntax

VARIANT_BOOL Hide();

Returns

-1 if successful, 0 if not.

Related methods

App. Show

Lock

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 consider 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.

**Syntax**

```c
VARIANT_BOOL Lock(BSTR szLockedBy);
```

**Parameters**

<table>
<thead>
<tr>
<th>szLockedBy</th>
<th>A string that is used as the name of the application that has locked the Acrobat application.</th>
</tr>
</thead>
</table>

**Returns**

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

**Related methods**

App.UnlockEx

**Minimize**

Minimizes the Acrobat application.

**Syntax**

```c
VARIANT_BOOL Minimize(long BMinimize);
```

**Parameters**

<table>
<thead>
<tr>
<th>BMinimize</th>
<th>If a positive number, the Acrobat application is minimized. If 0, the Acrobat application is returned to its normal state.</th>
</tr>
</thead>
</table>

**Returns**

-1 if successful, 0 if not.

**Related methods**

App.GetFrame
App.SetFrame
Maximize

Maximizes the Acrobat application.

Syntax

VARIANT_BOOL Maximize(long bMaximize);

Parameters

bMaximize

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

Returns

-1 if successful, 0 if not.

Related methods

App.GetFrame
App.SetFrame

MenuItemExecute

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

Syntax

VARIANT_BOOL MenuItemExecute(BSTR szMenuItemName);

Parameters

szMenuItemName

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.

Returns

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

Related methods

App.MenuItemIsEnabled
App.MenuItemIsMarked
App.MenuItemRemove
MenuITEMIsEnabled

Determines whether the specified menu item is enabled.

Syntax

```c
VARIANT_BOOL MenuITEMIsEnabled(BSTR szMenuITEMName);
```

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.

Returns

- `-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

Determines whether the specified menu item is marked.

Syntax

```c
VARIANT_BOOL MenuITEMIsMarked(BSTR szMenuITEMName);
```

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.

Returns

- `-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**

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

**Syntax**

VARIANT_BOOL MenuItemRemove(BSTR szMenuItemName);

**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.

**Returns**

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

**Related methods**

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

**Restore**

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 to become active.

**Syntax**

VARIANT_BOOL Restore(long bRestore);

**Parameters**

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

**Returns**

-1 if successful, 0 if not.

**Related methods**

App.**GetFrame**  
App.**SetFrame**
SetActiveTool

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).

Syntax

VARIANT_BOOL SetActiveTool(BSTR szButtonName, long bPersistent);

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>

Returns

-1 if the tool was set, 0 otherwise.

Related methods

App.GetActiveTool
App.ToolButtonIsEnabled
App.ToolButtonRemove

SetFrame

Sets the window’s frame to the specified rectangle. This method has no effect if the Acrobat application is in single document interface (SDI) mode.

Syntax

VARIANT_BOOL SetFrame(LPDISPATCH iAcroRect);

Parameters

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

Returns

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

**Note:** This method is 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.

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.

**Syntax**

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

**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>nValue</td>
<td>The value to set.</td>
</tr>
</tbody>
</table>

**Returns**

-1 if successful, 0 if not.

**Related methods**

- App. `GetFrame`
- App. `Maximize`

SetPreferenceEx

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

**Syntax**

```c
VARIANT_BOOL SetPreferenceEx(short nType, VARIANT* pVal);
```
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 <em>Acrobat and PDF Library API Reference</em> for a list of preference items.</td>
</tr>
<tr>
<td>pVal</td>
<td>The value to set.</td>
</tr>
</tbody>
</table>

Returns

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

Related methods

App.*GetLanguage*

App.*GetPreferenceEx*

**Show**

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 no documents are being displayed.

**Syntax**

```
VARIANT_BOOL Show();
```

**Returns**

-1 if successful, 0 if not.

**Related methods**

App.*Hide*

**ToolButtonIsEnabled**

Determines whether the specified toolbar button is enabled.

**Syntax**

```
VARIANT_BOOL ToolButtonIsEnabled(BSTR szButtonName);
```

**Parameters**

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

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

Related methods

App::GetActiveTool
App::SetActiveTool
App::ToolButtonRemove

ToolButtonRemove

Removes the specified button from the toolbar.

Syntax

VARIANT_BOOL ToolButtonRemove(BSTR szButtonName);

Parameters

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

Returns

-1 if the button was removed, 0 otherwise.

Related methods

App::GetActiveTool
App::SetActiveTool
App::ToolButtonIsEnabled

Unlock

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

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.

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

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

Syntax

VARIANT_BOOL Unlock();

Returns

-1 if successful, 0 if not.

Related methods

App.Lock
App.UnlockEx

UnlockEx

Unlocks the Acrobat application if it was previously locked.

Syntax

VARIANT_BOOL UnlockEx (BSTR szLockedBy);

Parameters

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

Returns

-1 if successful, 0 if not.

Related methods

App.Lock

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.

Methods

The AVDoc object has the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BringToFront</td>
<td>Brings the window to the front.</td>
</tr>
<tr>
<td>ClearSelection</td>
<td>Clears the current selection.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes a document.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FindText</td>
<td>Finds the specified text, scrolls so that it is visible, and highlights it.</td>
</tr>
<tr>
<td>GetAVPageView</td>
<td>Gets the AcroExch.AVPageView associated with an AcroExch.AVDoc.</td>
</tr>
<tr>
<td>GetFrame</td>
<td>Gets the rectangle specifying the window’s size and location.</td>
</tr>
<tr>
<td>GetPDDoc</td>
<td>Gets the AcroExch.PDDoc associated with an AcroExch.AVDoc.</td>
</tr>
<tr>
<td>GetTitle</td>
<td>Gets the window’s title.</td>
</tr>
<tr>
<td>GetViewMode</td>
<td>Gets the current document view mode (pages only, pages and thumbnails, or pages and bookmarks).</td>
</tr>
<tr>
<td>IsValid</td>
<td>Determines whether the AcroExch.AVDoc is still valid.</td>
</tr>
<tr>
<td>Maximize</td>
<td>Maximizes the window if bMaxSize is a positive number.</td>
</tr>
<tr>
<td>Open</td>
<td>Opens a file.</td>
</tr>
<tr>
<td>OpenInWindow</td>
<td>Opens a PDF file and displays it in a user-specified window.</td>
</tr>
<tr>
<td>OpenInWindowEx</td>
<td>Opens a PDF file and displays it in a user-specified window.</td>
</tr>
<tr>
<td>PrintPages</td>
<td>Prints a specified range of pages displaying a print dialog box.</td>
</tr>
<tr>
<td>PrintPagesEx</td>
<td>Prints a specified range of pages, displaying a print dialog box.</td>
</tr>
<tr>
<td>PrintPagesSilent</td>
<td>Prints a specified range of pages without displaying any dialog box.</td>
</tr>
<tr>
<td>PrintPagesSilentEx</td>
<td>Prints a specified range of pages without displaying any dialog box.</td>
</tr>
<tr>
<td>SetFrame</td>
<td>Sets the window’s size and location.</td>
</tr>
<tr>
<td>SetTextSelection</td>
<td>Sets the document’s selection to the specified text selection.</td>
</tr>
<tr>
<td>SetTitle</td>
<td>Sets the window’s title.</td>
</tr>
<tr>
<td>SetViewMode</td>
<td>Sets the mode in which the document will be viewed (pages only, pages and thumbnails, or pages and bookmarks)</td>
</tr>
<tr>
<td>ShowTextSelect</td>
<td>Changes the view so that the current text selection is visible.</td>
</tr>
</tbody>
</table>

**BringToFront**

Brings the window to the front.

**Syntax**

```c
VARIANT_BOOL BringToFront();
```

**Returns**

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

Clears the current selection.

Syntax

VARIANT_BOOL ClearSelection();

Returns

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

Related methods

AVDoc.SetTextSelection
AVDoc.ShowTextSelect
PDDoc.CreateTextSelect
PDPPage.CreatePageHilite
PDPPage.CreateWordHilite
PDTextSelect.Destroy
PDTextSelect.GetBoundingRect
PDTextSelect.GetNumText
PDTextSelect.GetPage
PDTextSelect.GetText

Close

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.

Syntax

VARIANT_BOOL Close(long bNoSave);

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.
Returns

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

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

Syntax

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

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>

Returns

-1 if the text was found, 0 otherwise.

GetAVPageView

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

Syntax

LPDISPATCH GetAVPageView();
Returns

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

Related methods

AVDoc.GetPDDoc
AVDoc.SetViewMode
AVPageView.GetAVDoc
AVPageView.GetDoc

GetFrame

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

Syntax

LPDISPATCH GetFrame();

Returns

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

Related methods

AVDoc.SetFrame

GetPDDoc

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

Syntax

LPDISPATCH GetPDDoc();

Returns

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

Related methods

AVDoc.GetAVPageView
AVPageView.GetAVDoc
AVPageView.GetDoc
GetTitle

Gets the window's title.

Syntax

BSTR GetTitle();

Returns

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

Related methods

AVDoc. Open
AVDoc. SetTitle
PDDoc. OpenAVDoc

GetViewMode

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

Syntax

long GetViewMode();

Returns

The current document view mode or 0 if no document is open. The return value is one of the following:

- 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

Related methods

AVDoc. GetAVPageView
AVDoc. SetViewMode

IsValid

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

Syntax

VARIANT_BOOL IsValid();
Returns

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

Related methods

App. GetAVDoc
AVPageView. GetAVDoc

Maximize

Maximizes the window if bMaxSize is a positive number.

Syntax

VARIANT_BOOL Maximize(long bMaxSize);

Parameters

bMaxSize Indicates whether the window should be maximized.

Returns

-1 if a document is open, 0 otherwise.

Related methods

AVDoc. GetFrame
AVDoc. SetFrame

Open

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).

Syntax

VARIANT_BOOL Open(BSTR szFullPath, BSTR szTempTitle);

Parameters

| szFullPath   | The full path 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. |
Returns

-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

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

Syntax

VARIANT_BOOL OpenInWindow(BSTR fileName, short hWnd);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>The full path 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>

Returns

-1

Related methods

App. CloseAllDocs
AVDoc. Close
AVDoc. Open
AVDoc. OpenInWindowEx
OpenInWindowEx

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.

**Syntax**

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

**Parameters**

| szFullPath | The full path of the file to open. |
| hWnd | Handle for the window in which the file is displayed. |
| openFlags | Type of window view. Must be one of the following:

  | AV_EXTERNAL_VIEW — Display the AVPageView, scrollbars, toolbar, and bookmark or thumbnails pane. Annotations are active.
  | AV_DOC_VIEW — Display the AVPageView, scrollbars, and bookmark or thumbnails pane. Annotations are active.
  | AV_PAGE_VIEW — Display only the AVPageView (the window that displays the PDF file). Do not display scrollbars, the toolbar, and bookmark or thumbnails pane. Annotations are active.

**Note:** Use either AV_DOC_VIEW or AV_PAGE_VIEW whenever possible. Use AV_EXTERNAL_VIEW only if you do not want the application to display its own toolbar. Use AV_PAGE_VIEW to open the file with no scrollbars and no status window at the bottom of the page.
### Returns

-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

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

**Syntax**

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

**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 can 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.

**Returns**

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

**Related methods**

- `ADoc.PrintPagesEx`
- `ADoc.PrintPagesSilent`
- `ADoc.PrintPagesSilentEx`

PrintPagesEx

Prints a specified range of pages, displaying a print dialog box. *PrintPagesEx* has more parameters than *PrintPages*. *PrintPagesEx* always uses the default printer setting.

**Syntax**

```c
VARIANT_BOOL printPagesEx(long nFirstPage, long nLastPage,
                           long nPSLevel, long bBinaryOk,
                           long bShrinkToFit, long bReverse,
                           long bFarEastFontOpt, long bEmitHalftones,
                           long iPageOption);
```
**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>

**Returns**

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**

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.

**Syntax**

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

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

Syntax

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

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>
</tbody>
</table>

Returns

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

Related methods

AVDoc.PrintPages
AVDoc.PrintPagesEx
AVDoc.PrintPagesSilentEx
SetFrame

Sets the window’s size and location.

Syntax

VARIANT_BOOL SetFrame(LPDISPATCH iAcroRect);

Parameters

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

Returns

Always returns -1.

Related methods

AVDoc.GetFrame

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.

bReverse (PostScript printing only) If a positive number, print the pages in reverse order. If false, print the pages in the regular order.

bFarEastFontOpt (PostScript printing only) Set to a positive number if the destination printer has multibyte fonts; set to 0 otherwise.

bEmitHalftones (PostScript printing only) If a positive number, emit the halftones specified in the document. If 0, do not.

iPageOption Pages in the range to print. Must be one of: PDAllPages, PDEvenPagesOnly, or PDOddPagesOnly.

Returns

0 if there were any exceptions while printing, -1 otherwise.

Related methods

AVDoc.PrintPages
AVDoc.PrintPagesEx
AVDoc.PrintPagesSilentEx
SetTextSelection

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.

**Syntax**

```c
VARIANT_BOOL SetTextSelection(LPDISPATCH iAcroPDTextSelect);
```

**Parameters**

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

**Returns**

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

Sets the window's title.

Syntax

VARIANT_BOOL SetTitle(BSTR szTitle);

Parameters

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

Returns

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

Related methods

AVDoc.Open
AVDoc.OpenAVDoc
PDDoc.OpenAVDoc

SetViewMode

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

Syntax

VARIANT_BOOL SetViewMode(long nType);

Parameters

<table>
<thead>
<tr>
<th>nType</th>
<th>The view mode to be set. Possible values:</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDDontCare</td>
<td>0 — leave the view mode as it is</td>
</tr>
<tr>
<td>PDUseNone</td>
<td>1 — display without bookmarks or thumbnails</td>
</tr>
<tr>
<td>PDUseThumbs</td>
<td>2 — display using thumbnails</td>
</tr>
<tr>
<td>PDUseBookmarks</td>
<td>3 — display using bookmarks</td>
</tr>
</tbody>
</table>

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

Returns

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

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

Syntax

```cpp
VARIANT_BOOL ShowTextSelect();
```

Returns

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

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.

Methods

The AVPageView object has the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DevicePointToPage</td>
<td>Converts the coordinates of a point from device space to user space.</td>
</tr>
<tr>
<td>DoGoBack</td>
<td>Goes to the previous view on the view history stack, if any.</td>
</tr>
<tr>
<td>DoGoForward</td>
<td>Goes to the next view on the view history stack, if any.</td>
</tr>
<tr>
<td>GetAperture</td>
<td>Gets the aperture of the specified page view.</td>
</tr>
<tr>
<td>GetAVDoc</td>
<td>Gets the AcroExch.AVDoc associated with the current page.</td>
</tr>
<tr>
<td>GetDoc</td>
<td>Gets the AcroExch.PDDoc corresponding to the current page.</td>
</tr>
<tr>
<td>GetPage</td>
<td>Gets the AcroExch.PDPage corresponding to the current page.</td>
</tr>
<tr>
<td>GetPageNum</td>
<td>Gets the page number of the current page.</td>
</tr>
<tr>
<td>GetZoom</td>
<td>Gets the current zoom factor, specified as a percent.</td>
</tr>
<tr>
<td>GetZoomType</td>
<td>Gets the current zoom type.</td>
</tr>
<tr>
<td>Goto</td>
<td>Goes to the specified page.</td>
</tr>
<tr>
<td>PointToDevice</td>
<td>Deprecated. Converts the coordinates of a point from user space to device space.</td>
</tr>
<tr>
<td>ReadPageDown</td>
<td>Scrolls forward through the document by one screen area.</td>
</tr>
<tr>
<td>ReadPageUp</td>
<td>Scrolls backward through the document by one screen area.</td>
</tr>
<tr>
<td>ScrollTo</td>
<td>Scrolls to the specified location on the current page.</td>
</tr>
<tr>
<td>ZoomTo</td>
<td>Zooms to the specified magnification.</td>
</tr>
</tbody>
</table>

DevicePointToPage

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

Syntax

LPDISPATCH DevicePointToPage(LPDISPATCH iAcroPoint);
Parameters

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

Returns

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

Related methods

AVPageView.PointToDevice

DoGoBack

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

Syntax

VARIANT_BOOL DoGoBack();

Returns

Always returns -1.

Related methods

AVPageView.DoGoForward

DoGoForward

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

Syntax

VARIANT_BOOL DoGoForward();

Returns

Always returns -1.

Related methods

AVPageView.DoGoBack
**GetAperture**

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.

**Syntax**

```c
CAcroRect* GetAperture();
```

**Returns**

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**

Gets the AcroExch.AVDoc associated with the current page.

**Syntax**

```c
LPDISPATCH GetAVDoc();
```

**Returns**

The LPDISPATCH for the AcroExch.AVDoc.

**Related methods**

AVDoc. GetAVPageView
AVDoc. GetPDDoc
AVPageView. GetDoc

**GetDoc**

Gets the AcroExch.PDDoc corresponding to the current page.

**Syntax**

```c
LPDISPATCH GetDoc();
```
Returns

The LPDISPATCH for the AcroExch.PDDoc.

Related methods

AVDoc.GetAVPageView
AVDoc.GetPDDoc
AVPageView.GetAVDoc

GetPage

Gets the AcroExch.PDPage corresponding to the current page.

Syntax

LPDISPATCH GetPage();

Returns

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

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

Syntax

long GetPageNum();
Returns

The current page's page number.

Related methods

AVPageView.GetPage
PDDoc.AcquirePage
PDDoc.GetNumPages
PDPPage.GetDoc
PDPPage.GetNumber
PDPPage.GetRotate
PDPPage.GetSize
PDPTextSelect.GetPage

GetZoom

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

Syntax

long GetZoom();

Returns

The current zoom factor.

Related methods

App.GetPreference
AVPageView.GetZoomType
AVPageView.ZoomTo

GetZoomType

Gets the current zoom type.

Syntax

short GetZoomType();
Returns

Zoom type. The value is one of the following:

- `AVZoomFitHeight` — Fits the page’s height in the window.
- `AVZoomFitPage` — Fits the page in the window.
- `AVZoomFitVisibleWidth` — Fits the page’s visible content into the window.
- `AVZoomFitWidth` — Fits the page’s width into the window.
- `AVZoomNoVary` — A fixed zoom, such as 100%.

Related methods

- `App.GetPreference`
- `AVPageView.GetZoomType`
- `AVPageView.ZoomTo`

Goto

Goes to the specified page.

Syntax

```
VARIANT_BOOL GoTo(long nPage);
```

Parameters

- `nPage` Page number of the destination page. The first page in a PDDoc object is page 0.

Returns

-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

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

**Note:** Deprecated. Do not use this method.

**Syntax**

```cpp
LPDISPATCH PointToDevice(LPDISPATCH iAcroPoint);
```

**Parameters**

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

**Returns**

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

**Related methods**

AVPageView. [DevicePointToPage](#)

ReadPageDown

Scrolls forward through the document by one screen area.

**Syntax**

```cpp
VARIANT_BOOL ReadPageDown();
```

**Returns**

Always returns -1.

**Related methods**

AVPageView. [DoGoBack](#)
AVPageView. [DoGoForward](#)
AVPageView. [Goto](#)
AVPageView. [ReadPageUp](#)
AVPageView. [ScrollTo](#)
AVPageView. [ZoomTo](#)
ReadPageUp

Scrolls backward through the document by one screen area.

Syntax

VARIANT_BOOL ReadPageUp();

Returns

Always returns -1.

Related methods

AVPageView.\texttt{DoGoBack}
AVPageView.\texttt{DoGoForward}
AVPageView.\texttt{Goto}
AVPageView.\texttt{ReadPageDown}
AVPageView.\texttt{ScrollTo}
AVPageView.\texttt{ZoomTo}

ScrollTo

Scrolls to the specified location on the current page.

Syntax

VARIANT_BOOL ScrollTo(short nX, short nY);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nX</td>
<td>The x–coordinate of the destination.</td>
</tr>
<tr>
<td>nY</td>
<td>The y–coordinate of the destination.</td>
</tr>
</tbody>
</table>

Returns

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

Related methods

AVPageView.\texttt{DoGoBack}
AVPageView.\texttt{DoGoForward}
AVPageView.\texttt{Goto}
AVPageView.ReadPageDown
AVPageView.ReadPageUp
AVPageView.ZoomTo

ZoomTo

Zooms to the specified magnification.

Syntax

VARIANT_BOOL ZoomTo(short nType, short nScale);

Parameters

<table>
<thead>
<tr>
<th>nType</th>
<th>Zoom type. Possible values are:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVZoomFitHeight</td>
<td>Fits the page's height into the window.</td>
</tr>
<tr>
<td>AVZoomFitPage</td>
<td>Fits the page into the window.</td>
</tr>
<tr>
<td>AVZoomFitVisibleWidth</td>
<td>Fits the page's visible content into the window.</td>
</tr>
<tr>
<td>AVZoomFitWidth</td>
<td>Fits the page's width into the window.</td>
</tr>
<tr>
<td>AVZoomNoVary</td>
<td>A fixed zoom, such as 100%.</td>
</tr>
</tbody>
</table>

| nScale        | The desired zoom factor, expressed as a percentage. For example, 100 is a magnification of 1.0. |

Returns

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

Related methods

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

A highlighted region of text in a PDF document, which may include one or more contiguous groups of characters or words on a single page. This is a creatable interface. This object has a single method, Add, and is used by the PDPage object to create PDTextSelect objects.

Add

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-based 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.

Syntax

VARIANT_BOOL Add(short nOffset, short nLength);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nOffset</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>nLength</td>
<td>The number of consecutive words or characters to be highlighted.</td>
</tr>
</tbody>
</table>

Returns

Always returns -1.

Related methods

PDPage.CreatePageHilite
PDPage.CreateWordHilite

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

Methods

The PDAnnot object has the following methods.
### Method Description

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetColor</td>
<td>Gets an annotation’s color.</td>
</tr>
<tr>
<td>GetContents</td>
<td>Gets a text annotation’s contents.</td>
</tr>
<tr>
<td>GetDate</td>
<td>Gets an annotation’s date.</td>
</tr>
<tr>
<td>GetRect</td>
<td>Gets an annotation’s bounding rectangle.</td>
</tr>
<tr>
<td>GetSubtype</td>
<td>Gets an annotation’s subtype.</td>
</tr>
<tr>
<td>GetTitle</td>
<td>Gets a text annotation’s title.</td>
</tr>
<tr>
<td>IsEqual</td>
<td>Determines whether an annotation is the same as the specified annotation.</td>
</tr>
<tr>
<td>IsOpen</td>
<td>Tests whether a text annotation is open.</td>
</tr>
<tr>
<td>IsValid</td>
<td>Tests whether an annotation is still valid.</td>
</tr>
<tr>
<td>Perform</td>
<td>Performs a link annotation’s action.</td>
</tr>
<tr>
<td>SetColor</td>
<td>Sets an annotation’s color.</td>
</tr>
<tr>
<td>SetContents</td>
<td>Sets a text annotation’s contents.</td>
</tr>
<tr>
<td>SetDate</td>
<td>Sets an annotation’s date.</td>
</tr>
<tr>
<td>SetOpen</td>
<td>Opens or closes a text annotation.</td>
</tr>
<tr>
<td>SetRect</td>
<td>Sets an annotation’s bounding rectangle.</td>
</tr>
<tr>
<td>SetTitle</td>
<td>Sets a text annotation’s title.</td>
</tr>
</tbody>
</table>

### GetColor

Gets an annotation’s color.

#### Syntax

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

#### Returns

The annotation’s color, a long value of the form 0x00BBGGR 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

Gets a text annotation's contents.

Syntax

BSTR GetContents();

Returns

The annotation's contents.

Related methods

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

GetDate

Gets an annotation's date.

Syntax

LPDISPATCH GetDate();

Returns

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

Related methods

PDAnnot. GetContents
PDAnnot. GetRect
PDAnnot. GetSubtype
PDAnnot. GetTitle
PDAnnot. SetDate

GetRect

Gets an annotation's bounding rectangle.
Syntax

LPDISPATCH GetRect();

Returns

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

Gets an annotation's subtype.

Syntax

BSTR GetSubtype();

Returns

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

Related methods

PDAnnot.GetContents
PDAnnot.GetDate
PDAnnot.GetRect
PDAnnot.GetTitle

GetTitle

Gets a text annotation's title.

Syntax

BSTR getTitle();
Returns

The annotation's title.

Related methods

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

isEqual

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

Syntax

VARIANT_BOOL IsEqual(LPDISPATCH PDAnnot);

Parameters

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

Returns

-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

Tests whether a text annotation is open.
IsValid

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.

Syntax

```cpp
VARIANT_BOOL IsOpen();
```

Returns

-1 if open, 0 otherwise.

Related methods

- PDAannot. GetContents
- PDAannot. GetDate
- PDAannot. GetRect
- PDAannot. GetSubtype
- PDAannot. GetTitle
- PDAannot. IsEqual
- PDAannot. IsValid
- PDAannot. isOpen

IsValid

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.

Syntax

```cpp
VARIANT_BOOL IsValid();
```

Returns

-1 if the annotation is valid, 0 otherwise.

Related methods

- PDAannot. GetContents
- PDAannot. GetDate
- PDAannot. GetRect
- PDAannot. GetSubtype
- PDAannot. GetTitle
- PDAannot. IsEqual
- PDAannot. IsOpen
Perform

Performs a link annotation's action.

Syntax

VARIANT_BOOL Perform(LPDISPATCH iAcroAVDoc);

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.

Returns

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

Related methods

PDAnnot.IsValid

SetColor

Sets an annotation's color.

Syntax

VARIANT_BOOL SetColor(long nRGBColor);

Parameters

nRGBColor

The color to use for the annotation.

Returns

-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.GetColor

PDAnnot.SetContents

PDAnnot.SetDate
SetContents

Sets a text annotation’s contents.

Syntax

VARIANT_BOOL SetContents(BSTR szContents);

Parameters

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

Returns

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

Related methods

PDAnnot.SetOpen
PDAnnot.SetRect
PDAnnotSetTitle

SetDate

Sets an annotation’s date.

Syntax

VARIANT_BOOL SetDate(LPDISPATCH iAcroTime);

Parameters

| iAcroTime | The LPDISPATCH for the date and time to use for the annotation. | iAcroTime’s instance variable m_lpDispatch contains this LPDISPATCH. |
Returns

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

Related methods

PDAnnot. SetTitle
PDAnnot. SetColor
PDAnnot. SetContents
PDAnnot. SetOpen
PDAnnot. SetRect
PDAnnot. setTitle

SetOpen

Opens or closes a text annotation.

Syntax

VARIANT_BOOL SetOpen(long bIsOpen);

Parameters

bIsOpen If a positive number, the annotation is open. If 0, the annotation is closed.

Returns

Always returns -1.

Related methods

PDAnnot. IsOpen
PDAnnot. SetColor
PDAnnot. SetContents
PDAnnot. SetDate
PDAnnot. SetRect
PDAnnot. setTitle

SetRect

Sets an annotation's bounding rectangle.
**Syntax**

VARIANT_BOOL SetRect(LPDISPATCH iAcroRect);

**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>

**Returns**

-1 if a rectangle was supplied, 0 otherwise.

**Related methods**

PDAnnot.SetRect
PDAnnot.SetColor
PDAnnot.SetContents
PDAnnot.SetDate
PDAnnot.SetOpen
PDAnnot.SetTitle

---

**SetTitle**

Sets a text annotation’s title.

**Syntax**

VARIANT_BOOL SetTitle(BSTR szTitle);

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>szTitle</td>
<td>The title to use.</td>
</tr>
</tbody>
</table>

**Returns**

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

**Related methods**

PDAnnot.GetByTitle
PDAnnot.SetColor
PDAnnot.SetContents
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. It is not possible to create a
bookmark with OLE—only to destroy one.

Methods

The PDBookmark object has the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destroy</td>
<td>Destroys a bookmark.</td>
</tr>
<tr>
<td>GetByTitle</td>
<td>Gets the bookmark that has the specified title.</td>
</tr>
<tr>
<td>GetTitle</td>
<td>Gets a bookmark's title.</td>
</tr>
<tr>
<td>IsValid</td>
<td>Determines whether the bookmark is valid.</td>
</tr>
<tr>
<td>Perform</td>
<td>Performs a bookmark's action.</td>
</tr>
<tr>
<td>setTitle</td>
<td>Sets a bookmark's title.</td>
</tr>
</tbody>
</table>

Destroy

Destroys a bookmark.

Syntax

VARIANT_BOOL Destroy();

Returns

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

Related methods

PDBookmark. IsValid
GetByTitle

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.

Syntax

VARIANT_BOOL GetByTitle(LPDISPATCH iAcroPDDoc,  
                        BSTR bookmarkTitle);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iAcroPDDoc</td>
<td>The LPDISPATCH for the document (AcroExch.PDDoc object) containing the bookmark. iAcroPDDoc contains the instance variable m_lpDispatch, which contains the LPDISPATCH.</td>
</tr>
<tr>
<td>bookmarkTitle</td>
<td>The title of the bookmark to get. The capitalization of the title must match that in the bookmark.</td>
</tr>
</tbody>
</table>

Returns

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

Related methods

PDBookmark.GetTitle

PDBookmark.SetTitle

Example

CACroPDBookmark* bookmark = new CACroPDBookmark;

bookmark->CreateDispatch("AcroExch.PDBookmark");

bookmark->GetByTitle(m_pACroAVDoc->GetPDDoc(), "Name of Bookmark");

if (bookmark->IsValid())
    bookmark->Perform(m_pACroAVDoc->m_lpDispatch);
else
    AfxMessageBox("Bookmark not valid");

GetTitle

Gets a bookmark's title.

Syntax

BSTR GetTitle();
Returns

The title.

Related methods

PDBookmark.GetByTitle
PDBookmark.SetTitle

IsValid

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.

Syntax

VARIANT_BOOL IsValid();

Returns

-1 if the bookmark is valid, 0 otherwise.

Related methods

PDBookmark.Destroy

Syntax

Perform

Performs a bookmark's action.

Syntax

VARIANT_BOOL Perform(LPDISPATCH iAcroAVDoc);

Parameters

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

Returns

-1 if the action was executed successfully, 0 otherwise.
Related methods

PDBookmark.IsValid

SetTitle

Sets a bookmark’s title.

Syntax

VARIANT_BOOL SetTitle(BSTR szNewTitle);

Parameters

szNewTitle The title to set.

Returns

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

Related methods

PDBookmark.GetByTitle
PDBookmark.GetTitle

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.

Methods

The PDDoc object has the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcquirePage</td>
<td>Acquires the specified page.</td>
</tr>
<tr>
<td>ClearFlags</td>
<td>Clears a document’s flags.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes a file.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Create</td>
<td>Creates a new AcroExch.PDDoc.</td>
</tr>
<tr>
<td>CreateTextSelect</td>
<td>Creates a text selection from the specified rectangle on the specified page.</td>
</tr>
<tr>
<td>CreateThumbs</td>
<td>Creates thumbnail images for the specified page range in a document.</td>
</tr>
<tr>
<td>CropPages</td>
<td>Crops the pages in a specified range in a document.</td>
</tr>
<tr>
<td>DeletePages</td>
<td>Deletes pages from a file.</td>
</tr>
<tr>
<td>DeleteThumbs</td>
<td>Deletes thumbnail images from the specified pages in a document.</td>
</tr>
<tr>
<td>GetFileName</td>
<td>Gets the name of the file associated with this AcroExch.PDDoc.</td>
</tr>
<tr>
<td>GetFlags</td>
<td>Gets a document’s flags.</td>
</tr>
<tr>
<td>GetInfo</td>
<td>Gets the value of a specified key in the document’s Info dictionary.</td>
</tr>
<tr>
<td>GetInstanceID</td>
<td>Gets the instance ID (the second element) from the ID array in the document’s trailer.</td>
</tr>
<tr>
<td>GetJSObject</td>
<td>Gets a dual interface to the JavaScript object associated with the PDDoc.</td>
</tr>
<tr>
<td>GetNumPages</td>
<td>Gets the number of pages in a file.</td>
</tr>
<tr>
<td>GetPageMode</td>
<td>Gets a value indicating whether the Acrobat application is currently displaying only pages, pages and thumbnails, or pages and bookmarks.</td>
</tr>
<tr>
<td>GetPermanentID</td>
<td>Gets the permanent ID (the first element) from the ID array in the document’s trailer.</td>
</tr>
<tr>
<td>InsertPages</td>
<td>Inserts the specified pages from the source document after the indicated page within the current document.</td>
</tr>
<tr>
<td>MovePage</td>
<td>Moves a page to another location within the same document.</td>
</tr>
<tr>
<td>Open</td>
<td>Opens a file.</td>
</tr>
<tr>
<td>OpenAVDoc</td>
<td>Opens a window and displays the document in it.</td>
</tr>
<tr>
<td>ReplacePages</td>
<td>Replaces the indicated pages in the current document with those specified from the source document.</td>
</tr>
<tr>
<td>Save</td>
<td>Saves a document.</td>
</tr>
<tr>
<td>SetFlags</td>
<td>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.</td>
</tr>
<tr>
<td>SetInfo</td>
<td>Sets the value of a key in a document’s Info dictionary.</td>
</tr>
<tr>
<td>SetPageMode</td>
<td>Sets the page mode in which a document is to be opened: display only pages, pages and thumbnails, or pages and bookmarks.</td>
</tr>
</tbody>
</table>
AcquirePage

Acquires the specified page.

Syntax

LPDISPATCH AcquirePage(long nPage);

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>

Returns

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
PDPdf. GetDoc
PDPdf. GetNumber
PDPdf. GetRotate
PDPdf. GetSize
PDPdfTextSelect. GetPage

ClearFlags

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 be used only to clear, not to set, the flag bits.

Syntax

VARIANT_BOOL ClearFlags(long nFlags);

Parameters

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

Always returns -1.

Related methods

PDDoc.

Close

Closes a file.

**Note:** If PDDoc and AVDoc are constructed with the same file, PDDoc.Close destroys both objects (which closes the document in the viewer).

Syntax

VARIANT_BOOL Close();

Returns

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

Related methods

App.

Create

Creates a new AcroExch.PDDoc.

Syntax

VARIANT_BOOL Create();
Returns

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

CreateTextSelect

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.

Syntax

LPDISPATCH CreateTextSelect(long nPage, LPDISPATCH iAcroRect);

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>

Returns

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
PDPage.CreatePageHilite
PDPage.CreateWordHilite
PDTextSelect.Destroy
PDTextSelect.GetBoundingRect
PDTextSelect.GetNumText
PDTextSelect.GetPage
PDTextSelect.GetText
CreateThumbs

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

Syntax

VARIANT_BOOL CreateThumbs(long nFirstPage, long nLastPage);

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>

Returns

-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

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).

Syntax

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

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 PDDoc object is page 0.</td>
</tr>
<tr>
<td>nEndPage</td>
<td>Last page that is cropped.</td>
</tr>
<tr>
<td>nEvenOrOddPagesOnly</td>
<td>Value indicating which pages in the range are cropped. Must be one of the following:</td>
</tr>
<tr>
<td></td>
<td>0 — crop all pages in the range</td>
</tr>
<tr>
<td></td>
<td>1 — crop only odd pages in the range</td>
</tr>
<tr>
<td></td>
<td>2 — crop only even pages in the range</td>
</tr>
<tr>
<td>iAcroRect</td>
<td>An LPDISPATCH for a CAcroRect specifying the cropping rectangle, which is specified in user space.</td>
</tr>
</tbody>
</table>
Returns

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

Related methods

PDPage. CropPages

DeletePages

Deletes pages from a file.

Syntax

VARIANT_BOOL DeletePages(long nStartPage, long nEndPage);

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>

Returns

-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

Deletes thumbnail images from the specified pages in a document.

Syntax

VARIANT_BOOL DeleteThumbs(long nStartPage, long nEndPage);
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nStartPage</td>
<td>First page whose thumbnail image is deleted. The first page in a PDDoc object is page 0.</td>
</tr>
<tr>
<td>nEndPage</td>
<td>Last page whose thumbnail image is deleted.</td>
</tr>
</tbody>
</table>

Returns

-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

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

Syntax

BSTR GetFileName();

Returns

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

Related methods

PDDoc.Save

GetFlags

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.

Syntax

long GetFlags();

Returns

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>
</tbody>
</table>
GetInfo

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

Syntax

BSTR GetInfo(BSTR szInfoKey);

Parameters

szInfoKey The key whose value is obtained.

Returns

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

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

BSTR GetInstanceID();

Returns

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

Related methods

PDDoc.GetPermanentID

GetJSObject

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 more information on working with JavaScript, see Developing Applications Using Interapplication Communication.

Syntax

LDispatch* GetJSObject();

Returns

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

GetNumPages

Gets the number of pages in a file.

Syntax

long GetNumPages();

Returns

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

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

Syntax

long GetPageMode();

Returns

The current page mode. Will be one of the following 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

Related methods

PDDoc.SetPageMode

GetPermanentID

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

Syntax

BSTR GetPermanentID();

Returns

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

Related methods

PDDoc.GetInstanceID

InsertPages

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

Syntax

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

Returns

-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

MovePage

Moves a page to another location within the same document.

Syntax

VARIANT_BOOL MovePage(long nMoveAfterThisPage, long nPageToMove);

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>

Returns

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

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

Syntax

VARIANT_BOOL Open(BSTR szFullPath);

Parameters

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

Returns

-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

Opens a window and displays the document in it.

Syntax

LPDISPATCH OpenAVDoc(BSTR szTitle);
Parameters

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

Returns

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
- AVDocSetTitle
- PDDoc.Close
- PDDoc.Open

ReplacePages

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.

Syntax

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

Returns

-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

Saves a document.

Syntax

VARIANT_BOOL Save(short nType, BSTR szFullPath);
Parameters

nType

Specifies the way in which the file should be saved.

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 optimized for the web, providing hint tables. This allows the PDF file to be byte-served. This flag can only be specified if PDSaveFull is also used.

**Note:** If you save a file optimized for the web using the PDSaveLinearized flag, you must follow this sequence:

1. Open the PDF file with PDDoc. **Open**.
2. Call PDDoc. **Save** using the PDSaveLinearized flag.
3. Call PDDoc. **Close**.

This allows batch optimization of files.

szFullPath

The new path to the file, if any.

Returns

-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

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 be used only to set, not to clear, the flag bits.

Syntax

```cpp
VARIANT_BOOL SetFlags(long nFlags);
```
Parameters

| nFlags                  | 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. |

Returns

Always returns -1.

Related methods

PDDoc.ClearFlags
PDDoc.GetFlags

SetInfo

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

Syntax

VARIANT_BOOL SetInfo(BSTR szInfoKey, BSTR szBuffer);

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>

Returns

-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

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

Syntax

VARIANT_BOOL SetPageMode(long nPageMode);
Parameters

<table>
<thead>
<tr>
<th>nPageMode</th>
<th>The page mode to be set. Possible values:</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDDontCare: 0</td>
<td>leave the view mode as it is</td>
</tr>
<tr>
<td>PDUseNone: 1</td>
<td>display without bookmarks or thumbnails</td>
</tr>
<tr>
<td>PDUseThumbs: 2</td>
<td>display using thumbnails</td>
</tr>
<tr>
<td>PDUseBookmarks: 3</td>
<td>display using bookmarks</td>
</tr>
</tbody>
</table>

Returns

Always returns -1.

Related methods

PDDoc.GetPageMode
PDDoc.SetPageMode

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 threads used in any articles that occur on the page. The first page in a PDDoc object is page 0.

Methods

The PDPage object has the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddAnnot</td>
<td>Adds a specified annotation at a specified location in the page's annotation array</td>
</tr>
<tr>
<td>AddNewAnnot</td>
<td>Creates a new text annotation and adds it to the page.</td>
</tr>
<tr>
<td>CopyToClipboard</td>
<td>Copies a PDF image to the clipboard without requiring an hWnd or hDC from the client.</td>
</tr>
<tr>
<td>CreatePageHilite</td>
<td>Creates a text selection from a list of character offsets and character counts on a single page.</td>
</tr>
<tr>
<td>CreateWordHilite</td>
<td>Creates a text selection from a list of word offsets and word counts on a single page.</td>
</tr>
<tr>
<td>CropPage</td>
<td>Crops the page.</td>
</tr>
<tr>
<td>Draw</td>
<td>Deprecated. Draws page contents into a specified window.</td>
</tr>
</tbody>
</table>
**AddAnnot**

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

**Syntax**

```c
VARIANT_BOOL AddAnnot(long nIndexAddAfter,
LPDISPATCH iPDAannot);
```

**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 to add the annotation. The first annotation on a page has an index of zero.</td>
</tr>
<tr>
<td>iPDAannot</td>
<td>The LPDISPATCH for the AcroExch.PDAnnot to add. iPDAannot contains the instance variable m_lpDispatch, which contains the LPDISPATCH.</td>
</tr>
</tbody>
</table>

**Returns**

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

**Related methods**

- [AddNewAnnot](#)
- [RemoveAnnot](#)
AddNewAnnot

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 to fill in the /Contents key.

Syntax

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

Parameters

| nIndexAddAfter | Location in the page's annotation array after which to add the annotation. The first annotation on a page has an index of zero. |
| szSubType     | Subtype of the annotation to be created. Must be text. |
| iAcroRect     | 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. |

Returns

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

Copies a PDF image to the clipboard without requiring an hWnd or hDC from the client. This method is only available on 32-bit systems.

Syntax

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

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.

Syntax

LPDISPATCH CreatePageHilite(LPDISPATCH iAcroHiliteList);

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_lDispatch, which contains the LPDISPATCH. Use HiliteList.Add to create a highlight list.</td>
</tr>
</tbody>
</table>

Returns

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
CreateWordHilite

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.

**Syntax**

LPDISPATCH CreateWordHilite(LPDISPATCH iAcroHiliteList);

**Parameters**

| iAcroHiliteList | 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. |

**Returns**

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
PDPPage.CreatePageHilite
PDTextSelect.Destroy
PDTextSelect.GetBoundingRect
PDTextSelect.GetNumText
PDTextSelect.GetPage
PDTextSelect.GetText
CropPage

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).

Syntax

VARIANT_BOOL CropPage(LPDISPATCH iAcroRect);

Parameters

- **iAcroRect**: An LPDISPATCH for a CAcroRect specifying the cropping rectangle, which is specified in user space.

Returns

-1 if the page was cropped successfully, 0 otherwise.

Related methods

PDDoc.CropPages

Draw

*Note*: Deprecated. As of Acrobat 3.0, this method simply returns false. Use the method AVDoc.DrawEx instead.

Syntax

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

Parameters

- **window**: HWND into which the page is to be drawn.
- **displayContext**: hDC to use for drawing. If NULL, the HDC for window is used.
- **XOrigin**: The x-coordinate of the portion of the page to be drawn.
- **YOrigin**: The y-coordinate of the portion of the page to be drawn.
- **zoom**: Zoom factor at which the page is to be drawn, specified as a percent. For example, 100 corresponds to a magnification of 1.0.

Returns

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

PDPAGE. **CopyToClipboard**

PDPAGE. **DrawEx**

**DrawEx**

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.

**Syntax**

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

**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; do not use it. Assign it a NULL value. If it is not assigned NULL, an exception is thrown. <strong>Note</strong>: displayContext cannot be reliably used as the hDC for a printer device. In particular, Visual Basic applications cannot use <strong>DrawEx</strong> 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. Use methods in the CAcroRect class to set the size of the rectangle. For example:</td>
</tr>
<tr>
<td>xOrigin</td>
<td>The x-coordinate of the portion of the page to be drawn.</td>
</tr>
</tbody>
</table>
GetAnnot

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

Syntax

    LPDISPATCH GetAnnot(long nIndex);

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. |

Returns

The LPDISPATCH for the AcroExch.PDAnnot object.

Related methods

PDPage.GetAnnotIndex
PDPage.GetNumAnnots

GetAnnotIndex

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

Syntax

    long GetAnnotIndex(LPDISPATCH iPDAannot);

Parameters

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

The annotation's index.

Related methods

PDPage. GetAnnot

PDPage. GetNumAnnots

GetDoc

Gets the AcroExch. PDDoc associated with the page.

Syntax

LPDISPATCH GetDoc();

Returns

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

Gets the number of annotations on the page.

Annotations that have associated pop-up windows, such as a strikeout, count as two annotations. Also note that widget annotations (Acrobat form fields) are included.

Syntax

long GetNumAnnots();
Returns

The number of annotations on the page.

Related methods

PDPages. GetAnnot
PDPages. GetAnnotIndex

GetNumber

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

Syntax

long GetNumber();

Returns

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

Related methods

AVPageView. GetPage
AVPageView. GetPageNum
PDDocs. AcquirePage
PDDocs. GetNumPages
PDPage. GetDoc
PDPage. GetRotate
PDPage. GetSize
PDTextSelect. GetPage

GetRotate

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

Syntax

short GetRotate();

Returns

Rotation value.
Related methods

AVPageView.\texttt{GetPage}

AVPageView.\texttt{GetPageNum}

PDDoc.\texttt{AcquirePage}

PDPage.\texttt{GetNumber}

PDPage.\texttt{GetSize}

PDPage.\texttt{SetRotate}

PDTextSelect.\texttt{GetPage}

\subsection*{GetSize}

Gets a page's width and height in points.

\textbf{Syntax}

\begin{verbatim}
LPDISPATCH GetSize();
\end{verbatim}

\textbf{Returns}

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

Related methods

AVPageView.\texttt{GetPage}

AVPageView.\texttt{GetPageNum}

PDDoc.\texttt{AcquirePage}

PDPage.\texttt{GetNumber}

PDPage.\texttt{SetRotate}

PDTextSelect.\texttt{GetPage}

\subsection*{RemoveAnnot}

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

\textbf{Syntax}

\begin{verbatim}
VARIANT_BOOL RemoveAnnot(long nIndex);
\end{verbatim}
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. |

Returns

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

Related methods

PDPage. AddAnnot
PDPage. AddNewAnnot
PDPage. GetAnnotIndex

SetRotate

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

Syntax

VARIANT_BOOL SetRotate(short nRotate);

Parameters

| nRotate  | Rotation value of 0, 90, 180, or 270. |

Returns

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

Related methods

PDPage. GetRotate
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 number of the first word of a series of selected words and “end” is the number of the next word after the last word in the selection.

Methods

The PDTextSelect object has the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destroy</td>
<td>Destroys a text selection object.</td>
</tr>
<tr>
<td>GetBoundingRect</td>
<td>Gets a text selection’s bounding rectangle.</td>
</tr>
<tr>
<td>GetNumText</td>
<td>Gets the number of text elements in a text selection.</td>
</tr>
<tr>
<td>GetPage</td>
<td>Gets the page number on which the text selection is located.</td>
</tr>
<tr>
<td>GetText</td>
<td>Gets the text from the specified element of a text selection.</td>
</tr>
</tbody>
</table>

Destroy

Destroys a text selection object.

Syntax

VARIANT_BOOL Destroy();

Returns

Always returns -1.

Related methods

AVDoc.ClearSelection
AVDoc.SetTextSelection
AVDoc.ShowTextSelect
PDDoc.CreateTextSelect
PDPage.CreatePageHilite
PDPage.CreateWordHilite
PDTextSelect.GetBoundingRect
PDTextSelect.GetNumText
GetBoundingRect

Gets a text selection's bounding rectangle.

Syntax

LPDISPATCH GetBoundingRect();

Returns

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

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.

Syntax

long GetNumText();

Returns

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

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

Syntax

long GetPage();

Returns

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
GetText

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.

Syntax

BSTR GetText(long nTextIndex);

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>

Returns

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
AcroExch.Point

Defines the location of an AcroPoint.

Properties

The Point object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Gets or sets the x-coordinate of an AcroPoint.</td>
</tr>
<tr>
<td>Y</td>
<td>Gets or sets the y-coordinate of an AcroPoint.</td>
</tr>
</tbody>
</table>

X

Gets or sets the x-coordinate of an AcroPoint.

Syntax

[get/set] Short

Return

The x-coordinate of the AcroPoint.

Y

Gets or sets the y-coordinate of an AcroPoint.

Syntax

[get/set] Short

Returns

The y-coordinate of the AcroPoint.

AcroExch.Rect

Defines the location of an AcroRect.

The Rect object has the following properties.
Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bottom</td>
<td>Gets or sets the bottom y-coordinate of an AcroRect.</td>
</tr>
<tr>
<td>Left</td>
<td>Gets or sets the left x-coordinate of an AcroRect.</td>
</tr>
<tr>
<td>Right</td>
<td>Gets or sets the right x-coordinate of an AcroRect.</td>
</tr>
<tr>
<td>Top</td>
<td>Gets or sets the top y-coordinate of an AcroRect.</td>
</tr>
</tbody>
</table>

Bottom

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

Syntax

[get/set] Short

Returns

The y-coordinate of the bottom of the AcroRect.

Left

Gets or sets left x-coordinate of an AcroRect.

Syntax

[get/set] Short

Returns

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

Right

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

Syntax

[get/set] Short

Returns

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

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

Syntax

\[\text{get/set}\] **Short**

Returns

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

**AcroExch.Time**

Defines a specified time, accurate to the millisecond.

Properties

The **Time** object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Date</strong></td>
<td>Gets or sets the date from an <strong>AcroTime</strong>.</td>
</tr>
<tr>
<td><strong>Hour</strong></td>
<td>Gets or sets the hour from an <strong>AcroTime</strong>.</td>
</tr>
<tr>
<td><strong>Millisecond</strong></td>
<td>Gets or sets the milliseconds from an <strong>AcroTime</strong>.</td>
</tr>
<tr>
<td><strong>Minute</strong></td>
<td>Gets or sets the minutes from an <strong>AcroTime</strong>.</td>
</tr>
<tr>
<td><strong>Month</strong></td>
<td>Gets or sets the month from an <strong>AcroTime</strong>.</td>
</tr>
<tr>
<td><strong>Second</strong></td>
<td>Gets or sets the seconds from an <strong>AcroTime</strong>.</td>
</tr>
<tr>
<td><strong>Year</strong></td>
<td>Gets or sets the year from an <strong>AcroTime</strong>.</td>
</tr>
</tbody>
</table>

**Date**

Gets or sets the date from an **AcroTime**.

Syntax

\[\text{get/set}\] **Short**

Returns

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

Gets or sets the hour from an `AcroTime`.

**Syntax**

```
[get/set] Short
```

**Returns**

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

Millisecond

Gets or sets the milliseconds from an `AcroTime`.

**Syntax**

```
[get/set] Short
```

**Returns**

The milliseconds from the `AcroTime`. Milliseconds run from 0 to 999.

Minute

Gets or sets the minutes from an `AcroTime`.

**Syntax**

```
[get/set] Short
```

**Returns**

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

Month

Gets or sets the month from an `AcroTime`.

**Syntax**

```
[get/set] Short
```

**Returns**

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

Gets or sets the seconds from an AcroTime.

Syntax

[get/set] Short

Returns

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

Year

Gets or sets the year from an AcroTime.

Syntax

[get/set] Short

Returns

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

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.

Methods

The AxAcroPDF object has the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetVersions</td>
<td>Deprecated</td>
</tr>
<tr>
<td>GoBackwardStack</td>
<td>Goes to the previous view on the view stack, if the previous view exists.</td>
</tr>
<tr>
<td>GoForwardStack</td>
<td>Goes to the next view on the view stack, if the next view exists.</td>
</tr>
<tr>
<td>GotoFirstPage</td>
<td>Goes to the first page in the document, maintaining the current location within the page and zoom level.</td>
</tr>
<tr>
<td>GotoLastPage</td>
<td>Goes to the last page in the document, maintaining the current location within the page and zoom level.</td>
</tr>
<tr>
<td>GotoNextPage</td>
<td>Goes to the next page in the document, if it exists. Maintains the current location within the page and zoom level.</td>
</tr>
</tbody>
</table>
### Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GotoPreviousPage</td>
<td>Goes to the previous page in the document, if it exists. Maintains the current location within the page and zoom level.</td>
</tr>
<tr>
<td>LoadFile</td>
<td>Opens and displays the specified document within the browser.</td>
</tr>
<tr>
<td>Print</td>
<td>Prints the document according to the options selected in a user dialog box.</td>
</tr>
<tr>
<td>PrintAll</td>
<td>Prints the entire document without displaying a user dialog box.</td>
</tr>
<tr>
<td>PrintAllFit</td>
<td>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.</td>
</tr>
<tr>
<td>PrintPages</td>
<td>Prints the specified pages without displaying a user dialog box.</td>
</tr>
<tr>
<td>PrintPagesFit</td>
<td>Prints the specified pages without displaying a user dialog box.</td>
</tr>
<tr>
<td>PrintWithDialog</td>
<td>Prints the document according to the options selected in a user dialog box.</td>
</tr>
<tr>
<td>SetCurrentHighlight</td>
<td>Highlights the text selection within the specified bounding rectangle on the current page.</td>
</tr>
<tr>
<td>SetCurrentPage</td>
<td>Goes to the specified page in the document.</td>
</tr>
<tr>
<td>SetLayoutManager</td>
<td>Sets the layout mode for a page view according to the specified string.</td>
</tr>
<tr>
<td>SetNamedDest</td>
<td>Changes the page view to the named destination in the specified string.</td>
</tr>
<tr>
<td>SetPageMode</td>
<td>Sets the page mode according to the specified string.</td>
</tr>
<tr>
<td>SetShowScrollbars</td>
<td>Determines whether scrollbars will appear in the document view.</td>
</tr>
<tr>
<td>SetShowToolbar</td>
<td>Determines whether a toolbar will appear in the viewer.</td>
</tr>
<tr>
<td>SetView</td>
<td>Sets the view of a page according to the specified string.</td>
</tr>
<tr>
<td>SetViewRect</td>
<td>Sets the view rectangle according to the specified coordinates.</td>
</tr>
<tr>
<td>SetViewScroll</td>
<td>Sets the view of a page according to the specified string.</td>
</tr>
<tr>
<td>SetZoom</td>
<td>Sets the magnification according to the specified value.</td>
</tr>
<tr>
<td>SetZoomScroll</td>
<td>Sets the magnification according to the specified value, and scrolls the page view both horizontally and vertically according to the specified amounts.</td>
</tr>
</tbody>
</table>

### Properties

The AxAcroPDF object has the following property.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src</td>
<td>Gets or sets the URL for the document.</td>
</tr>
</tbody>
</table>
GetVersions

   **Note:** Deprecated. This method is no longer available.

**Syntax**

   `VARIANT GetVersions();`

GoBackwardStack

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

**Syntax**

   `void GoBackwardStack();`

**Related methods**

   AcroPDF.Goforwardstack

GoForwardStack

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

**Syntax**

   `void GoForwardStack();`

**Related methods**

   AcroPDF.GoBackwardStack

GotoFirstPage

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

**Syntax**

   `void gotoFirstPage();`

**Related methods**

   AcroPDF.GotoLastPage
   AcroPDF.GotoNextPage
   AcroPDF.GotoPreviousPage
   AcroPDFsetCurrentPage
GotoLastPage

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

Syntax

```csharp
void gotoLastPage();
```

Related methods

- AcroPDF.GotoFirstPage
- AcroPDF.GotoNextPage
- AcroPDF.GotoPreviousPage
- AcroPDFsetCurrentPage

GotoNextPage

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

Syntax

```csharp
void gotoNextPage();
```

Related methods

- AcroPDF.GotoFirstPage
- AcroPDF.GotoLastPage
- AcroPDF.GotoPreviousPage
- AcroPDFsetCurrentPage

GotoPreviousPage

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

Syntax

```csharp
void gotoPreviousPage();
```

Related methods

- AcroPDF.GotoFirstPage
- AcroPDF.GotoLastPage
- AcroPDF.GotoNextPage
- AcroPDFsetCurrentPage
**LoadFile**

Opens and displays the specified document within the browser.

**Syntax**

```c
VARIANT_BOOL LoadFile(BSTR fileName);
```

**Parameters**

| fileName | The path of the file to be opened. |

**Returns**

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

**Print**

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 is ignored.

**Syntax**

```c
void Print();
```

**Related methods**

- AcroPDF.*PrintAll*
- AcroPDF.*PrintAllFit*
- AcroPDF.*PrintPages*
- AcroPDF.*PrintPagesFit*
- AcroPDF.*PrintWithDialog*

**PrintAll**

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 is ignored.

**Syntax**

```c
void printAll();
```
Related methods

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

PrintAllFit

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 is ignored.

**Syntax**

void printAllFit(VARIANT_BOOL bOn);

**Parameters**

| bOn | 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. |

---

PrintPages

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 is ignored.

**Syntax**

void printPages(Long nFrom, Long nTo);
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>

Related methods

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

PrintPagesFit

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

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

Syntax

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

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>

Related methods

AcroPDF.Print
AcroPDF.PrintAll
AcroPDF.PrintAllFit
AcroPDF.PrintPages
AcroPDF.PrintWithDialog
PrintWithDialog

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 is ignored.

**Syntax**

```c
void printWithDialog();
```

**Related methods**

- AcroPDF.Print
- AcroPDF.PrintAll
- AcroPDF.PrintAllFit
- AcroPDF.PrintPages
- AcroPDF.PrintPagesFit

SetCurrentHighlight

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

**Syntax**

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

**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>

SetCurrentPage

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

**Syntax**

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

| nPage  | The page number of the destination page. The first page in a document is page 0. |

Related methods

AcroPDF. `GotoFirstPage`
AcroPDF. `GotoLastPage`
AcroPDF. `GotoNextPage`
AcroPDF. `GotoPreviousPage`

SetLayoutMode

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

Syntax

```c
void setLayoutMode(BSTR szLayoutMode);
```

Parameters

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

Related methods

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

SetNamedDest

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

Syntax

```c
void setNamedDest(BSTR szNamedDest);
```
**SetPageMode**

Sets the page mode according to the specified string.

**Syntax**

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

**Parameters**

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

**Related methods**

- AcroPDF. `SetLayoutMode`
- AcroPDF. `SetView`
- AcroPDF. `SetViewRect`
- AcroPDF. `SetViewScroll`

**SetShowScrollbars**

Determines whether scrollbars will appear in the document view.

**Syntax**

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

**Parameters**

| bOn | A positive value indicates that scrollbars will appear, 0 indicates that they will not. |
Related methods

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

### SetShowToolbar

Determines whether a toolbar will appear in the viewer.

**Syntax**

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

**Parameters**

- `bOn` A positive value indicates that the toolbar will appear, 0 indicates that it will not.

### Related methods

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

### SetView

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

**Syntax**

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

**Parameters**

- `szViewMode` Possible values:
  - `Fit` — Fits the entire page within the window both vertically and horizontally.
  - `FitH` — Fits the entire width of the page within the window.
  - `FitV` — Fits the entire height of the page within the window.
  - `FitB` — Fits the bounding box within the window both vertically and horizontally.
  - `FitBH` — Fits the entire width of the bounding box within the window.
  - `FitB` — Fits the entire height of the bounding box within the window.
Related methods

AcroPDF. SetLayoutMode
AcroPDF. SetNamedDest
AcroPDF. SetViewRect
AcroPDF. SetViewScroll

SetViewRect

Sets the view rectangle according to the specified coordinates.

Syntax

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

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>

Related methods

AcroPDF. SetLayoutMode
AcroPDF. SetNamedDest
AcroPDF. SetView
AcroPDF. SetViewScroll

SetViewScroll

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.

Syntax

void setViewRect(BSTR szViewMode, FLOAT offset);
Parameters

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

Related methods

`AcroPDF.SetLayoutMode`

`AcroPDF.SetNamedDest`

`AcroPDF.SetView`

`AcroPDF.SetViewRect`

SetZoom

Sets the magnification according to the specified value.

Syntax

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

Parameters

| percent | The desired zoom factor, expressed as a percentage. For example, 1.0 represents a magnification of 100%. |

Related methods

`AcroPDF.SetZoomScroll`

SetZoomScroll

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

Syntax

```c
void setZoomScroll(FLOAT percent, FLOAT left, FLOAT top);
```
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>

Related methods

AcroPDF. **SetZoom**

**Src**

Gets or sets the URL for the document.

**Syntax**

```
[get/set] src
```

**Returns**

The URL for the document, formatted as a string.
DDE Messages

This chapter lists all DDE messages supported by Acrobat.

These DDE messages handle the display of the Acrobat application:
- **AppExit**
- **AppHide**
- **AppShow**
- **CloseAllDocs**
- **HideToolbar**
- **MenuItemExecute**
- **ShowToolbar**

These DDE messages control the display of the document:
- **DocClose**
- **DocDeletePages**
- **DocInsertPages**
- **DocOpen**
- **DocReplacePages**
- **DocSave**
- **DocSaveAs**
- **DocSetViewMode**
- **FileOpen**
- **FileOpenEx**

These DDE messages handle printing of a document:
- **DocPrint**
- **FilePrint**
- **FilePrintEx**
- **FilePrintSilent**
- **FilePrintSilentEx**
- **FilePrintTo**
- **FilePrintToEx**

These DDE messages control the view of a document:
- **DocGoTo**
- **DocGoToNameDest**
- **DocPageDown**
- **DocPageLeft**
Adobe Reader supports the following subset of DDE messages:

- **AppExit**
- **CloseAllDocs**
- **DocClose**
- **DocGoTo**
- **DocGoToNameDest**
- **DocOpen**
- **FileOpen**
- **FileOpenEx**
- **FilePrint**
- **FilePrintEx**
- **FilePrintSilent**
- **FilePrintSilentEx**
- **FilePrintTo**
- **FilePrintToEx**

### AppExit

Exits the Acrobat application.

*AppExit* is also supported in Adobe Reader.

#### Syntax

```plaintext
[AppExit()]
```

#### Returns

*true* if the Acrobat application exits successfully, *false* otherwise.

#### Related methods

- **AppHide**
- **AppShow**
AppHide

Iconifies or hides the Acrobat application.

Syntax

[AppHide()]

Returns

true if the Acrobat application is hidden successfully, false otherwise.

Related methods

AppExit
AppShow

AppShow

Shows the Acrobat application.

Syntax

[AppShow()]

Returns

true if the Acrobat application is shown successfully, false otherwise.

Related methods

AppExit
AppHide

CloseAllDocs

Closes all open documents.

CloseAllDocs is also supported in Adobe Reader.

Syntax

[CloseAllDocs()]

Returns

true if the documents are closed successfully, false otherwise.
Related methods

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

**DocClose**

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

*DocClose* is also supported in Adobe Reader.

**Syntax**

```
[DocClose(char* fullPath)]
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full path of the file to be closed.</td>
</tr>
</tbody>
</table>

**Returns**

- *true* if the document is closed successfully, *false* if the document does not exist or is not closed successfully.

**Related methods**

- **CloseAllDocs**
- **DocOpen**
- **FileOpen**

**DocDeletePages**

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

**Syntax**

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full path 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>

Returns

true if the pages are 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 are not deleted successfully.

Related methods

- DocInsertPages
- DocReplacePages

DocFind

Finds a string in a specified file. This does not use a cross-document search, but instead performs a page-by-page search of the specified file.

Syntax

```
[DocFind(char* fullPath, char* string, boolean caseSensitive, boolean wholeWords, boolean bReset)]
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full path 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 if the search begins on the current page.</td>
</tr>
</tbody>
</table>

Returns

false if the document specified by fullPath does not exist or if the text is not found, true otherwise.
DocGoTo

Goes to the specified page.

DocGoTo is also supported in Adobe Reader.

Syntax

\[DocGoTo(char* fullPath, long pageNum)\]

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full path of the file.</td>
</tr>
<tr>
<td>pageNum</td>
<td>The page number of the destination page.</td>
</tr>
</tbody>
</table>

Returns

false if the document specified by fullPath does not exist, true otherwise.

DocGoToNameDest

Goes to the specified named destination.

DocGoToNameDest is also supported in Adobe Reader.

Syntax

\[DocGoToNameDest(char* fullPath, char* nameDest)\]

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full path of the file.</td>
</tr>
<tr>
<td>nameDest</td>
<td>The named destination.</td>
</tr>
</tbody>
</table>

Returns

false if the document specified by fullPath does not exist, true otherwise.

DocInsertPages

Inserts pages from one file into another.

Syntax

\[DocInsertPages(char* fullPath, long insertAfterPage, char* sourcePath)\]
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full path 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 can be a page number or one of the following:  
  PDBeforeFirstPage — Pages are inserted at the beginning of the document.  
  PDLastPage — Pages are inserted at the end of the document. |
| sourcePath      | The full path of the source document. This file need not be open in the Acrobat application. |

Returns

true if the pages are inserted successfully, false if the document does not exist or the pages are not inserted successfully.

Related methods

DocDeletePages  
DocReplacePages

DocOpen

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.

Syntax

[DocOpen(char* fullPath)]

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full path of the file to be opened.</td>
</tr>
</tbody>
</table>

Returns

true if the file is opened successfully, false otherwise.

Related methods

CloseAllDocs  
DocClose  
FileOpen
DocPageDown

Scrolls forward through the document by one screen area.

Syntax

[DocPageDown(char* fullPath)]

Parameters

fullPath The full path of the document.

Returns

false if the document specified by fullPath does not exist, true otherwise.

Related methods

DocPageLeft
DocPageRight
DocPageUp
DocScrollTo

DocPageLeft

Scrolls to the left by a small amount.

Syntax

[DocPageLeft(char* fullPath)]

Parameters

fullPath The full path of the document.

Returns

false if the document specified by fullPath does not exist, true otherwise.

Related methods

DocPageDown
DocPageRight
DocPageUp
DocPageUp
DocPageRight

Scrolls to the right by a small amount.

Syntax

[DocPageRight(char* fullPath)]

Parameters

fullPath The full path of the document.

Returns

false if the document specified by fullPath does not exist, true otherwise.

Related methods

DocPageDown
DocPageLeft
DocPageUp
DocPageUp

DocPageUp

Scrolls backward through the document by one screen area.

Syntax

[DocPageUp(char* fullPath)]

Parameters

fullPath The full path of the document.

Returns

false if the document specified by fullPath does not exist, true otherwise.

Related methods

DocPageDown
DocPageLeft
DocPageRight
DocPageRight
DocScrollTo
DocPrint

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.

Syntax

[DocPrint(char* fullPath, long startPage, long endPage)]

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full path 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>

Returns

false if the document specified by fullPath does not exist, true otherwise.

Related methods

FilePrint
FilePrintSilent
FilePrintTo

DocReplacePages

Replaces pages in the target document using the specified pages from the source document.

Syntax

[DocReplacePages(char* fullPath, long startDestPage, char* sourcePath, long startSourcePage, long endSourcePage)]

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full path of the target document. This file must already be open in the Acrobat application.</td>
</tr>
<tr>
<td>startDestPage</td>
<td>The page number of the first page in the target document to be replaced.</td>
</tr>
<tr>
<td>sourcePath</td>
<td>The full path of the source document. This file does not have to be already open in the Acrobat application.</td>
</tr>
</tbody>
</table>
Returns

true if the pages are replaced successfully. Returns false if the document does not exist or the pages are not replaced successfully.

Related methods

DocDeletePages
DocInsertPages

DocSave

Saves the specified file. The user is not warned if there are any problems saving the file.

Syntax

[DocSave(char* fullPath)]

Parameters

fullPath The full path of the file to be saved.

Returns

true if the document is saved successfully, false if the document does not exist or is not saved successfully.

Related methods

DocSaveAs

DocSaveAs

Saves an open file to a new path. The user is not warned if there are any problems saving the file.

Syntax

[DocSaveAs(char* fullPath, char* newPath)]
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full path of the existing file.</td>
</tr>
<tr>
<td>newPath</td>
<td>The full path of the new file.</td>
</tr>
</tbody>
</table>

Returns

true if the document is saved successfully, false if the document does not exist or is not saved successfully.

Related methods

DocSave

DocScrollTo

Scrolls the view of the current page to the specified location.

Syntax

[DocScrollTo(char* fullPath, int x, int y)]

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full path 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>

Returns

false if the document specified by fullPath does not exist, true otherwise.

Related methods

DocPageDown
DocPageLeft
DocPageRight
DocPageUp
DocSetViewMode

Determines whether bookmarks, thumbnail images, or neither are shown in addition to the document.

Syntax

[DocSetViewMode(char* fullPath, char* viewType)]

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full path of the document.</td>
</tr>
<tr>
<td>viewType</td>
<td>The view mode to be used. Must be one of the following:</td>
</tr>
<tr>
<td></td>
<td>PDUseThumbs — Displays pages and thumbnail images.</td>
</tr>
<tr>
<td></td>
<td>PDUseNone — Displays only pages.</td>
</tr>
<tr>
<td></td>
<td>PDUseBookmarks — Displays pages and bookmarks.</td>
</tr>
</tbody>
</table>

Returns

ture if the view mode is set successfully, false if the document specified by fullPath does not exist or an unknown view mode is specified.

Related methods

FullMenus
ShortMenus

DocZoomTo

Sets the zoom for a specified document.

Syntax

[DocZoomTo(char* fullPath, char* zoomType, int scale)]

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full path of the file whose zoom to set.</td>
</tr>
<tr>
<td>zoomType</td>
<td>The zoom strategy to use. Must be one of the following:</td>
</tr>
<tr>
<td></td>
<td>AVZoomNoVary — A fixed zoom, such as 100%.</td>
</tr>
<tr>
<td></td>
<td>AVZoomFitPage — Fits the page in the window.</td>
</tr>
<tr>
<td></td>
<td>AVZoomFitWidth — Fits the page's width into the window.</td>
</tr>
<tr>
<td></td>
<td>AVZoomFitVisibleWidth — Fits the page's visible content into the window.</td>
</tr>
<tr>
<td>scale</td>
<td>The magnification specified as a percent (for example, 100 corresponds to a</td>
</tr>
<tr>
<td></td>
<td>magnification of 1.0). scale is used only when zoomType is AVZoomNoVary.</td>
</tr>
</tbody>
</table>
Returns

*false* if the document specified by *fullPath* does not exist, or if *zoomType* has an unknown value. Returns *true* otherwise.

**FileOpen**

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.

**Syntax**

```c
[FileOpen(char* fullPath)]
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full path of the file to be opened.</td>
</tr>
</tbody>
</table>

**Returns**

*true* if the file is opened successfully, *false* otherwise.

**Related methods**

- *CloseAllDocs*
- *DocClose*
- *DocOpen*

**FileOpenEx**

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.

**Syntax**

```c
[FileOpenEx(char* fullPath)]
```
Parameters

fullPath
The full path of the file to be opened.

Returns

true is always returned. The specified file may not actually open.

Related methods

FileOpen
CloseAllDocs
DocClose
DocOpen

FilePrint

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.

Syntax

[FilePrint(char* fullPath)]

Parameters

fullPath
The full path of the file to be printed.

Returns

false if the document specified by fullPath does not exist, true otherwise.

Related methods

DocPrint
FilePrintSilent
FilePrintTo
FilePrintEx

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 lost when printing a large number of files simultaneously.

FilePrintEx is also supported in Adobe Reader.

Syntax

```c
[FilePrintEx(char* fullPath)]
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full path of the file to print.</td>
</tr>
</tbody>
</table>

Returns

true is always returned.

Related methods

- DocPrint
- FileOpenEx
- FilePrint
- FilePrintSilent
- FilePrintSilentEx
- FilePrintTo
- FilePrintToEx

FilePrintSilent

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.

Syntax

```c
[FilePrintSilent(char* fullPath)]
```
Parameters

| fullPath | The full path of the file to be printed. |

Returns

false if the document specified by fullPath does not exist, true otherwise.

Related methods

DocPrint
FilePrint
FilePrintTo

FilePrintSilentEx

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 DDE command that returns true right away and does the action during idle periods. This is to ensure that no DDE commands are lost when printing a large number of files simultaneously.

FilePrintSilentEx is also supported in Adobe Reader.

Syntax

[FilePrintSilentEx(char* fullPath)]

Parameters

| fullPath | The full path of the file to be printed. |

Returns

true is always returned.

Related methods

DocPrint
FileOpenEx
FilePrintEx
FilePrintSilent
FilePrintTo
FilePrintToEx
FilePrintTo

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.

Syntax

[FilePrintTo(char* fullPath, char* printName,
            char* driverName, char* portName)]

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full path of the file to be printed.</td>
</tr>
<tr>
<td>printName</td>
<td>The name of the printer. Required for Windows 95 and later.</td>
</tr>
<tr>
<td>driverName</td>
<td>Printer driver name.</td>
</tr>
<tr>
<td>portName</td>
<td>Port name. Required for Windows NT.</td>
</tr>
</tbody>
</table>

Returns

false if the document specified by fullPath does not exist, true otherwise.

Related methods

DocPrint
FilePrint
FilePrintSilent

FilePrintToEx

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 DDE command that returns true right away and does the action during idle periods. This is to ensure that no DDE commands are lost when printing a large number of files simultaneously.

FilePrintToEx is also supported in Adobe Reader.

Syntax

[FilePrintToEx(char* fullPath, char* printName,
               char* driverName, char* portName)]
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullPath</td>
<td>The full path of the file to be printed.</td>
</tr>
<tr>
<td>printName</td>
<td>The name of the printer. Required for Windows 95 and later.</td>
</tr>
<tr>
<td>driverName</td>
<td>Printer driver name.</td>
</tr>
<tr>
<td>portName</td>
<td>Port name. Required for Windows NT.</td>
</tr>
</tbody>
</table>

Returns

true is always returned.

Related methods

- **DocPrint**
- **FileOpenEx**
- **FilePrintEx**
- **FilePrintSilentEx**
- **FilePrintTo**
- **FilePrintToEx**

**FullMenus**

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.

Syntax

```
[FullMenus()]
```

Returns

true if full menus are set successfully, false otherwise.

Related methods

- **DocSetViewMode**
- **ShortMenus**

**HideToolbar**

Hides the toolbar.
Syntax

[HideToolbar()]

Returns

true if the toolbar is hidden successfully, false otherwise.

Related methods

ShowToolbar

MenuitemExecute

Executes the menu item specified by its language-independent name.

Syntax

[MenuitemExecute(char* menuItemName)]

Parameters

| menuItemName | 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. |

ShortMenus

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.

Syntax

[ShortMenus()]

Returns

true if short menus are set successfully, false otherwise.

Related methods

DocSetViewMode

FullMenus
ShowToolbar

Shows the toolbar.

**Syntax**

```
[ShowToolbar()]
```

**Returns**

- true if the toolbar is shown successfully, false otherwise.

**Related methods**

- [HideToolbar](#)
This chapter describes the supported Apple event objects, with descriptions of each object’s elements and properties, and the supported Apple events.

Objects

Acrobat presents the following objects to the Apple event interface:

- annotation
- application
- bookmark
- conversion
- document
- Link Annotation
- menu
- menu item
- page
- PDF Window
- Text Annotation

**annotation**

An annotation on a page in a PDF file that corresponds to PDAnnot, an internal Acrobat class. This object was formerly known as PDAnnot.

Acrobat also has two built-in annotation objects. For more information, see “Link Annotation” on page 149 and “Text Annotation” on page 154.

**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>Property</td>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</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>Text annotations only. 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>Link annotations only. The page number to appear in the PDF window when the annotation link is activated.</td>
</tr>
<tr>
<td>destination rectangle</td>
<td>a list of small real</td>
<td>Link annotations only. 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>Link annotations only. 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>Text annotations only. The annotation's label.</td>
</tr>
<tr>
<td>open state</td>
<td>Boolean</td>
<td>Text annotations only. 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>Link annotations only. 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

The Acrobat or Adobe Reader application itself.

Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>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 text</td>
<td>The type of the currently active tool. See the Acrobat and PDF Library API 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 no vary, fit page, fit width, fit height, and fit visible width.</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>Property</td>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</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 Esc 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. For example, the following 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>Deprecated.</td>
</tr>
<tr>
<td>note font size</td>
<td>integer</td>
<td>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>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>Deprecated. Set the 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 optimized for the web.</td>
</tr>
<tr>
<td>show splash at startup</td>
<td>Boolean</td>
<td>Determines whether the splash screen is shown at startup.</td>
</tr>
</tbody>
</table>
### Property Class Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>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 <img src="null" alt="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 <img src="null" alt="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:

Deprecated. Use PDF Window instead.

### bookmark

A bookmark on a page in a PDF file. Corresponds to Acrobat's PDBBookmark object.

#### Note:

This object was formerly known as PDBBookmark.

### 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). Note: Set this only after setting fit type.</td>
</tr>
<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>Left Top Zoom — Sets a specified zoom and a specified location on the page.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fit Page — Sets the zoom factor so that the entire page fits into the window.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fit Width — Sets the zoom factor so that the width of the page fits into the window.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fit Height — Sets the zoom factor so that the height of the page fits into the window.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fit Rect — Sets the zoom factor so that the specified rectangle fits into the window.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fit BBox — 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>Fit BB Width — Sets the zoom factor so that the width of the bounding box fits into the window.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fit BB Height — 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

A file type converter that exports PDF files into other formats. Conversions correspond to the list of formats specified in the Acrobat Save As menu. A list of formats can 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><code>best type</code></td>
<td>type class [r/o]</td>
<td>The best descriptor type.</td>
</tr>
<tr>
<td><code>class</code></td>
<td>type class [r/o]</td>
<td>The class.</td>
</tr>
<tr>
<td><code>default type</code></td>
<td>type class [r/o]</td>
<td>The default descriptor type.</td>
</tr>
<tr>
<td><code>index</code></td>
<td>integer [r/o]</td>
<td>The index number of the converter.</td>
</tr>
<tr>
<td><code>name</code></td>
<td>international text</td>
<td>The conversion's description.</td>
</tr>
</tbody>
</table>

Related methods

- `save`

document

Represented a single open document in Acrobat or Adobe Reader.

Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Accessed by</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>page</code></td>
<td>Numeric index. The first page in a document is page 1.</td>
</tr>
<tr>
<td><code>bookmark</code></td>
<td>Name or numeric index.</td>
</tr>
<tr>
<td>PDF Window</td>
<td>An index of 1 or with the same keyword in AppleScript. No document has more than one PDF Window.</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

A file type converter that exports PDF files into EPS format.

Properties

Inherits from PostScript Conversion.

Related methods

g.save

Link Annotation

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.

Properties

Inherits from annotation.

Related methods

delete
perform

menu

A menu in the Acrobat or Adobe Reader menu bar.

Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>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>
</tbody>
</table>
menu item

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

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**

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>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` instead.
PDBookMark

Note: Deprecated. Use bookmark instead.

PDLinkAnnot

Note: Deprecated. Use Link Annotation instead.

PDPAGE

Note: Deprecated. Use page instead.

PDTextAnnot

Note: Deprecated. Use Text Annotation instead.

PDF Window

The area of the Acrobat or Adobe Reader window that displays the contents of a page within the
document. Corresponds to the Acrobat internal AvPageView object. A document that is not visible does
not have a PDF Window.

Note: This object was formerly known as AVPageView.

Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>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>
</tbody>
</table>
### Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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

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 [r/o]</td>
<td>Determines whether to include annotations.</td>
</tr>
<tr>
<td>binary</td>
<td>Boolean [r/o]</td>
<td>Determines whether the output file should be in binary or ASCII text format.</td>
</tr>
</tbody>
</table>
Related methods

**save**

### Text Annotation

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*.

### Properties

Inherits from *annotation*.

### Related methods

**find next note**

**perform**

**replace pages**
Required suite events

The following events are sent by the Finder to all applications:

- open
- print
- quit
- run

Note: Most of these events 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.

Adobe Reader also supports the Required suite events, but no others.

open

Opens a file.

Syntax

open [reference]

Parameters

<table>
<thead>
<tr>
<th>open</th>
<th>The file or files to open.</th>
</tr>
</thead>
</table>

print

Prints one or more files.

Syntax

print [reference]

Parameters

<table>
<thead>
<tr>
<th>print</th>
<th>The file or files to print.</th>
</tr>
</thead>
</table>

quit

Terminates an application. For information on a variant event in the Core suite that accepts options, see quit on page 160.

Syntax

quit
run

Launches the application and invokes its standard startup procedures.

Syntax

run

Core suite events

Acrobat supports the following subset of the Core suite of Apple events:

- close
- count
- delete
- exists
- get
- make
- move
- open
- quit
- save
- set

close

Closes a document.

Syntax

close [reference] saving [constant] linearize [boolean]

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>close</td>
<td>The document to close.</td>
</tr>
<tr>
<td>saving</td>
<td>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.</td>
</tr>
<tr>
<td>linearize</td>
<td>Determines whether the document should be optimized for the web when saving before closing.</td>
</tr>
</tbody>
</table>
Related events

open

count

Counts the number of instances of a particular class.

Syntax

count [type class] of [reference]

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. This keyword is optional.</td>
</tr>
</tbody>
</table>

Note: There is an alternate form using the keyword each in which the parameters are reversed:

    count [reference] each [type class]

Returns

An integer specifying the number of elements.

AppleScript example

    count annotation of document "dev_acro.pdf"
    count menu item of menu "View"
    count document 1 each bookmark

delete

Deletes one or more objects.

Syntax

delete [reference]

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>

Related events

make

exists
exists

Tests whether a specified object exists.

Syntax

exists [reference]

Parameters

exists Object whose existence is checked.

Returns

true if the object exists, false otherwise.

get

Retrieves the value of an object or property.

Syntax

get [reference] as [class]

Note: The keyword get is optional.

Parameters

get The object or property whose value is returned.

as The form in which the data is returned.

Returns

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

Creates a new object.

Syntax

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

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>

Returns

A reference to the newly created object.

Related events

delete
exists

AppleScript example

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

move

Moves a page object.

Syntax

move [reference] to [location reference]
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>

Returns

A reference to the page that is moved.

AppleScript example

```
move page 3 to before page 1
```

open

Opens a document or documents.

Syntax

```
open [list of alias] invisible [boolean] options [string]
```

Parameters

<table>
<thead>
<tr>
<th>open</th>
<th>The document or documents to open.</th>
</tr>
</thead>
<tbody>
<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>

Related events

`close`

quit

Causes the Acrobat application to quit.

Syntax

```
quit saving [constant]
```

Parameters

<table>
<thead>
<tr>
<th>saving</th>
<th>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.</th>
</tr>
</thead>
</table>
AppleScript example

```applescript
quit saving yes
```

save

Saves a document.

Syntax

```applescript
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>save</td>
<td>The document to be saved.</td>
</tr>
<tr>
<td>to</td>
<td>The file into which the document is to be saved. This parameter is optional in Acrobat 6.0 and higher. Specifying the to parameter is equivalent to doing a Save As. You can save a document in one of the supported formats with the using parameter.</td>
</tr>
<tr>
<td>linearize</td>
<td>Determines whether the document should be optimized for the web.</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>

AppleScript example

```applescript
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

Sets an object’s data or properties.

Syntax

```applescript
set [reference] to [anything]
```

Parameters

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

Related events

```applescript
get
```
AppleScript example

set the name of first bookmark to "Chapter 1"

Acrobat application events

This section describes a number of Acrobat API calls for the Apple event interface that are specific to Acrobat applications. The supported events in this suite are:

- bring to front
- clear selection
- close all docs
- create thumbs
- delete pages
- delete thumbs
- execute
- find next note
- find text
- get info
- go backward
- go forward
- goto
- goto next
- goto previous
- insert pages
- is toolbutton enabled
- maximize
- perform
- print pages
- read page down
- read page up
- remove toolbutton
- replace pages
- scroll
- select text
- set info
- zoom

Apple encourages the use of an application's signature as the name of its class for application-specific Apple events. The string `CARO` is the name of the class for Acrobat-specific Apple events:

```c
#define kAEAcrobatViewerClass 'CARO'
```

AppleScript does not need this information.
bring to front

Brings the specified document’s window to the front.

Syntax

\[
\text{bring to front [reference]}
\]

Parameters

| bring to front | The document to be displayed as the active document in the front window. |

AppleScript example

\[
\text{bring to front document "AppleEvt.pdf"}
\]

Apple event ID

\[
\text{kAEBringToFront ('bfrt')}
\]

clear selection

Clears the document’s current selection, if any.

Syntax

\[
\text{clear selection [reference]}
\]

Parameters

| clear selection | The document containing the selection to be cleared |

Related events

\[
\text{select text}
\]

AppleScript example

\[
\text{clear selection document "PLUGINS.PDF"}
\]

Apple event ID

\[
\text{kAEClearSelection ('clsl')}
\]

close all docs

Closes all documents.

Syntax

\[
\text{close all docs saving [constant]}
\]
create thumbs

create thumbs [reference]

Parameters

create thumbs The document in which thumbnails are created.

Related events

delete thumbs

AppleScript example

create thumbs document "roadmap.pdf"

Apple event ID

kAECCreateThumbs ('crtb')
delete pages

Deletes the specified pages in the document.

Syntax

delete pages [reference] first [integer] last [integer]

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. The first page in a document is page 1.</td>
</tr>
<tr>
<td>last</td>
<td>The last page to be deleted.</td>
</tr>
</tbody>
</table>

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

Deletes all thumbnails from the document.

Syntax

delete thumbs [reference]

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>

Related events

create thumbs
AppleScript example

```
delete thumbs document "AppleEvt.pdf"
```

Apple event ID

```
kAEDeleteThumbs ('dltb')
```

**execute**

Executes the specified menu item.

**Syntax**

```
execute [reference]
```

**Parameters**

<table>
<thead>
<tr>
<th>execute</th>
<th>The menu item to execute. See the Acrobat and PDF Library API Reference for a list of menu item names.</th>
</tr>
</thead>
</table>

AppleScript example

```
activate
eexecute menu item "Open"
```

Apple event ID

```
kAEExecute ('exec')
```

**find next note**

Finds and selects the next text note in a document.

**Syntax**

```
find next note [reference] wrap around [boolean]
```

**Parameters**

| find next note | The document in which to find the next text note. |
| wrap around    | 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. |

**Returns**

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

Finds text in a document.

Syntax


Parameters

<table>
<thead>
<tr>
<th>find text</th>
<th>The document to be searched.</th>
</tr>
</thead>
<tbody>
<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>

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

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

Syntax

get info [reference] key [international text]

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>

Returns

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

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

Syntax

go backward [reference]

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>go backward</td>
<td>A PDF Window object</td>
</tr>
</tbody>
</table>
Related events

- go forward
- goto
- goto next
- goto previous

AppleScript example

```
  go backward first PDF Window
```

**Apple event ID**

```
kAEGoBack ('gbck')
```

go forward

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

**Syntax**

```
go forward [reference]
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>go forward</td>
<td>A PDF Window object</td>
</tr>
</tbody>
</table>

Related events

- go backward
- goto
- goto next
- goto previous

AppleScript example

```
  go forward first PDF Window
```

**Apple event ID**

```
kAEGoForward ('gfwd')
```
**goto**

Displays the page that has the specified page number.

**Syntax**

```
goto [reference] page [integer]
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>goto</td>
<td>The <strong>PDF Window</strong> 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>

**Related events**

- go backward
- go forward
- goto next
- goto previous

**AppleScript example**

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

**Apple event ID**

```
kAEGotoPage ('gtpg')
```

**Apple event parameters**

```
keyAEPageNumber ('pg #')
```

**goto next**

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.

**Syntax**

```
goto next [reference]
```

**Parameters**

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

go_backward

go_forward

goto

goto_previous

AppleScript example

goto previous first PDF Window

Apple event ID

kAEGotoPrevPage ('pvpg')

goto previous

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.

Syntax

goto previous [reference]

Parameters

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

Related events

go_backward

go_forward

goto

goto_next

AppleScript example

goto previous first PDF Window

Apple event ID

kAEGotoPrevPage ('pvpg')
insert pages

Inserts one or more pages from one document into another.

Syntax


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>

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')
kAESourceStartPage ('stpg')
keyAENumPages ('nmpg')
keyAEInsertBookmarks ('inbm')

is toolbutton enabled

Determine whether the specified toolbar button is enabled.

Syntax

is toolbutton enabled named [international text]
Parameters

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

Returns

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 ('tbnm')

maximize

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

Syntax

maximize [reference] max size [integer]

Parameters

| maximize | The document whose window is to be resized. |
| max size | If true, the document’s window is set to full size. If false, the window is returned to its original size. |

AppleScript example

maximize document "AppleEvt.pdf" max size false

Apple event ID

kAEMaximize ('maxi')

Apple event parameters

keyAEMaxSize ('mxsz')
perform

Executes a bookmark’s or link annotation’s action.

Syntax

perform [reference]

Parameters

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

AppleScript example

perform last bookmark

Apple event ID

kAEPerform ('prfm')

print pages

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

Syntax


Parameters

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

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')

read page down
   Scrolls forward through the document by one screen.

Syntax
   read page down [reference]

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

Related events
   read page up
   scroll

AppleScript example
   read page down first PDF Window

Apple event ID
   kAEReadPageDown ('pgdn')

read page up
   Scrolls backward through the document by one screen.

Syntax
   read page up [reference]

Parameters
   read page up The PDF Window object to be scrolled.
Related events

read page down
scroll

AppleScript example

read page up first PDFPageWindow

Apple event ID

kAEReadPageUp ("pgup")

remove toolbutton

Removes the specified button from the toolbar.

Syntax

remove toolbutton named [international text]

Parameters

<table>
<thead>
<tr>
<th>named</th>
<th>The name of the toolbar button to be removed. See the Acrobat and PDF Library API Reference for a list of toolbar button names.</th>
</tr>
</thead>
</table>

Related events

is toolbutton enabled

AppleScript example

remove toolbutton named "ZoomIn"

Apple event ID

kAERemoveToolButton ('rmtb')

Apple event parameters

keyAEButtonname ('tbnm')

replace pages

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

Syntax

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>

Related events

- delete pages
- insert pages

AppleScript example

```applescript
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

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

Apple event parameters

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

scroll

Scrolls the view of a page by the specified amount.

Syntax

```applescript
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scroll</td>
<td>The PDF Window object in which to scroll the view.</td>
</tr>
<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>
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

Selects text as specified by either character or word offsets.

Syntax

select text [reference] from words [list of integer] from chars [list of integer]

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>

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')
keyAECharList ('fmch')

set info

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

Syntax

set info [reference] key [international text] value [international text]

Parameters

| set info | The PDF Window in which to set the value of an Info dictionary entry. |
| key      | The Info dictionary key whose value is to be set.               |
| value    | The value to be stored.                                      |

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

Changes the zoom level of the specified PDF Window.

Syntax

zoom [reference] to [small real]
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zoom</td>
<td>The PDF Window object to be zoomed.</td>
</tr>
<tr>
<td>to</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>

AppleScript example

```
zoom first PDFWindow to 150
```

Apple event ID

```
kAEZoomTo ('zmto')
```

Apple event parameters

```
keyAEZoomFactor ('zmft')
```

Miscellaneous events

Acrobat provides an Apple event that does not fall into one of the regular suites: `do script`

**do script**

Executes the specified JavaScript script.

**Syntax**

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

**Parameters**

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

**Returns**

Result of JavaScript execution as text.

**AppleScript example**

```
do script MyJavaScriptFile.js
```
This chapter describes IAC support for the Acrobat Catalog plug-in, which 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 Acrobat Search command to search the entire library quickly. Searches of full-text indexes created using Catalog are faster and more convenient than using the Find command.

For more information on Catalog, see the Acrobat Help and the Acrobat and PDF Library API Reference.

**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.

**Catalog DDE methods**

Clients can connect to the Catalog plug-in through DDE using the service name Acrobat and the topic name Control. This section lists the available DDE methods.

**AppExit**

Exits Acrobat Catalog.

**Syntax**

```
[AppExit()]
```

**Returns**

If true, Catalog exited successfully, otherwise false.

**AppFront**

Brings Catalog to the front.

**Syntax**

```
[AppExit()]
```
FileBuild

Builds an index based on the specified index definition file.

Syntax

[FileBuild(char* fullPath)]

Parameters

fullPath The full path of the file to be opened, including the .pdx extension.

Returns

If true, the file opened successfully, otherwise false.

FileOpen

Opens an index definition file and displays the Edit Index Definition dialog box.

Syntax

[FileOpen(char* fullPath)]

Parameters

fullPath The full path of the file to be opened, including the .pdx extension.

Returns

true if the file opened successfully, otherwise false.

FilePurge

Purges an index definition file.

Syntax

[FilePurge(char* fullPath)]

Parameters

fullPath The full path of the file to be purged, including the .pdx extension.

Returns

true if the file was successfully purged, otherwise false.
The Acrobat Forms plug-in allows a PDF document to act as a form; that is, the Acrobat equivalent of a paper form with fields. This chapter describes the OLE automation methods exported by the Acrobat AcroForm plug-in.

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 Adobe Reader Forms plug-in also does not allow users to save data to the local hard disk. Both Acrobat and Adobe Reader allow Web designers to send data from the form back to a Web server.

**Note:** Forms as used here do not refer to **XObject** forms as defined in the *PDF Reference*.

For more information on Forms, see the Acrobat Help and the *Acrobat and PDF Library API Reference*.

### Forms plug-in OLE automation

The Acrobat Forms plug-in works as an automation server in the Windows environment. Because 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:

1. Instantiate the Acrobat application by using the Visual Basic `CreateObject` method. For example:
   ```vbscript
   CreateObject("AcroExch.App")
   ```
   This causes the Acrobat Forms plug-in to run, at which time it registers its class object with OLE.

2. Instantiate the main exposed object:
   ```vbscript
   CreateObject("AFormAut.App")
   ```

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 `AFORMAULTLib`, and not simply as `Object`.

**Note:** Neither Acrobat nor the Acrobat Forms plug-in are thread-safe, and therefore Acrobat Forms OLE automation uses the single-threading model.

### 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 in this chapter. Others are Acrobat Forms-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.

**Methods**

The Field object has the following methods.

- PopulateListOrComboBox
- SetBackgroundColor
- SetBorderColor
- SetButtonCaption
- SetButtonIcon
- SetExportValues
- SetForegroundColor
- SetJavaScriptAction
- SetResetFormAction
- SetSubmitFormAction
PopulateListOrComboBox

Specifies the item names and optionally exports values for a field of type listbox or combobox.

**Syntax**

```c
void PopulateListOrComboBox ( const VARIANT& arrItems,  
   const VARIANT& arrExportVal);
```

**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. There is a limit of 64K for string data in a combo or list box control on Windows platforms. For Mac OS systems, the limit is 200 entries for the combo or list box control. Using more than these limits degrades performance and makes the control unusable.</td>
</tr>
<tr>
<td>arrExportVal</td>
<td>Optional. An array of strings, the same size as the first parameter, with each element representing an export value. Some of the elements in <code>exportString</code> may be empty strings.</td>
</tr>
</tbody>
</table>

**Exceptions**

Raises `AutErcNotToThisFieldType` if the field is not of type listbox or combobox.

**Related methods**

Add

SetBackgroundColor

Specifies the background color for a field. The background color is used to fill the field's rectangle.

**Syntax**

```c
void SetBackgroundColor (LPCTSTR bstrColorSpace, float GorRorC, float GorM, float BorY, float K);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrColorSpace</td>
<td>Values are defined by using a transparent, gray, RGB or CMYK color space. Valid strings include:</td>
</tr>
<tr>
<td></td>
<td>● T</td>
</tr>
<tr>
<td></td>
<td>● G</td>
</tr>
<tr>
<td></td>
<td>● RGB</td>
</tr>
<tr>
<td></td>
<td>● CMYK</td>
</tr>
<tr>
<td>GorRorC</td>
<td>Used if <code>bstrColorSpace</code> is set to T, G, or RGB. A float range between zero and one inclusive.</td>
</tr>
<tr>
<td>GorM</td>
<td>Used if <code>bstrColorSpace</code> is set to G. A float range between zero and one inclusive.</td>
</tr>
</tbody>
</table>
SetBorderColor

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.

Syntax

```c
void SetBorderColor (LPCTSTR bstrColorSpace, float GorRorC, float GorM, float BorY, float K);
```

Parameters

- **bstrColorSpace**: Values are defined by using a transparent, gray, RGB or CMYK 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.

Related methods

- SetBackgroundColor
- SetForegroundColor

Example

```c
Field.SetBackgroundColor "RGB", 0.7, 0.3, 0.6, 0
```
**Example**

Field.SetBorderColor "RGB", 0.7, 0.3, 0.6, 0

**SetButtonCaption**

The caption to be used for the appearance of a field of type `button`.

**Syntax**

```c
void SetButtonCaption (LPCTSTR bstrFace, LPCTSTR bstrCaption);
```

**Parameters**

| bstrFace   | A string that specifies the face for which the caption will be used. Valid strings include:
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N — Normal appearance</td>
</tr>
<tr>
<td></td>
<td>D — Down appearance</td>
</tr>
<tr>
<td></td>
<td>R — Appearance for rollover</td>
</tr>
<tr>
<td>bstrCaption</td>
<td>The caption for the button.</td>
</tr>
</tbody>
</table>

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`.

**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**

Field.SetButtonCaption "D", "Submit Form"

**SetButtonIcon**

Specifies the icon to be used for the appearance of a field of type `button`.

**Syntax**

```c
void SetButtonIcon (LPCTSTR bstrFace, LPCTSTR bstrFullPath, short pageNum);
```
**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 — Normal appearance</td>
</tr>
<tr>
<td></td>
<td>D — Down appearance</td>
</tr>
<tr>
<td></td>
<td>R — Appearance for rollover</td>
</tr>
<tr>
<td>bstrFullPath</td>
<td>The full path 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>

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.

**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**

```c
Field.SetButtonIcon "N", "c:\Clipart.pdf", 0
```

**SetExportValues**

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).

**Syntax**

```c
void SetExportValues (const VARIANT& arrExportVal);
```

**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>

**Exceptions**

Raises `AutErcNotToThisFieldType` if the field is not of type radio button or checkbox.
Related methods

Example

Dim arrExp(1) As String
arrExp(0) = "CreditCardA"
arrExp(1) = "CreditCardB"
Field.SetExportValues arrExp

SetForegroundColor

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.

The parameters are similar to SetBorderColor and SetBackgroundColor, except that the transparent color space is not allowed.

Syntax

void SetForegroundColor (LPCTSTR bstrColorSpace, float GorRorC, float GorM, float BorY, float K);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrColorSpace</td>
<td>Values are defined by using a transparent, gray, RGB or CMYK color space. Valid strings include:</td>
</tr>
<tr>
<td></td>
<td>● T</td>
</tr>
<tr>
<td></td>
<td>● G</td>
</tr>
<tr>
<td></td>
<td>● RGB</td>
</tr>
<tr>
<td></td>
<td>● CMYK</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>

Related methods

SetBackgroundColor

SetBorderColor

Example

Field.SetForegroundColor "CMYK", 0.25, 0.25, 0.25, 0.1
SetJavaScriptAction

Sets the action of the field to be of type JavaScript. When using SetJavaScriptAction within Visual Basic, you can use Chr(13) to add a <CR>, and Chr(9) for tabs, so that the function is well formatted.

Syntax

```c
void SetJavaScriptAction (LPCTSTR bstrTrigger, LPCTSTR bstrTheScript);
```

Parameters

- **bstrTrigger**: A string that specifies the trigger for the action. Valid strings include:
  - up
  - down
  - enter
  - exit
  - calculate
  - validate
  - format
  - keystroke

- **bstrTheScript**: The script itself.

If the trigger is calculate, an entry is added at the end of the calculation order array (see the CalcOrderIndex property).

Calculation script

A simple calculate script is 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 scripts

The following scripts and formats can be used for the format and keystroke triggers:

<table>
<thead>
<tr>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFDate_KeystrokeEx(cFormat)</td>
<td>cFormat is one of:</td>
</tr>
<tr>
<td>AFDate_Format(cFormat)</td>
<td>&quot;m/d&quot;, &quot;m/d/yy&quot;, &quot;mm/dd/yy&quot;, &quot;mm/yy&quot;, &quot;d-mmm&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;d-mmm-yy&quot;, &quot;dd-mmm-yy&quot;, &quot;yy-mm-dd&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;mmm-yy&quot;, &quot;mmmm-yy&quot;, &quot;mmm d, yyyy&quot;, &quot;mmmm d, yyyy&quot;, &quot;m/d/yy hh:mm tt&quot;, &quot;m/d/yy HH:MM&quot;</td>
</tr>
<tr>
<td>AFTime_Keystroke(ptf)</td>
<td>ptf is the time format:</td>
</tr>
<tr>
<td>AFTime_Format(ptf)</td>
<td>0 = 24HR_MM [14:30 ]</td>
</tr>
<tr>
<td></td>
<td>1 = 12HR_MM [2:30 PM ]</td>
</tr>
<tr>
<td></td>
<td>2 = 24HR_MM_SS [14:30:15 ]</td>
</tr>
<tr>
<td></td>
<td>3 = 12HR_MM_SS [2:30:15 PM ]</td>
</tr>
</tbody>
</table>
SetResetFormAction

Sets the action of the field to be of type ResetForm.

**Syntax**

```c
void SetResetFormAction (LPCTSTR bstrTrigger, long theFlags, const VARIANT& arrFields);
```

**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>up — Mouse up</td>
</tr>
<tr>
<td></td>
<td>down — Mouse down</td>
</tr>
<tr>
<td></td>
<td>enter — Mouse enter</td>
</tr>
<tr>
<td></td>
<td>exit — 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>
</tbody>
</table>
SetSubmitFormAction

Sets the action of the field to be of type SubmitForm.

Syntax

```c
void SetSubmitFormAction (LPCTSTR bstrTrigger, LPCTSTR bstrTheURL, long theFlags, const VARIANT& arrFields);
```

Parameters

- `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. See the PDF Reference to learn how the binary value of this `long` is interpreted.
- `arrFields` Optional. If specified, 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 way to cause all their children to be included in the submission.
  - If not specified, the created action does not include a `/Fields` key.

Properties

The Field object has the following properties:

- **Alignment**
- **BorderStyle**
- **BorderWidth**
- **ButtonLayout**
- **CalcOrderIndex**
- **CharLimit**
● DefaultValue
● Editable
● Highlight
● IsHidden
● IsMultiline
● IsPassword
● IsReadOnly
● IsRequired
● IsTerminal
● Name
● NoViewFlag
● PrintFlag
● Style
● TextFont
● TextSize
● Type
● Value

Alignment

The text alignment of a text field. Valid alignments are:

- left
- center
- right

Syntax

[get/set] String

Returns

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**

The border style for a field. Valid border styles include solid, dashed, beveled, inset, and underline.

**Syntax**

`[get/set] String`

**Returns**

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**

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.

**Syntax**

`[get/set] short`

**Returns**

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**

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, therefore the field can be non-terminal.

**Syntax**

\[get/set\] short

**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**

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 *JavaScript for Acrobat API Reference*.

**Syntax**

\[get/set\] short

**Example**

Set \( F_1 = \text{Fields("SubTotal"}) \)
Set \( F_2 = \text{Fields("Total")} \)
\( F_2.CalcOrderIndex = F_1.CalcOrderIndex + 1 \)
CharLimit

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.

**Syntax**

[get/set] short

**Exceptions**

If the field is not of type text, an exception `AutErcNotToThisFieldType` is returned.

DefaultValue

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.

**Syntax**

[get/set] String

**See also**

Value

Editable

Determines 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.

**Syntax**

[get/set] Boolean

**Exceptions**

Returns an exception of `AutErcNotToThisFieldType` if the field is not of type combobox.

**Example**

Field.Editable = False
Highlight

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.

Syntax

[get/set] String

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

Determines whether the field is hidden or visible to the user. If the value is **true** the field is invisible, and **false** indicates that the field is visible.

During get operations, 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 hidden flag for the first child, whichever annotation that happens to be.

During set operations, the property is propagated to any child annotations underneath, therefore a field can be non-terminal.

Syntax

[get/set] Boolean

Example

'Hide "name.last"
Set Field = Fields("name.last")
Field.IsHidden = True
IsMultiline

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.

Syntax

[get/set] Boolean

Exceptions

If the field is not of type text, an exception AutErcNotToThisFieldType is returned.

Example

Field.IsMultiline = True

IsPassword

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.

Syntax

[get/set] Boolean

Exceptions

If the field is not of type text, an exception AutErcNotToThisFieldType is returned.

Example

Field.IsPassword = True

IsReadOnly

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 click 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.

Syntax

[get/set] Boolean
IsRequired

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.

**Syntax**

[get/set] Boolean

IsTerminal

true if the field is terminal, otherwise false.

**Syntax**

[read-only] Boolean

**Example**

Dim Field As AFORMAUTLib.Field
Dim bTerminal As Boolean

'bTerminal should be True
bTerminal = Field.IsTerminal

Name

The fully qualified name of the field. It is the default member of the Field interface.

**Syntax**

[read-only] String

NoViewFlag

Determines whether a given field prints but does not 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.

**Syntax**

[get/set] Boolean
PrintFlag

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.

Syntax
[get/set] Boolean

Style

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, therefore a field can be non-terminal.

Syntax
[get/set] String

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"
TextFont

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.

Syntax

[get/set] String

Example

Field.TextFont = "Times-BoldItalic"

TextSize

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.

Syntax

[get/set] short

Example

Field.TextSize = 18

Type

The type of the field as a string. Valid types that are returned:

- text
- button
- combobox
- listbox
- checkbox
- radiobutton
- signature
Syntax

[read-only] String

Example

Set Field = Fields("name.last")
'Should print "name.last"
print Field
' Should print the type of field. Example,
' "text"
print Field.Type

Value

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 on page 188). 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.

Syntax

[get/set] String

Example

Dim arrExp(1) As String
arrExp(0) = "CreditCardV"
arrExp(1) = "CreditCardM"
Field.SetExportValues arrExp
Field.Value = arrExp(0)

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.
Methods

The **Fields** object has the following methods.

- **Add**
- **AddDocJavascript**
- **ExecuteThisJavascript**
- **ExportAsPDF**
- **ExportAsHtml**
- **ImportAnFDF**
- **Remove**

**Add**

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.

**Syntax**

```cpp
LPDISPATCH Add (LPCTSTR bstrFieldName, LPCTSTR bstrFieldType, short pageNum, float left, float top, float right, float bottom);
```

**Parameters**

<table>
<thead>
<tr>
<th><strong>bstrFieldName</strong></th>
<th>The fully-qualified name of the field.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bstrFieldType</strong></td>
<td>Field type for the newly created field. Valid types are:</td>
</tr>
<tr>
<td></td>
<td>● text</td>
</tr>
<tr>
<td></td>
<td>● button</td>
</tr>
<tr>
<td></td>
<td>● combobox</td>
</tr>
<tr>
<td></td>
<td>● listbox</td>
</tr>
<tr>
<td></td>
<td>● checkbox</td>
</tr>
<tr>
<td></td>
<td>● radio button</td>
</tr>
<tr>
<td></td>
<td>● signature</td>
</tr>
</tbody>
</table>

You must use the quotation marks. See the sample code below.

When creating list or combo boxes, there is a limit of 64K for string data on Windows platforms. Mac OS 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.
AddDocJavascript

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 well formatted.

**Syntax**

```cpp
void AddDocJavascript (LPCTSTR bstrScriptName, LPCTSTR bstrTheScript);
```

**Parameters**

- **bstrScriptName**: The name of the function to be added to the document.
- **bstrTheScript**: The definition to be added to the document.

**Example**

```
'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**

Executes the specified JavaScript script.

**Syntax**

```cpp
CString ExecuteThisJavascript (LPCTSTR bstrTheScript);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrTheScript</td>
<td>A string containing a JavaScript script, which is executed by Acrobat in the context of the currently active document. See the JavaScript for Acrobat API Reference for information on event level values.</td>
</tr>
</tbody>
</table>

**Returns**

Returns a result by assigning it to event value.

**Related methods**

*AddDocJavascript*

**Example**

```cpp
Fields.ExecuteThisJavaScript "var f = _this.getField("myButton"); f.delay = _false;"
```

To get the returns in Visual Basic:

```vbnet
Dim cSubmitName As String
    cSubmitName = Fields.ExecuteThisJavaScript
        "event.value = this.getField("myField").submitName;"
```

**ExportAsFDF**

Exports the data as FDF from an Acrobat form.

**Syntax**

```cpp
void ExportAsFDF (LPCTSTR bstrFullPath, LPCTSTR bstrSubmitButton, BOOL bEmptyFields, const VARIANT& arrFields);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrFullPath</td>
<td>A full path 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 can specify an empty string.</td>
</tr>
</tbody>
</table>
### Related methods

- [ImportAnFDF](#)
- [ExportAsHtml](#)

### Example

```vba
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

Exports the data as HTML from an Acrobat form. This method is similar to [ExportAsPDF](#). The only difference is that the form data is exported in URL-encoded format.

#### Syntax

```vba
void ExportAsHtml (LPCTSTR bstrFullPath, LPCTSTR bstrSubmitButton, 
                BOOL bEmptyFields, const VARIANT& arrFields);
```

#### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrFullPath</td>
<td>A full path 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</td>
</tr>
<tr>
<td></td>
<td>include it in the FDF file, as if it had been used to trigger a SubmitForm</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>the produced FDF file.</td>
</tr>
<tr>
<td>arrFields</td>
<td>Optional. An array of strings representing the fully-qualified names of the</td>
</tr>
<tr>
<td></td>
<td>fields to include in the FDF file. This array may include the names of</td>
</tr>
<tr>
<td></td>
<td>non-terminal fields, which is a fast and easy way to cause all their children</td>
</tr>
<tr>
<td></td>
<td>to be included in the FDF file.</td>
</tr>
</tbody>
</table>

#### Related methods

- [ExportAsPDF](#)
ImportAnFDF

Imports the FDF file into an Acrobat form.

**Syntax**

```c
void ImportAnFDF (LPCTSTR bstrFullPath);
```

**Parameters**

- `bstrFullPath`: The full path of the file containing the FDF file to be imported.

**Related methods**

- [ExportAsFDF](#)

---

Remove

Removes a field from the Acrobat Form and from the Fields collection.

**Syntax**

```c
void Remove (LPCTSTR bstrFieldName);
```

**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.

**Related methods**

- [Add](#)

**Example**

```
'Remove fields you no longer used.
Fields.Remove("MyOldField")
```

---

Properties

The Fields object has the following properties.

- **Count**
- **Item**
- **_NewEnum**

**Count**

The number of items in the collection.

**Syntax**

```c
[read-only] long
```
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

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, item is the property invoked if the object name is specified by itself without a property or a method in the controller script.

Syntax

[read-only] IDispatch*

Example

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

The IEnumVariant enumerator for the collection.

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

Syntax

[read-only] IUnknown*
This chapter describes IAC support for the Acrobat Search plug-in, which 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 in the form of DDE messages in Windows and Apple events in Mac OS. These messages and events allow remote clients to submit search queries and manipulate a list of indexes (the list of indexes is referred to as the shelf).

For more information on the Search plug-in, see the Acrobat Help and the Acrobat and PDF Library API Reference.

Search plug-in using DDE

A client can connect to the Search plug-in with 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 the table "Query language type constants" on page 211). It is not possible to choose another parser or to set word options using the simple query item.

```c
hszItemName = DdeCreateStringHandle(id, "Query", 0);
DdeClientTransaction(qd, nLen, hConv, hszItemName, CF_TEXT, XTYP_POKE, 1000, &dwResult);
DdeDisconnect(hConv)
```

Query item

A query has the item name "Query". When using query, a QueryData structure is used. 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;
```

**Query options**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>qlt</code></td>
<td>The query language type. Must be one of the values shown in “Query language type constants” on page 211.</td>
</tr>
<tr>
<td><code>bOverrideWordOptions</code></td>
<td>Indicates that the client wishes to use different word options than those currently set by the user.</td>
</tr>
<tr>
<td><code>nWordOptions</code></td>
<td>The word options. Must be an OR of the values shown in “Word option bit-flag constants” on page 211.</td>
</tr>
<tr>
<td><code>nMaxDocs</code></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><code>nSortOffsets</code></td>
<td>A list of offsets into the cData chunk. Each offset points to a NULL-terminated string containing the field name.</td>
</tr>
<tr>
<td></td>
<td>This value has no effect in Acrobat 6.0 or later, because sort options are not valid.</td>
</tr>
<tr>
<td><code>nQueryOffset</code></td>
<td>An offset into the cData chunk that points to a NULL-terminated string containing the query to execute.</td>
</tr>
<tr>
<td><code>nNumSorts</code></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>This value has no effect in Acrobat 6.0 or later, because sort options are not valid.</td>
</tr>
<tr>
<td><code>bSortWays</code></td>
<td>A list of sort order flags, one for each sort field. true indicates an ascending sort, and false indicates a descending sort.</td>
</tr>
<tr>
<td></td>
<td>This value has no effect in Acrobat 6.0 or later, because sort options are not valid.</td>
</tr>
</tbody>
</table>
Query language type constants

- **QLangType_Simple**: Allows only simple phrase searches; does not allow Boolean searching. This query type does not work in the DDE interface of the Search plug-in shipped with version 2.0 of Acrobat.

- **QLangType_CQL**: Allows Boolean searches using **AND**, **OR**, and **NOT**, as described in the Acrobat Search plug-in’s online help file.

- **QLangType_Passthrough**: The Verity BooleanPlus query language. Contact Verity for further information on this language.

Word option bit-flag constants

- **QPON_Case**: The search is case-sensitive.

- **QPON_Stemming**: Find not only the specified word, but other words that have the same stem. For example, run and ran have the same stem.

- **QPON_SoundsLike**: Find not only the specified word, but other words that sound like it.

- **QPON_Thesaurus**: Find not only the specified word, but other words that have the same meaning.

- **QPON_Proximity**: Consider the proximity of results when using the **AND** operator to look for more than one word in a document. Without this option, **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 this option, however, **AND** terms must be within two or three pages of each other to be found. Also, the closer **AND** terms appear together, the higher the relevance ranking of the document that contains them.

- **QPON_Refine**: Do not search the entire list of indexes, but only the documents that matched the previous search. This is used to refine the results of the previous search.

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");
```
Manipulating indexes through DDE

After a connection has been made, a single poke transaction can add, delete, add, or remove indexes. 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;
```

**Options**

<table>
<thead>
<tr>
<th>eAction</th>
<th>The operation to be performed on the index. Must be one of values listed in &quot;Index operation selectors&quot; on page 212.</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>

**Index operation selectors**

<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:

```c
nSize = sizeof(IndexData) + li + lt;
id = (IndexData *)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");
```
### Search plug-in using Apple events

The Search plug-in supports the Apple events described in this section.

#### SearchAddIndex

*Adds a specified index to the shelf.*

**Apple event ID**

```cpp
kSearchAddIndex ('addx')
```

**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>Mac OS full path representing an index, of the form:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MyDisk:TopFolder:BottomFolder:Strange.pdx</td>
<td></td>
</tr>
<tr>
<td>kFlagTag ('Flag'), typeLongInteger</td>
<td>Index flags. See SearchGetIndexFlags on page 216 for a description. The kIndexAvailable flag should always be set.</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

```cpp
kIndexTag ('SixP'), typeLongInteger
```

An opaque void* representing an index. Returns NULL if failure.

Returns

```cpp
#define kIndexExists ((SearchIndexPtr)-1)
```

if the index already exists in the index list. If the index already exists, you can retrieve it using SearchGetIndexByPath on page 215.

#### SearchCountIndexList

*Gets the number of indexes currently on the shelf.*

**Apple event ID**

```cpp
kSearchCountIndexList ('cidx')
```

**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>
</tbody>
</table>
## Returns

`kIndexListTag ('SilP'), typeLongInteger`

Number of indexes on the shelf (`kIndexListTag` here is not semantically correct, but works).

### SearchDoQuery

Executes a specified query, using the set of indexes currently on the shelf. The search results are displayed in the Acrobat Search plug-in's Results window.

### Apple event ID

`kSearchDoQuery ('kwry')`

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `kQueryStringTag ('Quryv'), 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 "Word options for Apple events" on page 215. 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 `maxDocs` are displayed. `maxDocs` cannot be greater than 999.
Word options for Apple events

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kWordOptionCase</td>
<td>The search is case-sensitive.</td>
</tr>
<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 AND operator to look for more than one word in a document. Without kWordOptionProximity, 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 kWordOptionProximity, however, AND terms must be within two or three pages of each other to be found. Also, with kWordOptionProximity, the closer 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 indexes, 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

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')

Parameters

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kIndexListTag</td>
<td>'SilP', typeLongInteger An opaque void* representing the shelf, obtained from SearchGetIndexList.</td>
</tr>
<tr>
<td>kPathTag</td>
<td>'Path', typeChar Mac OS full path representing an index, of the form: MyDisk:TopFolder:BottomFolder:Strange.pdx</td>
</tr>
</tbody>
</table>

Returns

kIndexTag ('SixP'), typeLongInteger

An opaque void* representing an index. Returns NULL if the specified index is gone.
SearchGetIndexFlags

Get the flags for an index.

Apple event ID

kSearchGetIndexFlags ('gfdx')

Parameters

kIndexTag ('SixP'), typeLongInteger
An opaque void* representing an index.

Returns

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

Gets a list of the indexes currently on the shelf.

Apple event ID

kSearchGetIndexList ('gidx')

Returns

kIndexListTag ('SilP'), typeLongInteger

An opaque void* representing the list of indexes 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 indexes on the shelf, and so forth.

SearchGetIndexPath

Gets the full path to an index.

Apple event ID

kSearchGetIndexPath ('gpdx')
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.

Returns

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

Gets the title of an index.

Apple event ID

kSearchGetIndexTitle ('gtdx')

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.

Returns

kTitleTag ('Title'), typeChar

A NULL-terminated character string representing the title of the index. If there is no title, it returns the index's path. Returns an empty string if the requested index is not valid.

SearchGetNthIndex

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')
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kIndexListTag ('SilP'), typeLongInteger</td>
<td>An opaque void* representing the shelf, obtained from SearchGetIndexList.</td>
</tr>
<tr>
<td>kNthIndexTag ('Enth'), typeLongInteger</td>
<td>The index to get. The first index on the shelf is index zero.</td>
</tr>
</tbody>
</table>

Returns

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kIndexTag ('SixP'), typeLongInteger</td>
<td>An opaque void* representing an index. Returns NULL if the n&lt;sup&gt;th&lt;/sup&gt; index is gone.</td>
</tr>
</tbody>
</table>

SearchRemoveIndex

Removes the specified index from the shelf.

Apple event ID

kSearchRemoveIndex ('rmdx')

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kIndexListTag ('SilP'), typeLongInteger</td>
<td>An opaque void* representing the shelf, obtained from SearchGetIndexList.</td>
</tr>
<tr>
<td>kIndexTag ('SixP'), typeLongInteger</td>
<td>An opaque void* representing the index to be removed. The index may be obtained using SearchGetIndexByPath, SearchGetNthIndex, or SearchAddIndex.</td>
</tr>
</tbody>
</table>

SearchSetIndexFlags

Sets the flags for an index.

Apple event ID

kSearchSetIndexFlags ('sfdx')

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kIndexTag ('SixP'), typeLongInteger</td>
<td>An opaque void* representing an index.</td>
</tr>
<tr>
<td>kFlagTag ('Flag'), typeLongInteger</td>
<td>Index flags. See the description in SearchGetIndexFlags. In practice, kIndexAvailableFlag should always be set.</td>
</tr>
</tbody>
</table>
Returns

kFlagTag ('Flag'), typeLongInteger

Index flags. See the description in “SearchGetIndexFlags” on page 216. This value is returned because it is possible for a request to set a flag to fail.

Search lists

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 box.</td>
</tr>
<tr>
<td>AcroSrch:Indexes</td>
<td>Displays the Index dialog box.</td>
</tr>
<tr>
<td>AcroSrch:Results</td>
<td>Displays the Results dialog box.</td>
</tr>
<tr>
<td>AcroSrch:Assist</td>
<td>Displays the Word Assistant dialog box.</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 box.</td>
</tr>
<tr>
<td>AcroSrch:Results</td>
<td>Displays the Acrobat Search plug-in’s search results dialog box.</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>
This chapter describes the coordinate systems used by IAC: user space and device space.

**User space**

The user space is the coordinate system used within PDF files. In the IAC interface, it is used for most PD layer objects (that is, objects such as **PDBookmark** whose names begin with “PD”). The following graphic shows the user space coordinate system. The orientation, origin, and scale of the user space coordinate system can be changed by operators in the page description in a PDF file.

The default user space is the user space coordinate system in effect immediately before each page begins drawing. The origin of this coordinate system is the lower left corner of a page's media box. The x-coordinate increases to the right, and the y-coordinate increases upward. One unit in the default user space is 1/72 of an inch.
Device space

The device space specifies coordinates in screen pixels, as shown in the following graphic. It is used in the AV layer of the IAC interface (that is, objects such as AVDoc whose names begin with “AV”).

The origin of the device space coordinate system is at the upper left corner of the visible page on the screen (that is, the upper left corner of the white part of the page). The x-coordinate increases to the right, and the y-coordinate increases downward.

The upper left corner of the visible page is determined by the intersection of a page's PDF crop box and media box. As a result, the device space coordinate system changes if the cropping on a page changes.
## Index

<table>
<thead>
<tr>
<th>A</th>
<th>AcquirePage method 71</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acrobat application events 162</td>
</tr>
<tr>
<td></td>
<td>AcroExch.App 14</td>
</tr>
<tr>
<td></td>
<td>AcroExch.AVDoc 30</td>
</tr>
<tr>
<td></td>
<td>AcroExch.AVPageView 47</td>
</tr>
<tr>
<td></td>
<td>AcroExch.HiliteList 56</td>
</tr>
<tr>
<td></td>
<td>AcroExch.PDAnnot 56</td>
</tr>
<tr>
<td></td>
<td>AcroExch.PDBookmark 66</td>
</tr>
<tr>
<td></td>
<td>AcroExch.PDDoc 69</td>
</tr>
<tr>
<td></td>
<td>AcroExch.PDPage 86</td>
</tr>
<tr>
<td></td>
<td>AcroExch.PDTextSelect 98</td>
</tr>
<tr>
<td></td>
<td>AcroExch.Point 102</td>
</tr>
<tr>
<td></td>
<td>AcroExch.Rect 102</td>
</tr>
<tr>
<td></td>
<td>AcroExch.Time 104</td>
</tr>
<tr>
<td></td>
<td>Add method 56, 203</td>
</tr>
<tr>
<td></td>
<td>AddAnnot method 87</td>
</tr>
<tr>
<td></td>
<td>AddDocJavascript method 204</td>
</tr>
<tr>
<td></td>
<td>AddNewAnnot method 88</td>
</tr>
<tr>
<td></td>
<td>Adobe Reader</td>
</tr>
<tr>
<td></td>
<td>Apple events 155</td>
</tr>
<tr>
<td></td>
<td>DDE support 121</td>
</tr>
<tr>
<td></td>
<td>AFormApp object 184</td>
</tr>
<tr>
<td></td>
<td>Alignment property 193</td>
</tr>
<tr>
<td></td>
<td>annotation object 141</td>
</tr>
<tr>
<td></td>
<td>App object 14</td>
</tr>
<tr>
<td></td>
<td>AppExit message 121</td>
</tr>
<tr>
<td></td>
<td>AppExit method 181</td>
</tr>
<tr>
<td></td>
<td>AppFront method 181</td>
</tr>
<tr>
<td></td>
<td>AppHide message 122</td>
</tr>
<tr>
<td></td>
<td>Apple events and objects 141</td>
</tr>
<tr>
<td></td>
<td>application object 143</td>
</tr>
<tr>
<td></td>
<td>AppShow message 122</td>
</tr>
<tr>
<td></td>
<td>AVDoc object 30</td>
</tr>
<tr>
<td></td>
<td>AVPageView object 47, 145</td>
</tr>
<tr>
<td></td>
<td>AxAcroPDF object 106</td>
</tr>
<tr>
<td></td>
<td>AxAcroPDFLib.AxAcroPDF 106</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>bookmark object 145</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BorderStyle property 194</td>
</tr>
<tr>
<td></td>
<td>BorderWidth property 194</td>
</tr>
<tr>
<td></td>
<td>Bottom property 103</td>
</tr>
<tr>
<td></td>
<td>bring to front event 163</td>
</tr>
<tr>
<td></td>
<td>BringToFront method 31</td>
</tr>
<tr>
<td></td>
<td>ButtonLayout property 195</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>CalcOrderIndex property 195</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Catalog plug-in 181</td>
</tr>
<tr>
<td></td>
<td>CharLimit property 196</td>
</tr>
<tr>
<td></td>
<td>clear selection event 163</td>
</tr>
<tr>
<td></td>
<td>ClearFlags method 71</td>
</tr>
<tr>
<td></td>
<td>ClearSelection method 32</td>
</tr>
<tr>
<td></td>
<td>close all docs event 163</td>
</tr>
<tr>
<td></td>
<td>close event 156</td>
</tr>
<tr>
<td></td>
<td>Close method 32, 72</td>
</tr>
<tr>
<td></td>
<td>CloseAllDocs message 122</td>
</tr>
<tr>
<td></td>
<td>CloseAllDocs method 16</td>
</tr>
<tr>
<td></td>
<td>conversion object 147</td>
</tr>
<tr>
<td></td>
<td>coordinate systems 221</td>
</tr>
<tr>
<td></td>
<td>CopyToClipboard method 88</td>
</tr>
<tr>
<td></td>
<td>Core suite events 156</td>
</tr>
<tr>
<td></td>
<td>count event 157</td>
</tr>
<tr>
<td></td>
<td>Count property 207</td>
</tr>
<tr>
<td></td>
<td>Create method 72</td>
</tr>
<tr>
<td></td>
<td>create thumbs event 164</td>
</tr>
<tr>
<td></td>
<td>CreatePageHilite method 89</td>
</tr>
<tr>
<td></td>
<td>CreateTextSelect method 73</td>
</tr>
<tr>
<td></td>
<td>CreateThumbs method 74</td>
</tr>
<tr>
<td></td>
<td>CreateWordHilite method 90</td>
</tr>
<tr>
<td></td>
<td>CropPage method 91</td>
</tr>
<tr>
<td></td>
<td>CropPages method 74</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D</th>
<th>Date property 104</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DDE</td>
</tr>
<tr>
<td></td>
<td>Adobe Reader support 121</td>
</tr>
<tr>
<td></td>
<td>messages 120</td>
</tr>
<tr>
<td></td>
<td>DefaultValue property 196</td>
</tr>
<tr>
<td></td>
<td>delete event 157</td>
</tr>
<tr>
<td></td>
<td>delete pages event 165</td>
</tr>
<tr>
<td></td>
<td>delete thumbs event 165</td>
</tr>
<tr>
<td></td>
<td>DeletePages method 75</td>
</tr>
<tr>
<td></td>
<td>DeleteThumbs method 75</td>
</tr>
<tr>
<td></td>
<td>Destroy method 66, 98</td>
</tr>
<tr>
<td></td>
<td>device space 222</td>
</tr>
<tr>
<td></td>
<td>DevicePointToPage method 47</td>
</tr>
<tr>
<td></td>
<td>do script event 180</td>
</tr>
<tr>
<td></td>
<td>DocClose message 123</td>
</tr>
<tr>
<td></td>
<td>DocDeletePages message 123</td>
</tr>
<tr>
<td></td>
<td>DocFind message 124</td>
</tr>
<tr>
<td></td>
<td>DocGoTo message 125</td>
</tr>
<tr>
<td></td>
<td>DocGoToNameDest message 125</td>
</tr>
<tr>
<td></td>
<td>DocInsertPages message 125</td>
</tr>
<tr>
<td></td>
<td>DocOpen message 126</td>
</tr>
<tr>
<td></td>
<td>DocPageDown message 127</td>
</tr>
<tr>
<td></td>
<td>DocPageLeft message 127</td>
</tr>
<tr>
<td></td>
<td>DocPageRight message 128</td>
</tr>
<tr>
<td></td>
<td>DocPageUp message 128</td>
</tr>
<tr>
<td></td>
<td>DocPrint message 129</td>
</tr>
<tr>
<td></td>
<td>DocReplacePages message 129</td>
</tr>
<tr>
<td></td>
<td>DocSave message 130</td>
</tr>
<tr>
<td></td>
<td>DocSaveAs message 130</td>
</tr>
<tr>
<td></td>
<td>DocScrollTo message 131</td>
</tr>
<tr>
<td>Method/Property/Event</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------</td>
</tr>
<tr>
<td>DocSetViewMode message</td>
<td>132</td>
</tr>
<tr>
<td>document object</td>
<td>147</td>
</tr>
<tr>
<td>DocZoomTo message</td>
<td>132</td>
</tr>
<tr>
<td>DoGoBack method</td>
<td>48</td>
</tr>
<tr>
<td>DoGoForward method</td>
<td>48</td>
</tr>
<tr>
<td>Draw method</td>
<td>91</td>
</tr>
<tr>
<td>DrawEx method</td>
<td>92</td>
</tr>
<tr>
<td>dual interfaces</td>
<td>14</td>
</tr>
<tr>
<td>E</td>
<td></td>
</tr>
<tr>
<td>Editable property</td>
<td>196</td>
</tr>
<tr>
<td>EPS Conversion object</td>
<td>149</td>
</tr>
<tr>
<td>events</td>
<td></td>
</tr>
<tr>
<td>Acrobat application</td>
<td>162</td>
</tr>
<tr>
<td>Core suite</td>
<td>156</td>
</tr>
<tr>
<td>miscellaneous</td>
<td>180</td>
</tr>
<tr>
<td>Required suite</td>
<td>155</td>
</tr>
<tr>
<td>exceptions, Forms plug-in</td>
<td>183</td>
</tr>
<tr>
<td>execute event</td>
<td>166</td>
</tr>
<tr>
<td>ExecuteThisJavascript method</td>
<td>205</td>
</tr>
<tr>
<td>exists event</td>
<td>158</td>
</tr>
<tr>
<td>Exit method</td>
<td>16</td>
</tr>
<tr>
<td>ExportAsFDF method</td>
<td>205</td>
</tr>
<tr>
<td>ExportAsHtml method</td>
<td>206</td>
</tr>
<tr>
<td>F</td>
<td></td>
</tr>
<tr>
<td>Field object</td>
<td>184</td>
</tr>
<tr>
<td>Fields collection</td>
<td>202</td>
</tr>
<tr>
<td>FileBuild method</td>
<td>182</td>
</tr>
<tr>
<td>FileOpen message</td>
<td>133</td>
</tr>
<tr>
<td>FileOpen method</td>
<td>182</td>
</tr>
<tr>
<td>FileOpenEx message</td>
<td>133</td>
</tr>
<tr>
<td>FilePrint message</td>
<td>134</td>
</tr>
<tr>
<td>FilePrintEx message</td>
<td>135</td>
</tr>
<tr>
<td>FilePrintSilent message</td>
<td>135</td>
</tr>
<tr>
<td>FilePrintSilentEx message</td>
<td>136</td>
</tr>
<tr>
<td>FilePrintTo message</td>
<td>137</td>
</tr>
<tr>
<td>FilePrintToEx message</td>
<td>137</td>
</tr>
<tr>
<td>FilePurge method</td>
<td>182</td>
</tr>
<tr>
<td>find next note event</td>
<td>166</td>
</tr>
<tr>
<td>find text event</td>
<td>167</td>
</tr>
<tr>
<td>FindText method</td>
<td>33</td>
</tr>
<tr>
<td>Forms plug-in</td>
<td>183</td>
</tr>
<tr>
<td>FullMenus message</td>
<td>138</td>
</tr>
<tr>
<td>G</td>
<td></td>
</tr>
<tr>
<td>get event</td>
<td>158</td>
</tr>
<tr>
<td>get info event</td>
<td>168</td>
</tr>
<tr>
<td>GetActiveDoc method</td>
<td>17</td>
</tr>
<tr>
<td>GetActiveTool method</td>
<td>17</td>
</tr>
<tr>
<td>GetAnnot method</td>
<td>93</td>
</tr>
<tr>
<td>GetAnnotIndex method</td>
<td>93</td>
</tr>
<tr>
<td>GetAperture method</td>
<td>49</td>
</tr>
<tr>
<td>GetAVDoc method</td>
<td>18, 49</td>
</tr>
<tr>
<td>GetAVPageView method</td>
<td>33</td>
</tr>
<tr>
<td>GetBoundingRect method</td>
<td>99</td>
</tr>
<tr>
<td>GetByTitle method</td>
<td>67</td>
</tr>
<tr>
<td>GetColor method</td>
<td>57</td>
</tr>
<tr>
<td>GetContents method</td>
<td>58</td>
</tr>
<tr>
<td>GetDate method</td>
<td>58</td>
</tr>
<tr>
<td>GetDoc method</td>
<td>49, 94</td>
</tr>
<tr>
<td>GetFileName method</td>
<td>76</td>
</tr>
<tr>
<td>GetFlags method</td>
<td>76</td>
</tr>
<tr>
<td>GetFrame method</td>
<td>18, 34</td>
</tr>
<tr>
<td>GetInfo method</td>
<td>77</td>
</tr>
<tr>
<td>GetInstanceID method</td>
<td>77</td>
</tr>
<tr>
<td>GetInterface method</td>
<td>19</td>
</tr>
<tr>
<td>GetJSObject method</td>
<td>78</td>
</tr>
<tr>
<td>GetLanguage method</td>
<td>19</td>
</tr>
<tr>
<td>GetNumAnnots method</td>
<td>94</td>
</tr>
<tr>
<td>GetNumAVDocs method</td>
<td>20</td>
</tr>
<tr>
<td>GetNumber method</td>
<td>95</td>
</tr>
<tr>
<td>GetNumPages method</td>
<td>78</td>
</tr>
<tr>
<td>GetNumText method</td>
<td>99</td>
</tr>
<tr>
<td>GetPage method</td>
<td>50, 100</td>
</tr>
<tr>
<td>GetPageMode method</td>
<td>79</td>
</tr>
<tr>
<td>GetPageNum method</td>
<td>50</td>
</tr>
<tr>
<td>GetPDDoc method</td>
<td>34</td>
</tr>
<tr>
<td>GetPermanentID method</td>
<td>79</td>
</tr>
<tr>
<td>GetPreference method</td>
<td>20</td>
</tr>
<tr>
<td>GetPreferenceEx method</td>
<td>21</td>
</tr>
<tr>
<td>GetRect method</td>
<td>58</td>
</tr>
<tr>
<td>GetRotate method</td>
<td>95</td>
</tr>
<tr>
<td>GetSize method</td>
<td>96</td>
</tr>
<tr>
<td>GetSubtype method</td>
<td>59</td>
</tr>
<tr>
<td>GetText method</td>
<td>101</td>
</tr>
<tr>
<td>GetTitle method</td>
<td>35, 59, 67</td>
</tr>
<tr>
<td>GetVersions method</td>
<td>108</td>
</tr>
<tr>
<td>GetViewMode method</td>
<td>35</td>
</tr>
<tr>
<td>GetZoom method</td>
<td>51</td>
</tr>
<tr>
<td>GetZoomType method</td>
<td>51</td>
</tr>
<tr>
<td>go backward event</td>
<td>168</td>
</tr>
<tr>
<td>go forward event</td>
<td>169</td>
</tr>
<tr>
<td>GoBackwardStack method</td>
<td>108</td>
</tr>
<tr>
<td>GoForwardStack method</td>
<td>108</td>
</tr>
<tr>
<td>goto event</td>
<td>170</td>
</tr>
<tr>
<td>Goto method</td>
<td>52</td>
</tr>
<tr>
<td>goto next event</td>
<td>170</td>
</tr>
<tr>
<td>goto previous event</td>
<td>171</td>
</tr>
<tr>
<td>GotoFirstPage method</td>
<td>108</td>
</tr>
<tr>
<td>GotoLastPage method</td>
<td>109</td>
</tr>
<tr>
<td>GotoNextPage method</td>
<td>109</td>
</tr>
<tr>
<td>GotoPreviousPage method</td>
<td>109</td>
</tr>
<tr>
<td>H</td>
<td></td>
</tr>
<tr>
<td>HFT</td>
<td>209</td>
</tr>
<tr>
<td>Hide method</td>
<td>21</td>
</tr>
<tr>
<td>HideToolbar message</td>
<td>138</td>
</tr>
<tr>
<td>Highlight property</td>
<td>197</td>
</tr>
<tr>
<td>HiliteList object</td>
<td>56</td>
</tr>
<tr>
<td>host function table</td>
<td>209</td>
</tr>
<tr>
<td>Hour property</td>
<td>105</td>
</tr>
<tr>
<td>I</td>
<td></td>
</tr>
<tr>
<td>ImportAnFDF method</td>
<td>207</td>
</tr>
<tr>
<td>index, Catalog plug-in</td>
<td>181</td>
</tr>
<tr>
<td>insert pages event</td>
<td>172</td>
</tr>
</tbody>
</table>
InsertPages method 79
is toolbutton enabled event 172
isEqual method 60
isHidden property 197
isMultiline property 198
isOpen method 60
isPassword property 198
isReadOnly property 198
isRequired property 199
isTerminal property 199
isValid method 35, 61, 68
Item property 208

L
left property 103
Link Annotation object 149
LoadFile method 110
Lock method 21

M
make event 159
maximize event 173
Maximize method 23, 36
menu item object 150
menu object 149
MenuItemExecute message 139
MenuItemExecute method 23
MenuItemIsEnabled method 24
MenuItemIsMarked method 24
MenuItemRemove method 25
Millisecond property 105
Minimize method 22
Minute property 105
Month property 105
move event 159
MovePage method 80

N
Name property 199
_NewEnum property 208
NoViewFlag property 199

O
OLE automation 14
open event 155, 160
Open method 36, 81
OpenAVDoc method 81
OpenInWindow method 37
OpenInWindowEx method 38

P
page object 151
PDAnnot object 56, 151
PDBookMark object 152
PDBookmark object 66
PDDoc object 69
PDF Window object 152
PDLinkAnnot object 152
PDFPage object 86, 152
PDTextAnnot object 152
PDTextSelect object 98
perform event 174
Perform method 62, 68
plug-ins
    Catalog 181
    Forms 183
    Search 209
Point object 102
PointToDevice method 53
PopulateListOrComboBox method 185
PostScript Conversion object 153
print event 155
Print method 110
print pages event 174
PrintAll method 110
PrintAllFit method 111
PrintFlag property 200
PrintPages method 40, 111
PrintPagesEx method 40
PrintPagesFit method 112
PrintPagesSilent method 41
PrintPagesSilentEx method 42
PrintWithDialog method 113

Q
queries 209
quit event 155, 160

R
read page down event 175
read page up event 175
ReadPageDown method 53
ReadPageUp method 54
Rect object 102
Remove method 207
remove toolbutton event 176
RemoveAnnot method 96
replace pages event 176
ReplacePages method 82
Required suite events 155
Restore method 25
Right property 103
run event 156

S
save event 161
Save method 83
scroll event 177
ScrollTo method 54
search lists 219
Search plug-in 209
SearchAddIndex event 213
SearchCountIndexList event 213
SearchDoQuery event 214
SearchGetIndexByPath event 215
SearchGetIndexFlags event 216
SearchGetIndexList event 216
SearchGetIndexPath event 216
SearchGetIndexTitle event 217
SearchGetNthIndex event 217
SearchRemoveIndex event 218
SearchSetIndexFlags event 218
Second property 106
select text event 178
set event 161
set info event 179
SetActiveTool method 26
SetBackgroundColor method 185
SetBorderColor method 186
SetButtonCaption method 187
SetButtonIcon method 187
SetColor method 62
SetContents method 63
SetCurrentHighlight method 113
SetCurrentPage method 113
SetDate method 63
SetExportValues method 188
SetFlags method 84
SetForegroundColor method 189
SetFrame method 26, 43
SetInfo method 85
SetJavaScriptAction method 190
SetLayoutMode method 114
SetNamedDest method 114
SetOpen method 64
SetPageMode method 85, 115
SetPreference method 27
SetPreferenceEx method 27
SetRect method 64
SetResetFormAction method 191
SetRotate method 97
SetShowScrollbars method 115
SetShowToolbar method 116
SetSubmitFormAction method 192
SetTextSelection method 44
SetTitle method 45, 65, 69
SetView method 116
SetViewMode method 45
SetViewRect method 117
SetViewScroll method 117
SetZoom method 118
SetZoomScroll method 118
ShortMenus message 139
Show method 28
ShowTextSelect method 46
ShowToolbar message 140
Src property 119
Style property 200
Text Annotation object 154
text searches 209
TextFont property 201
TextSize property 201
Time object 104
ToolButtonIsEnabled method 28
ToolButtonRemove method 29
Top property 104
Type property 201
Unlock method 29
UnlockEx method 30
user space 221
Value property 202
X property 102
Y property 102
Year property 106
zoom event 179
ZoomTo method 55