



Acrobat Viewer Interapplication Communication Support

Adobe Developer Support

Technical Note #5155

07 December 1994

Adobe Systems Incorporated

Corporate Headquarters
1585 Charleston Road PO Box 7900
Mountain View, CA 94039-7900
(415) 961-4400 Main Number
(415) 961-4111 Developer Support
Fax: (415) 969-4138

Adobe Systems Europe B.V.
Europlaza
Hoogoorddreef 54a
1101 BE Amsterdam Z-O, Netherlands
+31-20-6511 355
Fax: +31-20-6511 313

Adobe Systems Eastern Region
24 New England
Executive Park
Burlington, MA 01803
(617) 273-2120
Fax: (617) 273-2336

Adobe Systems Japan
Swiss Bank House 7F
4-1-8 Toranomom, Minato-ku
Tokyo 105, Japan
+81-3-3437-8950
Fax: +81-3-3437-8968

Copyright © 1994 by Adobe Systems Incorporated. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the publisher. Any software referred to herein is furnished under license and may only be used or copied in accordance with the terms of such license.

PostScript is a trademark of Adobe Systems Incorporated. All instances of the name PostScript in the text are references to the PostScript language as defined by Adobe Systems Incorporated unless otherwise stated. The name PostScript also is used as a product trademark for Adobe Systems' implementation of the PostScript language interpreter.

Any references to a "PostScript printer," a "PostScript file," or a "PostScript driver" refer to printers, files, and driver programs (respectively) which are written in or support the PostScript language. The sentences in this book that use "PostScript language" as an adjective phrase are so constructed to reinforce that the name refers to the standard language definition as set forth by Adobe Systems Incorporated.

Adobe, Acrobat, the Adobe logo, and PostScript are trademarks of Adobe Systems Incorporated or its subsidiaries and may be registered in certain jurisdictions. Apple and Macintosh are registered trademarks and AppleScript is a trademark of Apple Computer, Inc. Microsoft and MS-DOS are registered trademarks and Windows is a trademark of Microsoft Corporation. Other brand or product names are the trademarks or registered trademarks of their respective holders.

This publication and the information herein is furnished AS IS, is subject to change without notice, and should not be construed as a commitment by Adobe Systems Incorporated. Adobe Systems Incorporated assumes no responsibility or liability for any errors or inaccuracies, makes no warranty of any kind (express, implied or statutory) with respect to this publication, and expressly disclaims any and all warranties of merchantability, fitness for particular purposes and noninfringement of third party rights.



Contents

Chapter 1: Introduction 7

- 1.1 IAC overview 7
- 1.2 Organization of this document 7
- 1.3 Other useful documents 8

Chapter 2: Apple Event Support 9

- 2.1 Conventions used in this chapter 9
- 2.2 Differences among the Acrobat viewers 10
- 2.3 Acrobat viewer Apple event support 10
 - Required suite 11
 - Core Suite 12
 - Acrobat viewer suite 16
- 2.4 Acrobat viewer Apple event objects 29
 - Application 29
 - Document 32
 - AVPageView 33
 - PDPage 34
 - PDAnnot 35
 - PDTextAnnot 37
 - PDLinkAnnot 37
 - PDBookmark 37
 - Menu 38
 - Menu item 39

Chapter 3: OLE Support 41

- 3.1 OLE Automation support 41
- 3.2 Differences among the Acrobat viewers 42

3.3	Methods	42
	AcroExch.App	42
	AcroExch.AVDoc	48
	AcroExch.PDDoc	52
	AcroExch.AVPageView	59
	AcroExch.PDPage	62
	AcroExch.PDAnnot	66
	AcroExch.PDTextSelect	69
	AcroExch.PDBookmark	70
3.4	Data types	72
	AcroPoint	72
	AcroRect	72
	AcroExch.HiliteList	72
	AcroExch.Time	72
Chapter 4: DDE Support		73
4.1	General information	73
4.2	Differences among the Acrobat viewers	73
4.3	Acrobat viewer DDE messages	74
	Application configuration	74
	Document manipulation	76
	Document printing	79
	View manipulation	80
	Search-related	82
Appendix A: Constants		83
A.1	Acrobat viewer	83
	Menu names	83
	Menu item names	84
	Tool names	87
	Toolbar button names	88
	Zoom strategies	89
	Page rotation	89
	View mode	89
	Preference item names	90
Appendix B: Apple Event Constants		95
B.1	Event class	95
B.2	Objects and properties	95
	cApplication	96
	cDocument	96
	cMenu	96
	cMenuItem	96
	AVPageView	96
	PDPage	96
	PDBookmark	96
	PDAnnot	96

B.3	Object properties	97
	cApplication	97
	cDocument	97
	cMenu and/or cMenuItem	97
	cAVPageView	98
	cPDPage	98
	cPDBookmark	98
	cPDAnnot	98
B.4	Data Structures	98
	Appendix C: Acrobat Search Plug-in	101
C.1	Menu names	101
C.2	Menu item names	102
C.3	Toolbar button names	102
C.4	Apple events	102
C.5	DDE messages	108
	Running a Query Through DDE	108
	Manipulating Indices Through DDE	111

CHAPTER 1

Introduction

This document describes the interapplication communication (IAC) support provided by version 2 of the Adobe™ Acrobat™ viewers (Exchange and Reader). The viewers support a range of IAC through Apple® events and AppleScript™ (on the Apple Macintosh® computer), and through DDE and OLE 2 (under Microsoft® Windows™). The capabilities provided are a subset of those provided through the Acrobat viewer plug-in API (see Technical Note #5154, *Acrobat Viewer Plug-in API*, which is included with the Acrobat Plug-in SDK, for a description of the API). Just as for the plug-in API, IAC support differs between Exchange, Exchange LE, and Reader.

Readers of this document are expected to already understand the underlying technologies: Apple events, AppleScript, DDE, and OLE. See section 1.2 for a list of documents that describe these technologies if you are not familiar with them. In addition, some familiarity with the Acrobat viewer plug-in API, as described in Technical Note #5154, is useful both for understanding the objects used in the IAC interface and because many of the IAC messages are similar to methods present in the plug-in API.

1.1 IAC overview

The Acrobat viewers' IAC support allows another program to control an Acrobat viewer in the same ways a user could. In addition, the Acrobat viewers can be commanded to render a PDF file into any specified window instead of the Acrobat window. The OLE support provided by the Windows versions of the Acrobat viewers includes both OLE server and OLE automation.

1.2 Organization of this document

This document is split into chapters along technology lines: Apple events (Chapter 2), OLE (Chapter 3), and DDE (Chapter 4). Readers need only read the chapter relevant for their own platform and technology. Appendices contain additional Apple event constants, as well as platform-independent constants such as menu item names.

1.3 Other useful documents

Technical Note #5154, *Acrobat Viewer Plug-In API*. Describes the objects and methods provided by the Acrobat viewer's plug-in API.

Portable Document Format Reference Manual, ISBN 0-201-62628-4, Addison-Wesley. For detailed information about the PDF file format.

Technical Note #5156, *Updates to the Portable Document Format Reference Manual*. Describes enhancements made to the PDF format for version 2 of the Acrobat products.

Inside Macintosh: Interapplication Communication, ISBN 0-201-62200-9, Addison-Wesley. For further information on Apple events and some information on scripting.

AppleScript Language Guide, ISBN 0-201-40735-3, Addison-Wesley. For further information on the AppleScript language.

Apple Event Registry: Standard Suites, by Apple Developer Technical Publications, Part number 030-1958-A. For further information on the core and required Apple events.

OLE 2 Programmer's Reference Volumes One and Two, ISBN 1-55615-628-6 and ISBN 1-55615-629-4, Microsoft Press. For further information on OLE 2 and OLE Automation.

The AutoClik OLE Automation Sample program and tutorial in Microsoft Visual C++ 1.5. A example of using OLE automation.

CHAPTER 2

Apple Event Support

Version 2 of the Acrobat viewers on the Macintosh support Apple events and a number of Apple event objects. The support includes some of the objects and events described in the *Apple Event Registry: Standard Suites*, as well as Acrobat-specific objects and events. This chapter describes each of the events and objects provided.

Apple events can be used from programming languages such as C or from AppleScript. Because AppleScript is much more straightforward, it is the recommended way for developers to use Apple events with the Acrobat viewers whenever possible. Some Apple events, such as the one that allows the Acrobat viewer to render into another application's window, cannot be accessed from AppleScript.

This chapter describes the Apple events supported by the Acrobat viewers. Information on the Apple events supported by the Acrobat Search plug-in is contained in an appendix of this document. It is possible for other plug-ins to support additional Apple events, as described in Technical Note #5154, *Acrobat Viewer Plug-in API*, which is part of the Acrobat Plug-ins SDK.

Note When using Apple events, the first page of a document is page one, not page zero.

2.1 Conventions used in this chapter

All AppleScript examples in this chapter use the English dialect of the AppleScript syntax.

The AppleScript notation used in this section follows that used in the *AppleScript Language Guide*:

- Keywords are shown in roman faces, parameters in italics
- Optional items are enclosed in square brackets
- Lists of choices from which exactly one must be chosen are enclosed in parentheses

- A vertical bar separates items in a list from which one must be chosen.

For example, the command defined as:

```
count [ each | every ] elementType
      [ ( in | of ) object ]
```

may be written in any of the following ways (where the element type is menu item and the object is a menu):

```
count each menu item in menu "File"
count every menu item in menu "File"
count menu item of menu "File"
```

Each piece of AppleScript example code in this chapter assumes that it is being executed within an appropriate tell — end tell construct, for example:

```
tell application "Acrobat™ Exchange 2.0"
    ...example code here...
end tell
```

The constants needed to use the Apple events from a C program are included in header file AETypes.h, located in the IAC folder of the SDK.

2.2 Differences among the Acrobat viewers

Acrobat Exchange supports all of the Apple events described in section 2.3.

Acrobat Exchange LE supports all Apple events except those marked as “Not supported by Exchange LE.”

Acrobat Reader supports only the four required Apple events: run, open, print, and quit.

2.3 Acrobat viewer Apple event support

The Apple events supported by the Acrobat viewers are grouped into three categories:

- Required events — Events that the Finder sends to all applications.
- Core events — Events that are not universal to all applications, but are common to a wide variety of applications.
- Acrobat-specific events — Events that are specific to the Acrobat viewer.

This section describes each of the events in these categories.

The description of each Apple event includes information needed to use it from AppleScript. In addition, the descriptions of the Acrobat-specific events contain information needed to use them from the programming language level. AppleScript users can ignore the “Apple event ID” and “Apple event Parameters” information, while programming language users will generally need all the information provided for each Apple event. Programming language users should also see Appendix B for definitions of additional constants. Further information on the Apple events in the Required and Core suites can be found in the *Apple Event Registry: Standard Suites*.

Many of the Apple events take a parameter of type *reference*. There are many ways a reference can be constructed. For further information, see the *AppleScript Language Guide*.

2.3.1 Required suite

The required suite consists of four events the Finder sends to applications.

run

Description	Launches an application and invokes its standard startup procedures.
AppleScript syntax	<code>run</code>
AppleScript Parameters	None
Return Value	None

open

Description	Opens a file.
AppleScript Syntax	<code>open file</code>
AppleScript Parameters	file (reference) The file or files to open.
Return Value	None

print

Description	Prints one or more files.
AppleScript Syntax	<code>print file</code>
AppleScript Parameters	file (reference) The file or files to print.
Return Value	None

quit

Description	Terminates an application. See the quit event in the core suite for a variant that accepts options.
AppleScript Syntax	quit
AppleScript Parameters	None
Return Value	None

2.3.2 Core Suite

The subset of the Core suite that Acrobat viewers support.

open

Description	Opens a file either into a visible or a hidden window
AppleScript Syntax	open <i>file</i> [invisible <i>hiddenFlag</i>]
AppleScript Parameters	<i>file</i> (reference) The file to open. <i>hiddenFlag</i> (boolean) If true, the file is opened into a hidden window. If false, the file is opened into a visible window. Default is false (<i>i.e.</i> , the window is visible)
Return Value	None

close

Description	Closes one or more documents.
AppleScript Syntax	close <i>document</i> [saving <i>saveOption</i>]
AppleScript Parameters	<i>document</i> (reference) The document to close <i>saveOption</i> A constant that specifies whether or not to save a document that has been modified before quitting. Must be one of: yes — Save the document. no — Do not save the document. ask — Ask the user whether or not to save the document. The default value is ask.
Return Value	None

count

Description	Counts the number of elements of a particular class in an object.
AppleScript Syntax	<code>count [each every] <i>elementType</i> [(in of) <i>object</i>]</code>
AppleScript Parameters	<code>elementType</code> (class type) The class name of the elements to be counted. <code>object</code> (reference) The object whose elements are to be counted.
Return Value	An integer specifying the count.
AppleScript Example	<code>count PDAnnot of document "dev_acro.pdf"</code> <code>count menu item of menu "View"</code>

make

Description	<i>(Not available in Exchange LE)</i> Creates a new object.
AppleScript Syntax	<code>make [new] <i>objectType</i> [at <i>location</i>]</code>
AppleScript Parameters	<code>objectType</code> (class type) The class of the object to create. <code>location</code> (reference) The location at which to insert the new object
Return Value	A reference to the newly created object.
AppleScript Example	<code>set myAnnot to make PDTextAnnot at beginning</code> <code>set name of myAnnot to "Werner Heisenberg"</code> <code>set contents of myAnnot to "Might have been here"</code>

delete

Description	<i>(Not available in Exchange LE)</i> Deletes one or more objects.
AppleScript Syntax	<code>delete <i>object</i></code>
AppleScript Parameters	<code>object</code> (reference) The object to delete.
Return Value	None
AppleScript Example	<code>delete first PDBookmark of document "test.pdf"</code>

exists

Description	Tests whether or not a specified object exists.
AppleScript Syntax	<code>object exists</code> <code>exists object</code>
AppleScript Parameters	<code>object</code> (reference) Object whose existence is to be checked.
Return Value	true if the object exists, false otherwise.
AppleScript Example	<code>exists second document</code> <code>second document exists</code>

get

Description	Gets the value of a property of an object
AppleScript Syntax	<code>[get] property [as className]</code>
AppleScript Parameters	<code>property</code> (reference) The object property or data to get. <code>className</code> (class type) The form in which the data is to be returned.
Return Value	The value of the specified property. If the specified object does not exist, no result is returned.
AppleScript Example	<code>get the name of last PDBookmark</code> <code>get the index of last PDBookmark as string</code>

move

Description	<i>(Not available in Exchange LE)</i> Moves a PDPAGE object.
AppleScript Syntax	<code>move object to location</code>
AppleScript Parameters	<code>object</code> (reference) The PDPAGE to move. <code>location</code> (reference) The location to which <i>object</i> is moved.
Return Value	A reference to the object that was moved.
AppleScript Example	<code>move PDPAGE 3 to before PDPAGE 1</code>

quit

Description	Terminates the Acrobat viewer.
AppleScript Syntax	<code>quit[saving <i>saveOption</i>]</code>
AppleScript Parameters	<code>saveOption</code> A constant that specifies whether to save documents that have been modified before quitting. The possible values are: <code>yes</code> – save the document. <code>no</code> – Do not save the document. <code>ask</code> – If the file has been changed, ask the user whether or not to save the documents. The default value is <code>ask</code> .
Return Value	None
AppleScript Example	<code>quit saving yes</code>

set

Description	<i>(Not available in Exchange LE)</i> Assigns one or more values to one or more variables.
AppleScript Syntax	<code>set <i>location</i> to <i>value</i></code>
AppleScript Parameters	<code>location</code> (reference) The location whose value is to be set. <code>value</code> (expression) The value to which <i>location</i> is to be set.
Return Value	None
AppleScript Example	<code>set the name of first PDBookmark to "Chapter 1"</code>

hide

Description	Hides the Acrobat viewer.
AppleScript Syntax	<code>Hide</code>
AppleScript Parameters	None
Return Value	None
AppleScript Example	<code>Hide</code>

save

Description	Saves a document into a file. The document window's title changes to the name of the file into which it was saved. A file cannot be saved unless it has been modified.
AppleScript Syntax	<code>save document to file</code>
AppleScript Parameters	<code>document</code> (reference) The document to be saved. <code>file</code> (reference) The file into which the document is to be saved.
Return Value	None
AppleScript Example	<pre>save document "dev_acro.pdf" to file "Macintosh HD:backup.pdf"</pre>

2.3.3 Acrobat viewer suite

Acrobat-specific events and objects. Apple encourages the use of an application's signature as the name of its class for application-specific Apple events, hence CARO is the name of the class for Acrobat viewer-specific Apple events. AppleScript users do not need to use this information.

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

close all docs

Description	Closes all documents.
AppleScript Syntax	<code>close all docs [saving <i>saveOption</i>]</code>
AppleScript Parameters	<code>saveOption</code> (constant) A constant that specifies whether to save any document that was been modified before closing it. The possible values are: <code>yes</code> – Save the document. <code>no</code> – Do not save the document. <code>ask</code> – If the document has been modified, ask the user whether or not to save it. The default value is <code>ask</code> .
Return Value	None
AppleScript Example	<pre>close all docs</pre>
Apple event ID	<code>kAECloseAllDocs ('cldc')</code>

remove toolbarbutton

Description	Removes the specified button from the toolbar.
AppleScript Syntax	<code>remove toolbarbutton named <i>buttonName</i></code>
AppleScript Parameters	<code>buttonName</code> (string) The name of the button to remove. See Appendix A for a list of button names.
Return Value	None
AppleScript Example	<code>remove toolbarbutton named "ZoomIn"</code>
Apple event ID	<code>kAERemoveToolButton ('rmtb')</code>
Apple event Parameters	<code>keyAEBUTTONNAME ('tbnm')</code>

is toolbarbutton enabled

Description	Tests whether or not the specified button is enabled.
AppleScript Syntax	<code>is toolbarbutton enabled named <i>buttonName</i></code>
AppleScript Parameters	<code>buttonName</code> (string) Button name. See Appendix A for a list of button names.
Return Value	true if the toolbarbutton is enabled, false otherwise.
AppleScript Example	<code>is toolbarbutton enabled named "AcroSrch:Query"</code>
Apple event ID	<code>kAEIsToolButtonEnabled ('tben')</code>
Apple event Parameters	<code>keyAEBUTTONNAME ('tbnm')</code>

bring to front

Description	Brings the specified document's window to the front.
AppleScript Syntax	<code>bring to front <i>document</i></code>
AppleScript Parameters	<code>document</code> (reference) The document to bring to the front.
Return Value	None
AppleScript Example	<code>bring to front document "AppleEvt.pdf"</code>
Apple event ID	<code>kAEBringToFront ('bfrt')</code>

create thumbs

Description	<i>(Not available in Exchange LE)</i> Creates thumbnail images for all pages in the document.
AppleScript Syntax	<code>create thumbs document</code>
AppleScript Parameters	document (reference) The document in which thumbnails are to be created.
Return Value	None
AppleScript Example	<code>create thumbs document "roadmap.pdf"</code>
Apple event ID	kAECreatethumbs ('crtb')

delete thumbs

Description	<i>(Not available in Exchange LE)</i> Deletes all thumbnails from the document.
AppleScript Syntax	<code>delete thumbs document</code>
AppleScript Parameters	document (reference) The document from which thumbnails are to be deleted.
Return Value	None
AppleScript Example	<code>delete thumbs document "AppleEvt.pdf"</code>
Apple event ID	kAEDelethethumbs ('dltb')

delete pages

Description	Deletes the specified pages in the document (the first page in a document is page 1).
AppleScript Syntax	<code>delete pages document first firstPage last lastPage</code>
AppleScript Parameters	document (reference) The document containing the page to delete. firstPage (integer) The first page to delete. lastPage (integer) The last page to delete.
Return Value	None
AppleScript Example	<code>delete pages document "AppleEvt.pdf" first 1 last 3</code>
Apple event ID	kAEDeletePages ('dlpg')
Apple event Parameters	keyAEFirstPage ('frpg') keyAELastPage ('lapg')

find next note

Description	Finds and selects the next text note in a document.
AppleScript Syntax	<code>find next note <i>document</i> [<i>wrap around</i> <i>wrapToFind</i>]</code>
AppleScript Parameters	<code>document</code> (reference) The document in which to find the next text note. <code>wrapToFind</code> (boolean) Whether or not to continue the search at the beginning of a document if a note has not been found when the end of the document is reached. If true, the search wraps around, otherwise it does not. The default value is false.
Return Value	The text annotation found.
AppleScript Example	<code>find next note document "dev_acro.pdf"</code>
Apple event ID	<code>kAEFindNextNote ('fnnt')</code>

insert pages

Description	<i>(Not available in Exchange LE)</i> Inserts one or more pages from one document into another.
AppleScript Syntax	<code>insert pages <i>destDocument</i> after <i>afterPage</i> from <i>sourceDocument</i> starting with <i>firstPage</i> number of pages <i>numPages</i> [insert bookmarks <i>copyBookmarks</i>]</code>
AppleScript Parameters	<code>destDocument</code> (reference) The document to receive the inserted page or pages. <code>afterPage</code> (integer) The page after which the inserted pages will be placed. <code>sourceDocument</code> (reference) The document containing the page or pages to be inserted. <code>firstPage</code> (integer) The first page to insert. <code>numPages</code> (integer) The number of pages to insert <code>copyBookmarks</code> (boolean) Whether or not to copy bookmarks that point to the inserted pages. Default is true.
Return Value	None
AppleScript Example	<code>insert pages document "AppleEvt.pdf" after 2 from document "dev_acro.pdf" starting with 1 number of pages 4</code>

Apple event ID	kAEInsertPages ('inpg')
Apple event Parameters	keyAEInsertAfter ('inaf') keyAESourceDoc ('srdc') kAESourceStartPage ('stpg') keyAENumPages ('nmpg') keyAEInsertBookmarks ('inbm')

replace pages

Description	<i>(Not available in Exchange LE)</i> Replaces one or more pages in a document with pages from another document.
AppleScript Syntax	<code>replace pages destDocument over destPage from sourceDocument starting with firstPage number of pages numPages [merge notes copyNotes]</code>
AppleScript Parameters	<p>destDocument (reference) The document in which pages are to be replaced.</p> <p>destPage (integer) The first page to replace.</p> <p>sourceDocument (reference) The document from which the replacement page or pages are obtained.</p> <p>firstPage (integer) The first page in sourceDocument to use.</p> <p>numPages (integer) The number of pages to replace.</p> <p>copyNotes (boolean) Whether or not to copy notes from sourceDocument. Default is true.</p>
Return Value	None
AppleScript Example	<code>replace pages document "AppleEvt.pdf" over 2 from document "dev_acro.pdf" starting with 1 number of pages 4 merge notes false</code>
Apple event ID	kAEReplacePages ('rppg')
Apple event Parameters	keyAEDestStartPage ('dtpg') keyAESourceDoc ('srdc') keyAESourceStartPage ('stpg') keyAENumPages ('nmpg') keyAEMergeNotes ('mgnt')

maximize

Description	Sets the document's window size to be the maximum or its original size.
AppleScript Syntax	<code>maximize document maxSize winSize</code>
AppleScript Parameters	<code>document</code> (reference) The document whose window is to be resized. <code>winMaximize</code> (boolean) If true, the document's window is made full size. If false, the window is returned to its original size.
Return Value	None
AppleScript Example	<pre>maximize document "AppleEvt.pdf" with max size false</pre>
Apple event ID	kAEMaximize ('maxi')
Apple event Parameters	keyAEMaxSize ('mxsz')

print pages

Description	Prints one or more pages from a document, without displaying a modal Print dialog box.
AppleScript Syntax	<pre>print pages document [first firstPage] [last lastPage] [PS Level psLevel] [binary output printBinary] [shrink to fit printShrinkToFit]</pre>
AppleScript Parameters	<code>document</code> (reference) The document containing the page or pages to be printed. <code>firstPage</code> (integer) The first page to be printed (defaults to first page in the document). <code>lastPage</code> (integer) The last page to print (defaults to last page in the document). <code>psLevel</code> (integer) The PostScript language level (1 or 2) to use when printing to a PostScript printer. The default value is 1. <code>printBinary</code> (boolean) Whether binary output is OK (Used for PostScript printing only). The default value is false. <code>printShrinkToFit</code> (boolean) Whether or not pages should be shrunk to fit paper in printer. The default value is false.
Return Value	None

AppleScript Example	<pre>print pages document "AppleEvt.pdf" first 1 last 3 PS Level 2 binary output true shrink to fit true</pre>
Apple event ID	<code>kAEPrintPages ('prpg')</code>
Apple event Parameters	<pre>keyAEFirstPage ('frpg') keyAELastPage ('lapg') keyAEPSLevel ('pslv') keyAEBinaryOK ('binO') keyAEShrinkToFit ('s2ft')</pre>

find text

Description	Finds text in a document.
AppleScript Syntax	<pre>find text <i>document</i> string <i>searchString</i> [Case sensitive <i>searchCase</i>] [whole words <i>wordsOnly</i>] [wrap around <i>wrapToFind</i>]</pre>
AppleScript Parameters	<p>document (reference) The document to be searched.</p> <p>searchString (string) The string to be found.</p> <p>searchCase (boolean) Whether or not searching is case-sensitive. The default value is <code>false</code>.</p> <p>wordsOnly (boolean) Whether or not to search only for a whole words. The default value is <code>false</code>.</p> <p>wrapToFind (boolean) Whether or not to continue the search at the beginning of a document if the specified text has not been found when the end of the document is reached.. If <code>true</code>, the search wraps around, otherwise it does not. The default value is <code>false</code>.</p>
Return Value	None
AppleScript Example	<pre>find text document "PLUGINS.PDF" string "Develop" whole words true</pre>
Apple event ID	<code>kAEFindText ('ftxt')</code>
Apple event Parameters	<pre>keyAESearchString ('sstr') keyAECaseSensitive ('case') keyAEWholeWordsOnly ('whwd') keyAEWrapAround ('wrar')</pre>

select text

Description	Selects the specified text.
AppleScript Syntax	<pre>select text <i>pageView</i> ([from words <i>wordPairs</i>] [from chars <i>charPairs</i>])</pre>
AppleScript Parameters	<p>pageView (reference) The page view in which to select text.</p> <p>wordPairs (list of integer pairs) Words to select. One or more pairs of word offsets (from beginning of document) and word lengths (number of contiguous words).</p> <p>charPairs (list of integer pairs) Characters to select. One or more pairs of character offsets (from beginning of document) and character lengths (number of contiguous characters).</p>
Return Value	None
AppleScript Example	<pre>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</pre>
Apple event ID	kAESetTextSelection ('stxs')
Apple event Parameters	<pre>keyAEWordList ('fmwd') keyAECharList ('fmch')</pre>

clear selection

Description	Clears the document's current selection, if any.
AppleScript Syntax	<pre>clear selection <i>document</i></pre>
AppleScript Parameters	<p>document (reference) The document whose selection is to be cleared.</p>
Return Value	None
AppleScript Example	<pre>clear selection document "PLUGINS.PDF"</pre>
Apple event ID	kAEClearSelection ('cls1')

get info

Description	Gets the value of the specified key in the document's Info dictionary.
AppleScript Syntax	<code>get info document key keyName</code>
AppleScript Parameters	<code>document</code> (reference) The document from which to get the Info dictionary entry. <code>keyName</code> (string) The case-sensitive Info dictionary key whose value is to be obtained. The predefined keys are: <code>Creator</code> , <code>Producer</code> , <code>CreationDate</code> , <code>Author</code> , <code>Title</code> , <code>Subject</code> , and <code>Keywords</code> . None of these is required in the PDF file.
Return Value	A string containing the specified key's value, or an empty string if the key is not found.
AppleScript Example	<code>get info document "PLUGINS.PDF" key "CreationDate"</code>
Apple event ID	<code>kAEGetInfo ('gnfo')</code>
Apple event Parameters	<code>keyAEInfoKey ('inky')</code>

set info

Description	<i>(Not available in Exchange LE)</i> Sets the value of a specified key in the document's Info dictionary
AppleScript Syntax	<code>set info document key keyName value newValue</code>
AppleScript Parameters	<code>document</code> (reference) The document in which to set the value of an Info dictionary entry. <code>keyName</code> (string) The Info dictionary key whose value is to be set. <code>newValue</code> (string) The value to set. All keys in the Info dictionary have strings as values.
Return Value	None
AppleScript Example	<code>set info document "PlugIns.pdf" key "Author" value : "Wolfgang Pauli"</code>
Apple event ID	<code>kAESetInfo ('snfo')</code>
Apple event Parameters	<code>keyAEInfoKey ('inky')</code> <code>keyAEInfoValue ('invl')</code>

go forward

Description	Goes to the next view in the stored view history. Does nothing if the current view is the last view in the history.
AppleScript Syntax	<code>go forward <i>pageView</i></code>
AppleScript Parameters	<code>pageView</code> (reference) The AVPageView object in which to change the view.
Return Value	None
AppleScript Example	<code>go forward first AVPageView</code>
Apple event ID	kAEGoForward ('gfwd')

go backward

Description	Goes to the previous view in the stored view history. Does nothing if the current view is the first view in the history.
AppleScript Syntax	<code>go backward <i>pageView</i></code>
AppleScript Parameters	<code>pageView</code> (reference) The AVPageView object in which to change the view.
Return Value	None
AppleScript Example	<code>go backward first AVPageView</code>
Apple event ID	kAEGoBack ('gbck')

goto

Description	Displays the page that has the specified page number.
AppleScript Syntax	<code>goto <i>pageView</i> page <i>pageNumber</i></code>
AppleScript Parameters	<code>pageView</code> (reference) The AVPageView object in which to change the page number. <code>pageNumber</code> (integer) The page number of the page to display. The first page in a document is page 1.
Return Value	None
AppleScript Example	<code>goto first AVPageView page 2</code>
Apple event ID	kAEGotoPage ('gtpg')
Apple event Parameters	keyAEPageNumber ('pg #')

goto next

Description	Displays the page after the one currently displayed in the AVPageView. Does nothing if the current is the last page in the document.
AppleScript Syntax	<code>goto next <i>pageView</i></code>
AppleScript Parameters	<code>pageView</code> (reference) The AVPageView object in which to change the page number.
Return Value	None
AppleScript Example	<code>goto next first AVPageView</code>
Apple event ID	<code>kAEGotoNextPage ('nxpg')</code>

goto previous

Description	Displays the page before the one currently displayed in the AVPageView. Does nothing if the current page is the first page in the document.
AppleScript Syntax	<code>goto previous <i>pageView</i></code>
AppleScript Parameters	<code>pageView</code> (reference) The AVPageView object in which to change the page number.
Return Value	None
AppleScript Example	<code>goto previous first AVPageView</code>
Apple event ID	<code>kAEGotoPrevPage ('pvpg')</code>

read page up

Description	Scrolls backward through the document by one screen.
AppleScript Syntax	<code>read page up <i>pageView</i></code>
AppleScript Parameters	<code>pageView</code> (reference) The AVPageView object to be scrolled.
Return Value	None
AppleScript Example	<code>read page up first AVPageView</code>
Apple event ID	<code>kAEReadPageUp ('pgup')</code>

read page down

Description	Scrolls forward through the document by one screen.
AppleScript Syntax	<code>read page down <i>pageView</i></code>
AppleScript Parameters	<code>pageView</code> (reference) The AVPageView object to be scrolled.
Return Value	None
AppleScript Example	<code>read page down first AVPageView</code>
Apple event ID	kAEReadPageDown ('pgdn')

zoom

Description	Changes the zoom level of specified AVPageView.
AppleScript Syntax	<code>zoom <i>pageView</i> to <i>newZoomLevel</i></code>
AppleScript Parameters	<code>pageView</code> (reference) The AVPageView object to be zoomed. <code>newZoomLevel</code> (real) The zoom factor, in percent (<i>i.e.</i> , a zoom factor of 100 displays the document with a magnification of 1.0).
Return Value	None
AppleScript Example	<code>zoom first AVPageView to 150</code>
Apple event ID	kAEZoomTo ('zmto')
Apple event Parameters	<code>keyAEZoomFactor ('zmft')</code>

scroll

Description	Scrolls the view of a page page by the specified amount.
AppleScript Syntax	<code>scroll <i>pageView</i> X Amount <i>deltaX</i> Y Amount <i>deltaY</i></code>
AppleScript Parameters	<code>pageView</code> (reference) The AVPageView object in which to scroll the view. <code>deltaX</code> (small integer) The amount to scroll in the horizontal direction, in pixels. Positive values move the view to the right. <code>deltaY</code> (small integer) The amount to scroll in the vertical direction, in pixels. Positive values move the view down.

Return Value	None
AppleScript Example	<code>scroll first AVPageView X Amount 20 Y Amount 100</code>
Apple event ID	<code>kAEScroll ('scl')</code>
Apple event Parameters	<code>keyAEXDelta ('xdlt')</code> <code>keyAEYDelta ('ydlt')</code>

perform

Description	Executes a bookmark's or link annotation's action.
AppleScript Syntax	<code>perform <i>object</i></code>
AppleScript Parameters	<code>object</code> (reference) The PDBookmark or PDLinkAnnot object whose action is to be performed.
Return Value	None
AppleScript Example	<code>perform last PDBookmark</code>
Apple event ID	<code>kAEPerform ('prfm')</code>

execute

Description	Executes the specified menu item as if the user selected it.
AppleScript Syntax	<code>execute <i>menuItem</i></code>
AppleScript Parameters	<code>menuItem</code> (reference) The menu item to execute. See Appendix A for a list of menu item names.
Return Value	None
AppleScript Example	<code>activate</code> <code>execute menu item "Open"</code>
Apple event ID	<code>kAEExecute ('exec')</code>

DrawPageToWindow

Description	<p>Instructs the Acrobat viewer to render into a specified window instead of its own window. Use this event if you wish to have the Acrobat viewer render into your application's window. This event must be sent to a PDPage object.</p> <p>See the AERView example program in the IAC directory for an example of the use of this Apple event.</p>
AppleScript Syntax	None
Apple event ID	kAEDrawPageToWindow ('dwpg')
Apple event Parameters	<p>keyAEXFormMatrix ('xfmm') — descriptor list containing six typeShortFloat.</p> <p>Transformation matrix from user space to the device space of the window into which drawing is occurring. Elements are in the order [a b c d e f]. See Section 3.9 of the <i>Portable Document Format Reference Manual</i> for more information on transformation matrices.</p> <p>keyAEUpdateRect ('updr') — descriptor list containing four typeShortFloat.</p> <p>Region of the page to update. List elements are in the order [left top right bottom]. If this parameter is absent, the entire page is updated.</p> <p>keyAEWindow — typeLongInteger</p> <p>WindowPtr for the window into which drawing is to occur.</p>

2.4 Acrobat viewer Apple event objects

The objects presented to the Apple event interface are: application, document, AVPageView, PDPage, PDBookmark, PDAnnot, PDTextAnnot, PDLINKAnnot, menu, and menu item. This section describes each object, lists its elements, its properties, and the methods to which it responds.

Note *The abbreviation r/o is used in the following tables for properties that are read-only.*

2.4.1 Application

Represents the Acrobat viewer application itself.

Elements

<i>Element</i>	<i>Can be accessed...</i>
document	by name, by numeric index
menu	by name, by numeric index
menu item	by name

<i>Element</i>	<i>Can be accessed...</i>
AVPageView	by the name of the document displayed, by numeric index

Properties

<i>Property</i>	<i>Class</i>	<i>r/w</i>	<i>Description</i>
best type	type class	[r/o]	The best descriptor type.
default type	type class	[r/o]	The default descriptor type.
class	type class	[r/o]	The class.
name	string	[r/o]	The application's name.
frontmost	boolean	[r/o]	Is this the frontmost application?
version	integer	[r/o]	The version number of the application. The two upper bytes contain the major version and the two lower bytes contain the minor version. For example, Acrobat 2.0 has a major version of 2 and a minor version of 0.
active doc	reference		The active document.
active tool	string		The currently active tool. See Appendix A for a list of tool names.
toolbar visibility	boolean		Whether or not the toolbar is visible.
long menu visibility	boolean		Whether or not long menus are in use.
UI language	string	[r/o]	Identifies which language the Acrobat viewer's UI is using. This is a 3 character language codes. Language codes are: DEU – German ENU – English ESP – Spanish FRA – French ITA – Italian NLD – Dutch SVE – Swedish
open dialog at startup	boolean		Whether the “Open...” dialog is shown at startup when the Acrobat viewer is launched without a PDF file to open.
show splash at startup	boolean		Whether or not the splash screen is shown at startup.

<i>Property</i>	<i>Class</i>	<i>r/w</i>	<i>Description</i>
default zoom factor	small real		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.
default zoom type	constant		The default zoom type, which determines the zoom and fit attributes when a document is opened. Valid values are “no vary”, “fit page”, “fit width”, “fit height”, and “fit visible width”.
skip warnings	boolean		Whether or not to skip warning dialogs when bookmarks, notes, and links are deleted.
PS level	integer		The PostScript level to be used when printing to a PostScript printer. May be either 1 or 2.
shrink to fit	boolean		If true, the page’s contents are shrunk to fit on the page when printing.
case sensitivity	boolean		Whether searches made using the Find command are case sensitive.
whole word searching	boolean		Whether searches made using the Find command look for whole words only.
note color	'RGB '		The color of the border around newly created text annotations. It is a list of three values, each between 0 and 65535. For example, to set the note color to a deep blue from AppleScript, you might use: <code>set the note color to {0, 0, 32768}</code> .
text note label	string		The text that will appear in the “title bar” of all newly created text notes.

Methods

The application object responds to:

Method

run
open
print
quit
count
make
close all docs
remove toolbarbutton

Method

is toolbutton enabled

2.4.2 Document

The document object represents a single open document in the Acrobat viewer.

Elements

<i>Element</i>	<i>Can be accessed...</i>
PDPage	by numeric index
PDBookmark	by name, by numeric index
AVPageView	No document has more than one AVPageView, so you can access it using an index of one, or use the "some" keyword from AppleScript.

Properties

<i>Property</i>	<i>Class</i>	<i>r/w</i>	<i>Description</i>
best type	type class	[r/o]	The best descriptor type
default type	type class	[r/o]	The default descriptor type
class	type class	[r/o]	The class
name	string	[r/o]	The document's name (as shown in the window's titlebar).
modified	boolean	[r/o]	Whether the document has been modified enough to warrant saving.
bounds	bounding rectangle	[r/o]	The rectangle bounding the window, specified in device space (<i>i.e.</i> , in pixels). The coordinates are specified as {left, top, right, bottom}, and the point (0,0) is in the upper left corner of the screen.
file alias	alias	[r/o]	An alias to the file where the document will be saved to if no other name is supplied. This is usually the file from which the document was read.
view mode	constant		The view mode of the document (just pages, pages and thumbs, or pages and bookmarks).

Methods

Method

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

2.4.3 AVPageView

AVPageView represents the view of the document in its window. Documents that aren't visible don't have AVPageViews.

Elements

<i>Element</i>	<i>Can be accessed...</i>
----------------	---------------------------

PDPPage

Properties

<i>Property</i>	<i>Class</i>	<i>r/w</i>	<i>Description</i>
best type	type class	[r/o]	The best descriptor type
default type	type class	[r/o]	The default descriptor type

<i>Property</i>	<i>Class</i>	<i>r/w</i>	<i>Description</i>
class	type class	[r/o]	The class
name	string	[r/o]	The document's name (as shown in the window's titlebar).
zoom factor	real		The current zoom factor, in percent. For example, a value of 100 corresponds to a zoom factor of 1.0.
zoom type	constant		The zooming and content fitting algorithm current employed. Valid values are "no vary", "fit page", "fit width", "fit height", and "fit visible width".
page number	integer		the number of the current displayed page

Methods

Method

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

2.4.4 PDPage

PDPage represents a single page in a PDF file.

Elements

<i>Element</i>	<i>Can be accessed...</i>
PDAnnot	by numeric index

Properties

<i>Property</i>	<i>Class</i>	<i>r/w</i>	<i>Description</i>
best type	type class	[r/o]	The best descriptor type
default type	type class	[r/o]	The default descriptor type
class	type class	[r/o]	The class
bounds	a list of small real		The boundary rectangle for the page { left, top, right, bottom}.
number	integer	[r/o]	The page's number
rotation	integer		The rotation angle of the page (0, 90, 180, or 270).

Methods

Method

count

delete

move

DrawPageToWindow
(cannot be accessed via
AppleScript)

2.4.5 PDAnnot

A PDAnnot is an annotation on a page of a document. The PDTextAnnot and PDLinkAnnot classes (described in following sections) are two specific annotation types.

Properties

<i>Property</i>	<i>Class</i>	<i>r/w</i>	<i>Description</i>
best type	type class	[r/o]	The best descriptor type
default type	type class	[r/o]	The default descriptor type
class	type class	[r/o]	The class
name	string		The annotation's label (text notes only)
index	integer	[r/o]	The annotation's index within the PDPAGE.
subtype	string	[r/o]	The subtype of the annotation
open state	boolean		Whether the annotation is open (text notes only)

<i>Property</i>	<i>Class</i>	<i>r/w</i>	<i>Description</i>
bounds	a list of small real		The boundary rectangle for the annotation, in PDF space (left, top, right, bottom).
contents	string		The textual contents of the note (text notes only)
modification date	date		The date and time the annotation was last modified
color	'RGB '		The color of the border around the annotation
destination page number	integer		The number of the page to which the link's action goes when it is performed (link annotations only)
destination rectangle	a list of small real		Destination rectangle (specified in user space) on the destination page for the link's action (link annotations only). Coordinates are specified in the following order: (left, top, right, bottom).
fit type	constant		Controls how the destination rectangle is fitted to the window (link annotations only). Must be one of: Left Top Zoom Fit Page Fit Width Fit Height Fit Rect Fit BBox Fit BB Width Fit BB Height These are described in Section 6.6 of the <i>Portable Document Format Reference Manual</i> and in Technical Note #5156, Updates to the <i>Portable Document Format Reference Manual</i> .
zoom factor	small real		The zoom factor used when fit type is Left Top Zoom; ignored otherwise. Setting this property automatically sets fit type to Left Top Zoom (link annotations only)

Methods

Method

perform (performs the action, if any, associated with the annotation)

delete

2.4.6 PDTextAnnot

Can only be used as the target of a *make* event. All other access is via the PDAnnot class. See PDAnnot for properties and elements.

2.4.7 PDLinkAnnot

Can only be used as the target of a *make* event. All other access is via the PDAnnot class. See PDAnnot for properties and elements.

2.4.8 PDBookmark

A bookmark

Elements

None

Properties

<i>Property</i>	<i>Class</i>	<i>r/w</i>	<i>Description</i>
best type	type class	[r/o]	The best descriptor type
default type	type class	[r/o]	The default descriptor type
class	type class	[r/o]	The class
name	string		The bookmark's title
index	integer	[r/o]	The bookmark's index within the document.
destination page number	integer		The number of the page to which the bookmark's action goes when it is performed
destination rectangle	a list of small real		Destination rectangle (specified in user space) on the destination page for the bookmark's action. Coordinates are specified in the following order: (left, top, right, bottom).

<i>Property</i>	<i>Class</i>	<i>r/w</i>	<i>Description</i>
fit type	constant		<p>Controls how the destination rectangle is fitted to the window. Must be one of:</p> <p>Left Top Zoom — Sets a specified zoom and a specified location on the page.</p> <p>Fit Page — Sets the zoom factor so that the entire page into the window.</p> <p>Fit Width — Sets the zoom factor so that the width of the page fits into the window.</p> <p>Fit Height — Sets the zoom factor so that the height of the page fits into the window.</p> <p>Fit Rect — Sets the zoom factor so that the specified rectangle fits into the window.</p> <p>Fit BBox — Sets the zoom so that the rectangle enclosing all marks on the page (known as the <i>bounding box</i>) fits into the window.</p> <p>Fit BB Width — Sets the zoom factor so that the width of the bounding box fits into the window.</p> <p>Fit BB Height — Sets the zoom factor so that the height of the bounding box fits into the window.</p>
zoom factor	small real		The zoom factor used when fit type is Left Top Zoom; ignored otherwise. Setting this property automatically sets fit type to Left Top Zoom

Methods

Method

delete

perform

2.4.9 Menu

A menu in the Acrobat viewer.

Elements

Element

Can be accessed...

menu item

by name, by numeric index.

Properties

<i>Property</i>	<i>Class</i>	<i>r/w</i>	<i>Description</i>
best type	type class	[r/o]	The best descriptor type.
default type	type class	[r/o]	The default descriptor type.
class	type class	[r/o]	The class.
name	string	[r/o]	The menu's name (a language-independent name that uniquely identifies the menu). See Appendix A for a list of menu names.
title	string	[r/o]	The menu's title (as shown in the menu itself). This title will be in the application's UI language.

Methods

Method

count

delete

2.4.10 Menu item

Menu item is a single item in a menu.

Elements

None

Properties

<i>Property</i>	<i>Class</i>	<i>r/w</i>	<i>Description</i>
best type	type class	[r/o]	The best descriptor type
default type	type class	[r/o]	The default descriptor type
class	type class	[r/o]	The class
name	string	[r/o]	The menuitem's name (a language-independent name that uniquely identifies the menuitem). See Appendix A for a list of menu item names.
title	string	[r/o]	The menu item's title (as shown in the menu item itself). This title will be in the application's UI language.
enabled	boolean	[r/o]	Whether the menuitem is enabled

<i>Property</i>	<i>Class</i>	<i>r/w</i>	<i>Description</i>
marked	boolean	[r/o]	Whether the menuitem is checked

Methods

Method

delete

execute

CHAPTER 3

OLE Support

This chapter describes the OLE support present in version 2 of Acrobat Exchange and Acrobat Exchange LE under Windows. The Acrobat viewers are OLE servers and also respond to a variety of OLE automation messages. As an OLE server, PDF documents can be embedded into documents created by an application that is an OLE client. The viewer's support for OLE automation allows it to respond to a variety of requests from other applications. The syntax used in this chapter follows that used in Microsoft Visual Basic 3.0.

Developers who would like to improve their understanding of OLE automation should examine the AutoClick OLE Automation Sample program and tutorial in Microsoft Visual C++ 1.5.

Note: The header files needed by C and C++ programmers to use the OLE automation support described in this chapter are located in the IAC directory. Visual Basic users do not need to use these header files.

3.1 OLE Automation support

The Acrobat viewer is represented as several objects:

- AcroExch.App — The application itself.
- AcroExch.AVDoc — A document as seen in the user interface.
- AcroExch.PDDoc — The underlying PDF representation of a document.
- AcroExch.AVPageView — The window pane in which the document is drawn.
- AcroExch.PDPage — A single page in the PDF representation of a document.
- AcroExch.PDAnnot — An annotation on a page in the PDF file.
- AcroExch.PDBookmark — A bookmark in a PDF file.

- **AcroExch.PDTextSelect** — A selection of text in the Acrobat viewer, as if a user had used the mouse to select a region of text.

There are also several data types used in the calls to these functions:

- **AcroExch.Point** — A point, specified by its *x*- and *y*-coordinates.
- **AcroExch.Rect** — A rectangle, specified by the coordinates of its four sides.
- **AcroExch.Hilite** — An entry in a highlight list. A highlight list is used to highlight one or more groups of characters/words on a single page.
- **AcroExch.Time** — A date and time.

The following sections describe the methods that each of these objects provide, and the data types.

3.2 Differences among the Acrobat viewers

Acrobat Exchange supports all of the OLE automation methods listed in this chapter.

Acrobat Exchange LE supports all the OLE automation methods except those stating “Not available in Exchange LE”.

Acrobat Reader does not support OLE automation.

3.3 Methods

3.3.1 AcroExch.App

Lock

```
BOOL Lock(LPCSTR szLockedBy);
```

Description	Locks the Acrobat viewer. This method only needs to be called when using AcroExch.AVDoc.OpenInWindow to draw into another application's window, and it must be called before invoking that method. Locking the viewer disables all other access to it — neither the user nor other programs will be able to use AcroExch until the Unlock() method is called. Unlock() must be called when you are done using OLE automation.	
Parameters	szLockedBy	A string that is used as the name of the application that has locked the Acrobat viewer.
Return Value	true if the Acrobat viewer was locked successfully, false if it was not. Locking will fail if the user already has AcroExch.EXE open.	

Unlock

`BOOL Unlock();`

Description	Unlocks the Acrobat viewer if it was previously locked.
Parameters	None
Return Value	Always returns true.

Exit

`BOOL Exit();`

Description	Exits the Acrobat viewer.
Parameters	None
Return Value	Always returns true.

Hide

`BOOL Hide();`

Description	Hides the Acrobat viewer. When the viewer is hidden, the user has no control over it, and the Acrobat viewer exits when the last automation object is closed.
Parameters	None
Return Value	Always returns true.

Show

`BOOL Show();`

Description	Shows the Acrobat viewer. When the viewer is shown, the user is in control, and the Acrobat viewer does not automatically exit when the last automation object is destroyed.
Parameters	None
Return Value	Always returns true.

CloseAllDocs

`BOOL CloseAllDocs();`

Description	Closes all open documents.
Parameters	None
Return Value	Always returns true.

MenuItemExecute

```
BOOL MenuItemExecute(LPCSTR szMenuItemName);
```

Description	Executes the menu item whose language-independent menu item name is specified.	
Parameters	szMenuItemName	The language-independent name of the menu item to execute. See Appendix A for a list of menu item names.
Return Value	Returns true if the menu item executes successfully, false otherwise.	

GetActiveTool

```
BSTR GetActiveTool();
```

Description	Gets the name of the currently active tool.	
Parameters	None	
Return Value	false if the call fails. Returns the name of the currently active tool otherwise. See Appendix A for a list of tool names.	

SetActiveTool

```
BOOL SetActiveTool(LPCSTR szButtonName, BOOL bPersistent);
```

Description	Sets the active tool to the tool whose name is specified. Also sets whether or not the tool is a one-shot tool, or should remain active.	
Parameters	szButtonName	The name of the toolbutton to set as the active tool. See Appendix A for a list of tool names.
	bPersistent	A suggestion as to whether the tool should stay activated after it has been used, or is a one-shot tool. If true, the Acrobat viewer is requested to leave the tool active after it has been used. If false, the Acrobat viewer reverts to the previously-active tool after this tool is used once.
Return Value	true if the tool was set, false otherwise.	

ToolButtonRemove

```
BOOL ToolButtonRemove(LPCSTR szButtonName);
```

Description	Removes the specified button from the toolbar.	
Parameters	szButtonName	The name of the button to remove. See Appendix A for a list of button names.
Return Value	true if the button was removed, false otherwise.	

ToolButtonIsEnabled

`BOOL ToolButtonIsEnabled(LPCSTR szButtonName);`

Description	Is the specified toolbar button enabled?	
Parameters	<code>szButtonName</code>	The name of the button whose enabled state is obtained. See Appendix A for a list of button names.
Return Value	true if the button is enabled, false if it is not enabled or does not exist.	

MenuItemRemove

`BOOL MenuItemRemove(LPCSTR szMenuItemName);`

Description	Removes the menu item whose language-independent menu item is specified.	
Parameters	<code>szMenuItemName</code>	The language-independent name of the menu item to remove. See Appendix A for a list of menu item names.
Return Value	true if the menu item was removed, false if the menu item does not exist.	

MenuItemIsEnabled

`BOOL MenuItemIsEnabled(LPCSTR szMenuItemName);`

Description	Is the specified menu item enabled?	
Parameters	<code>szMenuItemName</code>	The language-independent name of the menu item whose enabled state is obtained. See Appendix A for a list of menu item names.
Return Value	true if the menu item is enabled, false if it is disabled or does not exist.	

MenuItemIsMarked

`BOOL MenuItemIsMarked(LPCSTR szMenuItemName);`

Description	Is the specified menu item marked?	
Parameters	<code>szMenuItemName</code>	The language-independent name of the menu item whose marked state is obtained. See Appendix A for a list of menu item names.
Return Value	true if the menu item is marked, false if it is not marked or does not exist.	

GetNumAVDocs

`long GetNumAVDocs();`

Description	Gets the number of open AVDocs. The Acrobat viewer currently supports a maximum of 10 open documents.	
Parameters	None	
Return Value	The number of open AVDocs.	

GetAVDoc

LPDISPATCH GetAVDoc(long nIndex);

Description	Gets an AVDoc by its index in the list of open AVDocs. Use GetNumAVDocs() to determine the number of AVDocs.	
Parameters	nIndex	The index of the document to get.
Return Value	The specified document, or NULL if nIndex is greater than the number of open documents.	

GetActiveDoc

LPDISPATCH GetActiveDoc();

Description	Gets the frontmost document.	
Parameters	None	
Return Value	The frontmost document.	

GetLanguage

BSTR GetLanguage();

Description	Gets a code that specifies which language the Acrobat viewer's user interface is using.	
Parameters	None	
Return Value	String containing a three-letter language code. Must be one of the following: DEU – German ENU – English ESP – Spanish FRA – French ITA – Italian NLD – Dutch SVE – Swedish	

SetPreference

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

Description	Sets a value in the preferences file. Zoom values (used in <code>avpDefaultZoomScale</code> and <code>avpMaxPageCacheZoom</code>) must be passed as percentages and are automatically converted to fixed point numbers, e.g., 100 is automatically converted to 1.0. Colors (used in <code>avpNoteColor</code>) are automatically converted from RGB values to the representation used in the preferences file.	
Parameters	<code>nType</code>	The preferences item whose value is set. Section A.1.8 lists the preferences items.
	<code>nValue</code>	The value to set.
Return Value	Always returns true.	

GetPreference

```
long GetPreference(short nType);
```

Description	Gets a value from the preferences file. Zoom values (used in <code>avpDefaultZoomScale</code> and <code>avpMaxPageCacheZoom</code>) are returned as percentages, e.g., 1.00 is returned as 100. Colors (used in <code>avpNoteColor</code>) are automatically converted to RGB values from the representation used in the preferences file.	
Parameters	<code>nType</code>	The name of the preferences item whose value is obtained.
Return Value	The value of the specified preference item.	

Maximize

```
BOOL Maximize(BOOL bMaximize);
```

Description	Maximizes the Acrobat viewer.	
Parameters	<code>bMaximize</code>	If true, the Acrobat viewer is maximized. If false, the Acrobat viewer is returned to its normal state.
Return Value	Always returns true.	

SetFrame

```
BOOL SetFrame(LPDISPATCH iAcroRect);
```

Description	Sets the window's frame to the specified rectangle.	
Parameters	<code>iAcroRect</code>	A rectangle specifying the window frame.
Return Value	true if the frame was set, false if <code>iAcroRect</code> is not of type <code>AcroRect</code> .	

GetFrame

LPDISPATCH GetFrame();

Description	Gets the window's frame.
Parameters	None
Return Value	The window's frame, specified as an AcroRect.

3.3.2 AcroExch.AVDoc

Open

BOOL Open(LPCSTR szFullPath, LPCSTR szTempTitle);

Description	Opens a file.	
Parameters	szFullPath	The full pathname of the file to open.
	szTempTitle	An optional title for the window in which the file is opened. If szTempTitle is NULL or the empty string (""), it is ignored. Otherwise, szTempTitle is used as the window title.
Return Value	true if the file was opened successfully, false if it was not.	

GetPDDoc

LPDISPATCH GetPDDoc();

Description	Gets the PDDoc associated with an AVDoc.	
Parameters	None	
Return Value	The PDDoc.	

GetAVPageView

LPDISPATCH GetAVPageView();

Description	Gets the AVPageView associated with an AVDoc.	
Parameters	None	
Return Value	The AVPageView.	

SetViewMode

BOOL SetViewMode(long nType);

Description	Sets the mode in which the document will be viewed (pages only, pages and thumbnails, or pages and bookmarks).	
Parameters	nType	The view mode to set. Must be one of the values listed in Section A.1.7.
Return Value	false if an error occurred while setting the view mode, true otherwise.	

FindText

```
BOOL FindText(LPCSTR szText, BOOL bCaseSensitive, BOOL bWholeWordsOnly,  
             BOOL bReset);
```

Description	Finds the specified text, scrolls so that it is visible, and highlights it.	
Parameters	szText	The text that is to be found.
	bCaseSensitive	If true, the search is case-sensitive. If false, it is case-insensitive.
	bWholeWordsOnly	If true, the search matches only whole words. If false, it matches partial words.
	bReset	If true, the search begins on the first page of the document. If false, it begins on the current page.
Return Value	true if the text was found, false if it was not.	

Close

```
BOOL Close(BOOL bNoSave);
```

Description	Closes a document.	
Parameters	bNoSave	If true, the document is closed without saving it. If false and the document has been modified, the user is asked whether or not the file should be saved.
Return Value	false if an error occurred while closing the file, true otherwise.	

GetViewMode

```
long GetViewMode();
```

Description	Gets the current document view mode (pages only, pages and thumbnails, or pages and bookmarks).	
Parameters	None	
Return Value	The current document view mode. Will be one of the values listed in Section A.1.7.	

PrintPages

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

Description	Prints a specified range of pages without displaying any modal Print dialogs.	
Parameters	nFirstPage	The first page to print.
	nLastPage	The last page to print.
	nPSLevel	If 1, PostScript Level 1 operators are used. If 2, PostScript Level 2 operators are also used.
	bBinaryOk	If true, binary data may be included in the PostScript program. If false, all data is encoded as 7-bit ASCII.
	bShrinkToFit	If true, the page is shrunk (if necessary) to fit within the imageable area of the printed page. If false, it is not.
Return Value	false if there were any exceptions while printing, true otherwise.	

ClearSelection

```
BOOL ClearSelection();
```

Description	Clears the current selection.
Parameters	None
Return Value	true if the selection was cleared, false otherwise.

BringToFront

```
BOOL BringToFront();
```

Description	Brings the window to the front.
Parameters	None
Return Value	Always returns true.

GetTitle

```
BSTR GetTitle();
```

Description	Gets the window's title.
Parameters	None
Return Value	The window's title.

Maximize

```
BOOL Maximize(BOOL bMaxSize);
```

Description	Maximizes the window if <code>MaxSize</code> is true.
Parameters	<code>bMaxSize</code>
Return Value	Always returns true.

SetTitle

```
BOOL SetTitle(LPCSTR szTitle);
```

Description	Sets the window's title.
Parameters	<code>szTitle</code> The title to set. This method cannot be used on document windows, but only on windows created by plug-ins.
Return Value	Always returns true.

OpenInWindow

```
BOOL OpenInWindow(LPCSTR szFullPath, short hWnd);
```

Description	Opens a PDF file and displays it in a user-specified window. A lock must have been acquired to use this method. See <code>App.Lock/Unlock</code> .
Parameters	<code>szFullPath</code> The full pathname of the file to open. <code>hWnd</code> The window into which the file is displayed.
Return Value	true if the document was opened successfully, false otherwise.

SetTextSelection

```
BOOL SetTextSelection(LPDISPATCH iAcroPDTextSelect);
```

Description	Sets the document's selection to the specified text selection. Before calling this method, use one of the following to create the text selection: <ul style="list-style-type: none">• <code>PDDoc.CreateTextSelect()</code> — Creates from a rectangle• <code>PDPage.CreatePageHilite()</code> — Creates from a list of character offsets and counts• <code>PDPage.CreateWordHilite()</code> — Creates from a list of word offsets and counts After calling this method, use <code>AVDoc.ShowTextSelect()</code> to show the selection.
Parameters	<code>iAcroPDTextSelect</code> The text selection to use.
Return Value	Always returns true.

ShowTextSelect

BOOL ShowTextSelect();

Description	Changes the view so that the current text selection is visible.
Parameters	None
Return Value	Always returns true.

SetFrame

BOOL SetFrame(LPDISPATCH iAcroRect);

Description	Sets the window's size and location.
Parameters	iAcroRect Rectangle specifying the window's frame.
Return Value	Always returns true.

GetFrame

LPDISPATCH GetFrame();

Description	Gets the rectangle specifying the window's size and location.
Parameters	None
Return Value	An AcroRect containing the frame.

IsValid

BOOL IsValid();

Description	Determines whether the AVDoc is still valid. This method only checks whether the document has been closed or deleted; it does not check the internal structure of the document.
Parameters	None
Return Value	true if the document can still be used, false otherwise.

3.3.3 AcroExch.PDDoc

Open

BOOL Open(LPCSTR szFullPath);

Description	Opens a file.
Parameters	szFullPath Full pathname of the file to open.
Return Value	true if the document was opened successfully, false otherwise.

Close

```
BOOL Close();
```

Description	Closes a file.
Parameters	None
Return Value	true if the document was closed successfully, false otherwise.

InsertPages

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

Description	<i>(Not available in Exchange LE)</i> Inserts pages into a file.	
Parameters	nInsertPageAfter	The page after which pages are inserted.
	iPDDocSource	A PDDoc containing the pages to insert.
	nStartPage	The first page in iPDDocSource to insert.
	nNumPages	The number of pages to insert.
	bBookmarks	If true, bookmarks are copied from the source document. If false, they are not.
Return Value	true if the pages were successfully inserted. Returns false if they were not or if the Acrobat viewer does not support editing.	

ReplacePages

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

Description	<i>(Not available in Exchange LE)</i> Replaces pages in a file with those from a specified file. No links or bookmarks are copied from iPDDocSource, but text annotations may optionally be copied.	
Parameters	nStartPage	The first page to be replaced.
	iPDDocSource	A PDDoc containing the new copies of pages that are replaced.
	nStartSourcePage	The first page in iPDDocSource to use as a replacement page.
	nNumPages	The number of pages to replace.
	bMergeTextAnnotations	If true, text annotations from iPDDocSource are copied. If false, they are not.
Return Value	true if the pages were successfully replaced. Returns false if they were not or if the Acrobat viewer does not support editing.	

DeletePages

BOOL DeletePages(long nStartPage, long nEndPage);

Description	<i>(Not available in Exchange LE)</i> Deletes pages from a file.	
Parameters	nStartPage	The first page to delete.
	nEndPage	The last page to delete.
Return Value	true if the pages were successfully deleted. Returns false if they were not or if the Acrobat viewer does not support editing.	

GetNumPages

long GetNumPages();

Description	Gets the number of pages in a file.	
Parameters	None	
Return Value	The number of pages, or -1 if the number of pages cannot be determined.	

Create

BOOL Create();

Description	<i>(Not available in Exchange LE)</i> Creates a new PDDoc.	
Parameters	None	
Return Value	true if the document is created successfully, false if it is not or if the Acrobat viewer does not support editing.	

GetInfo

BSTR GetInfo(LPCSTR szInfoKey);

Description	Gets the value of a specified key in the document's info dictionary. A maximum of 512 bytes are returned.	
Parameters	szInfoKey	The key whose value is obtained.
Return Value	The string if the value was read successfully. Returns an empty string if the key does not exist or its value cannot be read.	

SetInfo

BOOL SetInfo(LPCSTR szInfoKey, LPCSTR szBuffer);

Description	<i>(Not available in Exchange LE)</i> Sets the value of a key in a document's info dictionary.	
Parameters	szInfoKey	The key whose value is set.
	szBuffer	The value to set.
Return Value	true if the value was added successfully, false if it was not or if the Acrobat viewer does not support editing.	

DeleteThumbs

BOOL DeleteThumbs(long nStartPage, long nEndPage);

Description	<i>(Not available in Exchange LE)</i> Deletes thumbnail images from the specified pages in a document.	
Parameters	nStartPage	First page whose thumbnail image is deleted.
	nEndPage	Last page whose thumbnail image is deleted.
Return Value	true if the thumbnails were deleted, false if they were not deleted or if the Acrobat viewer does not support editing.	

MovePage

BOOL MovePage(long nMoveAfterThisPage, long nPageToMove);

Description	<i>(Not available in Exchange LE)</i> Moves a page to another location within the same document.	
Parameters	nMoveAfterThisPage	The page being moved is placed after this page number.
	nPageToMove	Page number of the page to move.
Return Value	false if the Acrobat viewer does not support editing, true otherwise.	

GetFileName

BSTR GetFileName();

Description	Gets the name of the file associated with this PDDoc.
Parameters	None
Return Value	The file name, which can currently contain up to 256 characters.

GetPageMode

long GetPageMode();

Description	Gets a value indicating whether the Acrobat viewer is currently displaying only pages, pages and thumbnails, or pages and bookmarks.
Parameters	None
Return Value	The current page mode. Will be one of the values listed in Section A.1.7.

SetPageMode

`BOOL SetPageMode(long nPageMode);`

Description	Sets the Acrobat viewer to display only pages, pages and thumbnails, or pages and bookmarks.	
Parameters	<code>nPageMode</code>	The page mode to be set. Must be one of the values listed in Section A.1.7.
Return Value	Always returns true.	

CreateThumbs

`BOOL CreateThumbs(long nFirstPage, long nLastPage);`

Description	<i>(Not available in Exchange LE)</i> Creates thumbnail images for the specified page range in a document.	
Parameters	<code>nFirstPage</code>	First page for which thumbnail images are created.
	<code>nLastPage</code>	Last page for which thumbnail images are created.
Return Value	true if thumbnail images were created successfully, false if they were not or if the Acrobat viewer does not support editing.	

CreateTextSelect

`LPDISPATCH CreateTextSelect(long nPage, LPDISPATCH iAcroRect);`

Description	Creates a text selection from the specified rectangle on the specified page. After creating the text selection, use the <code>AVDoc.SetTextSelection()</code> method of to use it as the document's selection, and use <code>AVDoc.ShowTextSelect()</code> to show the selection.	
Parameters	<code>nPage</code>	The page on which the selection is created.
	<code>iAcroRect</code>	Rectangle enclosing the region to select.
Return Value	A <code>PDTextSelect</code> containing the text selection. Returns <code>NULL</code> if the text selection was not created successfully.	

AcquirePage

`LPDISPATCH AcquirePage(long nPage);`

Description	Acquires the specified page.	
Parameters	<code>nPage</code>	The number of the page to acquire.
Return Value	The <code>PDPage</code> object for the acquired page. Returns <code>NULL</code> if the page could not be acquired.	

GetInstanceID

BSTR GetInstanceID();

Description	Gets the instance ID (the second element) from the ID array in the document's trailer.
Parameters	None
Return Value	A string (up to 32 characters) containing the document's instance ID.

GetPermanentID

BSTR GetPermanentID();

Description	Gets the permanent ID (the first element) from the ID array in the document's trailer.
Parameters	None
Return Value	A string (up to 32 characters) containing the document's permanent ID.

GetFlags

long GetFlags();

Description	Gets a document's flags. The flags indicate whether the document has been modified, whether the document is a temporary document and should be deleted when closed, and the version of PDF used in the file.
Parameters	None
Return Value	The document's flags, containing an OR of the following:

<i>Flag</i>	<i>Description</i>
PDDocNeedsSave	Document has been modified and needs to be saved.
PDDocRequiresFullSave	Document cannot be saved incrementally; it must be written using PDSaveFull.
PDDocIsModified	Document has been modified slightly (such as bookmarks or text annotations have been opened or closed), but not in a way that warrants saving.
PDDocDeleteOnClose	Document is based on a temporary file that must be deleted when the document is closed or saved.
PDDocWasRepaired	Document was repaired when it was opened.

PDDocNewMajorVersion	Document's major version is newer than current.
PDDocNewMinorVersion	Document's minor version is newer than current.
PDDocOldVersion	Document's version is older than current.
PDDocSuppressErrors	Don't display errors.

SetFlags

```
BOOL SetFlags(long nFlags);
```

Description	Sets 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. Note that this method can only be used to set, not to clear, the flag bits.	
Parameters	nFlags	Flags to be set. See GetFlags() for a description of the flags. The flags PDDocWasRepaired, PDDocNewMajorVersion, PDDocNewMinorVersion, and PDDocOldVersion are read-only and cannot be set.
Return Value	Always returns true.	

OpenAVDoc

```
LPDISPATCH OpenAVDoc(LPCSTR szTitle);
```

Description	Opens a window and displays the document in it.	
Parameters	szTitle	the title to be used for the window. A default title is used if szTitle is NULL or the empty string ("").
Return Value	The AVDoc that was opened, or NULL if the open fails.	

Save

```
BOOL Save(short nType, LPCSTR szFullPath);
```

Description Saves a document.

Parameters nType Specifies the way in which the file should be saved. Must be one of the following:

Flag	Description
PDSaveIncremental	Write changes only, not the complete file.
PDSaveFull	Write the entire file.
PDSaveCopy	Write a copy of the file.

szFullPath The new pathname to the file, if any.

Return Value true if the document was successfully saved. Returns false if it was not or if the Acrobat viewer does not support editing.

3.3.4 AcroExch.AVPageView

Goto

```
BOOL GoTo(long nPage);
```

Description Goes to the specified page.

Parameters nPage Page number of the destination page.

Return Value true if the Acrobat viewer successfully went to the page, false otherwise.

ZoomTo

```
BOOL ZoomTo(short nType, short nScale);
```

Description Zooms to a specified magnification.

Parameters nType Zoom type. See Section A.1.5 in Appendix A for a list of zoom types.

nScale The desired zoom factor, expressed as a percent, *e.g.*, 100 is a magnification of 1.0

Return Value true if the magnification was set successfully, false otherwise.

ScrollTo

`BOOL ScrollTo(short nX, short nY);`

Description	Scrolls to the specified location on the current page.	
Parameters	nX	x-coordinate of the destination.
	nY	y-coordinate of the destination.
Return Value	true if the Acrobat viewer successfully scrolled to the specified location, false otherwise.	

ReadPageUp

`BOOL ReadPageUp();`

Description	Scrolls backward through the document by “one screenful”
Parameters	None
Return Value	Always returns true.

ReadPageDown

`BOOL ReadPageDown();`

Description	Scrolls forward through the document by “one screenful”
Parameters	None
Return Value	Always returns true.

DoGoBack

`BOOL DoGoBack();`

Description	Goes to the previous view on the view history stack, if any.
Parameters	None
Return Value	Always returns true.

DoGoForward

`BOOL DoGoForward();`

Description	Goes to the next view on the view history stack, if any.
Parameters	None
Return Value	Always returns true.

GetAVDoc

`LPDISPATCH GetAVDoc();`

Description	Gets the AVDoc associated with the current page.
Parameters	None
Return Value	The AVDoc.

GetPage

`LPDISPATCH GetPage();`

Description	Gets the PDPAGE corresponding to the current page.
Parameters	None
Return Value	The PDPAGE.

GetDoc

`LPDISPATCH GetDoc();`

Description	Gets the PDDoc corresponding to the current page.
Parameters	None
Return Value	The PDoc.

GetZoom

`long GetZoom();`

Description	Gets the current zoom factor, specified as a percent, <i>e.g.</i> , 100 is returned if the magnification is 1.0.
Parameters	None
Return Value	The current zoom factor.

GetZoomType

`short GetZoomType();`

Description	Gets the current zoom type.
Parameters	None
Return Value	Zoom type. See Section A.1.5 in Appendix A for a list of zoom types.

GetPageNum

```
long GetPageNum();
```

Description	Gets the page number of the page. The first page in a document is page zero.
Parameters	None
Return Value	The current page's page number.

PointToDevice

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

Description	Converts the coordinates of a point from user space to device space.
Parameters	iAcroPoint Point whose coordinates are converted.
Return Value	An AcroPoint containing the converted coordinates.

DevicePointToPage

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

Description	Converts the coordinates of a point from device space to user space.
Parameters	iAcroPoint Point whose coordinates are converted.
Return Value	An AcroPoint containing the converted coordinates.

3.3.5 AcroExch.PDPage

GetSize

```
LPDISPATCH GetSize();
```

Description	Gets a page's width and height.
Parameters	None
Return Value	An AcroPoint containing the width and height, measured in points.

GetAnnot

```
LPDISPATCH GetAnnot(long nIndex);
```

Description	Gets the nIndex th annotation on the page.
Parameters	nIndex Index (in the page's annotation array) of the annotation to get. The first annotation in the array has an index of zero.
Return Value	An annotation.

AddNewAnnot

```
LPDISPATCH AddNewAnnot(long nIndexAddAfter, LPCSTR szSubType,  
    LPDISPATCH iAcroRect);
```

Description	<i>(Not available in Exchange LE)</i> Creates a new annotation of a specified type and adds it to the page.	
Parameters	nIndexAddAfter	Location in the page's annotation array to add the annotation. The first annotation on a page has an index of zero.
	szSubType	Subtype of the annotation to create. The subtypes of the built-in annotations are Text and Link.
	iAcroRect	Rectangle bounding the annotation's location on the page.
Return Value	Returns a PDAnnot object, or NULL if the annotation could not be added.	

AddAnnot

```
BOOL AddAnnot(long nIndexAddAfter, LPDISPATCH iPDAnnot);
```

Description	<i>(Not available in Exchange LE)</i> Adds a specified annotation at a specified location in the page's annotation array.	
Parameters	nIndexAddAfter	Location in the page's annotation array to add the annotation. The first annotation on a page has an index of zero.
	iPDAnnot	Annotation to add.
Return Value	false if the Acrobat viewer does not support editing, true otherwise.	

RemoveAnnot

```
BOOL RemoveAnnot(long nIndex);
```

Description	<i>(Not available in Exchange LE)</i> Removes the specified annotation from the page's annotation array.	
Parameters	nIndex	Index in the page's annotation array of the annotation to delete. The first annotation on a page has an index of zero.
Return Value	false if the Acrobat viewer does not support editing, true otherwise.	

GetAnnotIndex

```
long GetAnnotIndex(LPDISPATCH iPDAnnot);
```

Description	Gets the index (in the page's annotation array) of the specified annotation.	
Parameters	iPDAnnot	Annotation whose index is obtained.
Return Value	The annotation's index.	

GetNumAnnots

```
long GetNumAnnots();
```

Description	Gets the number of annotations on the page.	
Parameters	None	
Return Value	The number of annotations on the page.	

CreatePageHilite

```
LPDISPATCH CreatePageHilite(LPDISPATCH iAcroHiliteList);
```

Description	Creates a text selection from a list of character offsets and character counts on a single page. The text selection can then be set as the current selection using <code>AVDoc.SetTextSelect()</code> , and the view can be set to show the selection using <code>AVDoc.ShowTextSelect()</code> .	
Parameters	iAcroHiliteList	Highlight list for which a text selection is created. Use <code>HiliteList.Add()</code> to create a highlight list.
Return Value	AcroPDTextSelect containing the text selection, or NULL if the selection could not be created.	

CreateWordHilite

```
LPDISPATCH CreateWordHilite(LPDISPATCH iAcroHiliteList);
```

Description	Creates a text selection from a list of word offsets and word counts on a single page. The text selection can then be set as the current selection using <code>AVDoc.SetTextSelect()</code> , and the view can be set to show the selection using <code>AVDoc.ShowTextSelect()</code> .	
Parameters	iAcroHiliteList	Highlight list for which a text selection is created. Use <code>HiliteList.Add()</code> to create a highlight list.
Return Value	The text selection, or NULL if the selection could not be created.	

GetDoc

```
LPDISPATCH GetDoc();
```

Description	Gets the PDDoc associated with the page.
Parameters	None
Return Value	The page's PDDoc.

GetNumber

```
long GetNumber();
```

Description	Gets the page number of the current page. The first page in a document is page zero.
Parameters	None
Return Value	The page number of the current page.

GetRotate

```
short GetRotate();
```

Description	Gets the rotation value for the current page.
Parameters	None
Return Value	Rotation value. See Section A.1.6 in Appendix A for a list of rotation values.

SetRotate

```
BOOL SetRotate(short nRotate);
```

Description	<i>(Not available in Exchange LE)</i> Sets the rotation for the current page.	
Parameters	nRotate	Rotation value. See Section A.1.6 in Appendix A for a list of rotation values.
Return Value	false if the Acrobat viewer does not support editing, true otherwise.	

Draw

```
Draw(short window, short displayContext, short XOrigin, short YOrigin,  
    short Zoom);
```

Description	Instructs the Acrobat viewer to draw into a specified window.	
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	x-coordinate of the portion of the page to draw.
	YOrigin	y-coordinate of the portion of the page to draw.
	Zoom	Zoom factor at which the page is to be drawn, in percent (<i>i.e.</i> , 100 corresponds to a magnification of 1.0).
Return Value	true if there were no errors in the parameters, false otherwise.	

3.3.6 AcroExch.PDAnnot

IsValid

```
BOOL IsValid();
```

Description	Tests whether or not an annotation is still valid. This method is intended only to test whether or not the annotation has been deleted, not whether it is a completely valid annotation object.	
Parameters	None	
Return Value	true if the annotation is valid, false otherwise.	

GetSubtype

```
BSTR GetSubtype();
```

Description	Gets an annotation's subtype.	
Parameters	None	
Return Value	The annotation's subtype. The built-in subtypes are Text and Link.	

IsEqual

```
BOOL IsEqual(LPDISPATCH PDAnnot);
```

Description	Determines whether or not an annotation is the same as the specified annotation.	
Parameters	PDAnnot	The annotation to be tested.
Return Value	true if the annotations are the same, false otherwise.	

GetColor

```
long GetColor();
```

Description	Gets an annotation's color.
Parameters	None
Return Value	The annotation's color.

SetColor

```
BOOL SetColor(long nRGBColor);
```

Description	Sets an annotation's color.
Parameters	nRGBColor The color to use for the annotation.
Return Value	true if the annotation's color was set, false if the Acrobat viewer does not support editing.

GetTitle

```
BSTR GetTitle();
```

Description	Gets an annotation's title.
Parameters	None
Return Value	The annotation's title (up to 512 characters)

SetTitle

```
BOOL SetTitle(LPCSTR szTitle);
```

Description	<i>(Not available in Exchange LE)</i> Sets an annotation's title.
Parameters	szTitle The title to use.
Return Value	true if the title was set, false if the Acrobat viewer does not support editing.

GetContents

```
BSTR GetContents();
```

Description	Gets an annotation's contents.
Parameters	None
Return Value	The annotation's contents (currently, up to 1024 characters).

SetContents

`BOOL SetContents(LPCSTR szContents);`

Description	<i>(Not available in Exchange LE)</i> Sets an annotation's contents.	
Parameters	<code>szContents</code>	The contents to use for the annotation.
Return Value	false if the Acrobat viewer does not support editing, true otherwise.	

IsOpen

`BOOL IsOpen();`

Description	Tests whether or not an annotation is open.	
Parameters	None	
Return Value	true if open, false if closed.	

SetOpen

`BOOL SetOpen(BOOL bIsOpen);`

Description	Opens or closes an annotation.	
Parameters	<code>bIsOpen</code>	If true, the annotation is open. If false, the annotation is closed.
Return Value	Always returns true.	

GetRect

`LPDISPATCH GetRect();`

Description	Gets an annotation's bounding rectangle.	
Parameters	None	
Return Value	An AcroRect containing the annotation's bounding rectangle.	

SetRect

`BOOL SetRect(LPDISPATCH iAcroRect);`

Description	Sets an annotation's bounding rectangle.	
Parameters	<code>iAcroRect</code>	Bounding rectangle to set.
Return Value	true if a rectangle was supplied, false otherwise.	

GetDate

```
LPDISPATCH GetDate();
```

Description	Gets an annotation's date.
Parameters	None
Return Value	An AcroTime object containing the date.

SetDate

```
BOOL SetDate(LPDISPATCH iAcroTime);
```

Description	Sets an annotation's date.
Parameters	iAcroTime Date and time to use for the annotation.
Return Value	true if the date was set, false if the Acrobat viewer does not support editing.

Perform

```
BOOL Perform(LPDISPATCH iAcroAVDoc);
```

Description	Performs a link annotation's action.
Parameters	iAcroAVDoc AVDoc in which the annotation is located.
Return Value	true if the action was executed successfully, false otherwise.

3.3.7 AcroExch.PDTextSelect

Destroy

```
BOOL Destroy();
```

Description	Destroys a text selection.
Parameters	None
Return Value	Always returns true.

GetNumText

```
long GetNumText();
```

Description	Gets the number of text elements in a text selection. Use this method to determine how many times to call the <code>GetText()</code> method to obtain all of a text selection's text.
Parameters	None
Return Value	The number of elements in the text selection.

GetBoundingRect

`LPDISPATCH GetBoundingRect();`

Description	Gets a text selection's bounding rectangle.
Parameters	None
Return Value	An AcroRect corresponding to the text selection's bounding rectangle

GetPage

`long GetPage();`

Description	Gets the page number on which a text selection is located.
Parameters	None
Return Value	The text selection's page number.

GetText

`BSTR GetText(long nTextIndex);`

Description	Gets the text from the specified element of a text selection. To obtain all text in a text selection, use <code>GetNumText()</code> to determine the number of elements in the text selection, then use this method in a loop to obtain each of the elements.	
Parameters	<code>nTextIndex</code>	The element of the text selection to get.
Return Value	The text, or an empty string if <code>nTextIndex</code> is greater than the number of elements in the text selection.	

3.3.8 AcroExch.PDBookmark

GetByTitle

`BOOL GetByTitle(LPDISPATCH iAcroPDDoc, LPCSTR bookmarkTitle);`

Description	Gets the bookmark that has a specified title.	
Parameters	<code>iAcroPDDoc</code>	The document containing the bookmark.
	<code>bookmarkTitle</code>	The title of the bookmark to get. The capitalization of the title must match that in the bookmark in order for a match to be found.
Return Value	The specified bookmark.	

Destroy

`BOOL Destroy();`

Description	<i>(Not available in Exchange LE)</i> Destroys a bookmark.
Parameters	None
Return Value	false if the Acrobat viewer does not support editing (making it impossible to delete the bookmark), true otherwise.

IsValid

`BOOL IsValid();`

Description	Tests whether or not the bookmark is valid. This test only ensures that the bookmark has not been deleted; it does not thoroughly check the bookmark's data structures.
Parameters	None
Return Value	true if the bookmark is valid, false if not.

GetTitle

`BSTR GetTitle();`

Description	Gets a bookmark's title (up to 256 characters).
Parameters	None
Return Value	The title.

SetTitle

`BOOL SetTitle(LPCSTR szNewTitle);`

Description	<i>(Not available in Exchange LE)</i> Sets a bookmark's title.
Parameters	<code>szNewTitle</code> The title to set.
Return Value	false if the Acrobat viewer does not support editing, true otherwise.

Perform

`BOOL Perform(LPDISPATCH iAcroAVDoc);`

Description	Performs a bookmark's action.
Parameters	<code>iAcroAVDoc</code> Document in which the bookmark is located.
Return Value	true if the action was executed successfully, false otherwise.

3.4 Data types

3.4.1 AcroPoint

```
short m_x;  
short m_y;
```

3.4.2 AcroRect

```
short m_left;  
short m_top;  
short m_right;  
short m_bottom;
```

3.4.3 AcroExch.HiliteList

Add

```
BOOL Add(short nOffset, short nLength);
```

Description	Adds the highlight specified by <code>nOffset</code> and <code>nLength</code> to the current highlight list. Highlight lists are used to highlight one or more contiguous groups of characters or words on a single page.	
	Highlight lists are used both for character- and word-based highlighting, although a single highlight list cannot contain a mixture of character and word highlights. After creating a highlight list, use <code>PDPage.CreatePageHilite()</code> or <code>PDPage.CreateWordHilite()</code> (depending on whether the highlight list contains character or word highlights) to create a text selection from the highlight list.	
Parameters	<code>nOffset</code>	Offset of the first word or character to highlight. The first word/character on a page has an offset of zero.
	<code>nLength</code>	The number of consecutive words or characters to highlight.
Return Value	Always returns true.	

3.4.4 AcroExch.Time

```
short m_year;  
short m_month;  
short m_date;  
short m_hour;  
short m_minute;  
short m_second;  
short m_millisecond;  
short m_day;
```


CHAPTER 4

DDE Support

This chapter describes the DDE support present in version 2 of the Acrobat viewers under Windows. Descriptions of each DDE message are provided. In general, developers are encouraged to use OLE automation instead of DDE, where possible.

4.1 General information

- For all DDE messages listed in this chapter, the service name is `acroview`, and the topic name is `control`.
- Where a pathname is used, it may be `NULL`, in which case the DDE message operates on the front document.
- If more than one command is sent at once, they will be executed sequentially, and the results will appear to the user as a single action. This can be used, for example, to open a document to a certain page and zoom level.
- Page numbers are zero-based (*i.e.*, the first page in a document is page 0).
- Quotes are needed only if a parameter contains white space.
- You must open a document using the `DocOpen()` DDE message in order to be able to use other DDE messages on it. You cannot use DDE messages, for example, to close a document that a user opened manually.
- The document manipulation methods (*e.g.*, deleting pages or scrolling), only work on documents that are already open.

4.2 Differences among the Acrobat viewers

Acrobat Exchange supports all of the DDE messages listed in section 4.3

Acrobat Exchange LE supports all the DDE messages listed in section 4.3 except those marked as “Not available in Exchange LE.”

Acrobat Reader supports only the following DDE messages: FileOpen, FilePrint, and AppExit.

4.3 Acrobat viewer DDE messages

Note *The square bracket characters [and] in the DDE messages are significant, and must be included as part of the message.*

4.3.1 Application configuration

AppExit

[AppExit()]

Description	Exits the Acrobat viewer.
Parameters	None
Return Value	true if the Acrobat viewer exited successfully, false otherwise.

AppHide

[AppHide()]

Description	Iconifies or hides the Acrobat viewer.
Parameters	None
Return Value	true if the Acrobat viewer was hidden successfully, false otherwise.

AppShow

[AppShow()]

Description	Shows the Acrobat viewer.
Parameters	None
Return Value	true if the Acrobat viewer was shown successfully, false otherwise..

HideToolbar

[HideToolbar()]

Description	Hides the toolbar.
Parameters	None
Return Value	true if the toolbar was hidden successfully, false otherwise.

ShowToolbar

[ShowToolbar()]

Description	Shows the toolbar.
Parameters	None
Return Value	true if the toolbar was shown successfully, false otherwise.

FullMenus

[FullMenus()]

Description	Displays full menus, and sets this option in the Acrobat viewer's preferences file.
Parameters	None
Return Value	true if full menus were set successfully, false otherwise.

ShortMenus

[ShortMenus()]

Description	Displays short menus, and sets this option in the Acrobat viewer's preferences file.
Parameters	None
Return Value	true if short menus were set successfully, false otherwise.

CloseAllDocs

[CloseAllDocs()]

Description	Closes all open documents.
Parameters	None
Return Value	true if the documents were closed successfully, false otherwise.

MenuItemExecute

[MenuItemExecute(char* menuItemName)]

Description	Invokes a menu item, given its language-independent name.	
Parameters	menuItemName	The language-independent name of the menu item to execute. See Appendix A for a list of menu item names.
Return Value	true if the menu item was executed successfully. Returns false if the menu item does not exist, is not enabled, or was not executed successfully.	

4.3.2 Document manipulation

FileOpen

[FileOpen(char* fullPath)]

Description	Opens and displays a file. If the file is already open, makes it the current document and brings it to the front. This DDE message does not add the document to the list that can be manipulated using DDE message; use DocOpen() to do that.	
Parameters	fullPath	The full pathname of the file to open.
Return Value	true if the file was opened successfully, false otherwise.	

DocOpen

[DocOpen(char* fullPath)]

Description	Opens a document and adds it to the list of documents known to DDE, allowing it to be manipulated by other DDE messages (<i>cf</i> FileOpen()).	
Parameters	fullPath	The full pathname of the file to open.
Return Value	true if the file was opened successfully, false otherwise.	

DocClose

[DocClose(char* fullPath)]

Description	Closes the file fullPath without saving it and without prompting the user to save the document if it has been modified.	
Parameters	fullPath	The full pathname of the file to close.
Return Value	true if the document was closed successfully. Returns false if the document does not exist or was not closed successfully.	

DocSave

[DocSave(char* fullPath)]

Description	<i>(Not available in Exchange LE)</i> Saves the specified file. The user is not warned if there are any problems saving the file.	
Parameters	fullPath	The full pathname of the file to save.
Return Value	true if the document was saved successfully. Returns false if the document does not exist or was not saved successfully.	

DocSaveAs

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

Description	<i>(Not available in Exchange LE)</i> Saves an open file into a new file. The user is not warned if there are any problems saving the file.	
Parameters	fullPath	The full pathname of the existing file.
	newPath	The full pathname of the new file.
Return Value	true if the document was saved successfully. Returns false if the document does not exist or was not saved successfully.	

DocInsertPages

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

Description	<i>(Not available in Exchange LE)</i> Inserts pages from one file into another.	
Parameters	fullPath	The full pathname of the file into which pages are being inserted. This file must already be open in the Acrobat viewer.
	insertAfterPage	The page number in fullPath after which pages are being inserted. insertAfterPage may be a number or one of the following special strings: PDBeforeFirstPage— Pages are inserted at the beginning of the document. PDLastPage— Pages are inserted at the end of the document.
	sourcePath	The full pathname of the file containing the new pages to insert. <i>All</i> pages from this file will be inserted into fullPath. This file does not have to be already open in the Acrobat viewer.
Return Value	true if the pages were inserted successfully. Returns false if the document does not exist or the pages were not inserted successfully.	

DocReplacePages

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

Description	<i>(Not available in Exchange LE)</i> Replaces pages in <code>fullPath</code> using pages from <code>sourcePath</code> . <code>fromPage</code> is the first page in <code>fullPath</code> that is replaced. Pages in <code>fullPath</code> are replaced with pages from <code>sourcePage</code> to <code>toSourcePage</code> from <code>sourcePage</code> .	
Parameters	<code>fullPath</code>	The full pathname of the file in which pages are being replaced. This file must already be open in the Acrobat viewer.
	<code>startDestPage</code>	The page number of the first page in <code>fullPath</code> that is to be replaced.
	<code>sourcePath</code>	The full pathname of the file from which replacement pages are obtained. This file does not have to be already open in the Acrobat viewer.
	<code>startSourcePage</code>	The page number of the first page in <code>sourcePath</code> to use as a replacement page.
	<code>endSourcePage</code>	The page number of the last page in <code>sourcePath</code> to use as a replacement page.
Return Value	true if the pages were replaced successfully. Returns false if the document does not exist or the pages were not replaced successfully.	

DocDeletePages

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

Description	<i>(Not available in Exchange LE)</i> Deletes pages between <code>fromPage</code> and <code>toPage</code> in document <code>fullPath</code> . Requests to delete all pages in a document are not carried out, since a document must have at least one page.	
Parameters	<code>fullPath</code>	The full pathname of the file from which pages are being deleted.
	<code>fromPage</code>	The page number of the first page to delete.
	<code>toPage</code>	The page number of the last page to delete.
Return Value	true if the pages were deleted successfully. Returns false if the document specified by <code>fullPath</code> does not exist, if the request was to delete all the document's pages, or if the pages were not deleted successfully.	

DocSetViewMode

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

Description	Controls whether bookmarks, thumbnail images, or neither are shown in addition to the document.	
Parameters	fullPath	The full pathname of the file whose view mode is being set.
	viewType	The view mode to use. Must be one of the following: PDUseThumbs—Displays pages and thumbnail images PDUseNone— Displays only pages PDUseBookmarks—Displays pages and bookmarks
Return Value	true if the view mode was set successfully. Returns false if the document specified by fullPath does not exist or an unknown view mode is specified.	

4.3.3 Document printing

FilePrint

[FilePrint(char* fullPath)]

Description	Prints all pages in a document, without displaying any modal Print dialog 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 onto the imageable area of the printed page.	
Parameters	fullPath	The full pathname of the file being printed.
Return Value	false if the document specified by fullPath does not exist, true otherwise.	

DocPrint

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

Description	Prints a specified range of pages from a document, without displaying any modal Print dialog 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 onto the imageable area of the printed page.	
Parameters	fullPath	The full pathname of the file being printed.
	startPage	The page number of the first page to print.
	endPage	The page number of the last page to print.
Return Value	false if the document specified by fullPath does not exist, true otherwise.	

4.3.4 View manipulation

DocGoTo

[DocGoTo(char* fullPath, long pageNum)]

Description	Goes to the page specified by pageNum.	
Parameters	fullPath	The full pathname of the file.
	pageNum	The page number of the destination page.
Return Value	false if the document specified by fullPath does not exist, true otherwise.	

DocZoomTo

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

Description	Sets the zoom for a specified document.	
Parameters	fullPath	The full pathname of the file whose zoom is being set.
	zoomType	The zoom strategy to use. Must be one of the following: AVZoomNoVary — A fixed zoom, such as 100% AVZoomFitPage — Fits the page in the window AVZoomFitWidth — Fits the page's width into the window AVZoomFitVisibleWidth — Fits the page's visible content into the window
	scale	The magnification in percent (<i>i.e.</i> , 100 corresponds to a magnification of 1.0). scale is used only when zoomType is AVZoomNoVary.
Return Value	false if the document specified by fullPath does not exist, or if zoomType has an unknown value. Returns true otherwise.	

DocScrollTo

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

Description	Scrolls the view of the current page to a specified location.	
Parameters	fullPath	The full pathname of the file being scrolled.
	x	The destination's x-coordinate.
	y	The destination's y-coordinate.
Return Value	false if the document specified by fullPath does not exist, true otherwise.	

DocPageUp

[DocPageUp(char* fullPath)]

Description	Same as the “Page Up” keyboard accelerator; scrolls backward through the document by “one screenful.”	
Parameters	fullPath	The full pathname of the file being scrolled.
Return Value	false if the document specified by fullPath does not exist, true otherwise.	

DocPageDown

[DocPageDown(char* fullPath)]

Description	Same as the “Page Down” keyboard accelerator; scrolls forward through the document by “one screenful.”	
Parameters	fullPath	The full pathname of the file being scrolled.
Return Value	false if the document specified by fullPath does not exist, true otherwise.	

DocPageRight

[DocPageRight(char* fullPath)]

Description	Same as Shift right arrow keyboard accelerator; scrolls to the right by a small amount.	
Parameters	fullPath	The full pathname of the file being scrolled.
Return Value	false if the document specified by fullPath does not exist, true otherwise.	

DocPageLeft

[DocPageLeft(char* fullPath)]

Description	Same as Shift left arrow keyboard accelerator; scrolls to the left by a small amount	
Parameters	fullPath	The full pathname of the file being scrolled.
Return Value	false if the document specified by fullPath does not exist, true otherwise.	

4.3.5 Search-related

DocFind

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

Description	Finds a string in a specified file. This does not use the cross-document search present in version 2 of Acrobat Exchange, but performs a page-by-page search of the specified file.	
Parameters	<code>fullPath</code>	The full pathname of the file being searched.
	<code>string</code>	The string to find.
	<code>caseSensitive</code>	If <code>true</code> , the search is case-sensitive. If <code>false</code> , it is not.
	<code>wholeWords</code>	If <code>true</code> , the search will only match whole words. If <code>false</code> , the search will match partial words.
	<code>bReset</code>	If <code>true</code> , the search begins on the first page of the document. If <code>false</code> , the search begins on the current page.
Return Value	<code>false</code> if the document specified by <code>fullPath</code> does not exist or if the text is not found. Returns <code>true</code> otherwise.	

APPENDIX A

Acrobat Viewer Constants

This appendix contains the names of menus, menu items, tools, and toolbar buttons present in the Acrobat viewers and in the Acrobat Search plug-in. It also contains other constants (such as zoom types). These constants are used in various Apple event, OLE, and DDE calls.

A.1 Acrobat viewer

A.1.1 Menu names

<i>Menu name</i>	<i>Description</i>
Window	Window menu.
Edit	Edit menu.
Tools	Tools menu.
Apple	Apple menu (<i>Macintosh only</i>)
View	View menu.
Extensions	Plug-ins menu.
Pages	Pages submenu of Edit menu.
Bookmarks	Bookmarks submenu of Edit menu.
Thumbnails	Thumbnails submenu of Edit menu.
Notes	Notes submenu of Edit menu.
DocInfo	Document Info submenu of File menu.
AboutExtensions	About Acrobat Plug-ins submenu.
File	File menu.
Prefs	Preferences submenu of Edit menu.
Help	Help menu (Windows only)

A.1.2 Menu item names

Table 1 *Help menu item names (these appear in the Apple menu on the Macintosh)*

About
AboutExtensions
UsingViewer (Windows only)
UsingExtensions (Windows only)

Table 2 *File menu item names*

Open
Close
endFileAccessGroup
Save
SaveAs
endSaveGroup
DocInfo
GeneralInfo
FontsInfo
OpenInfo
SecurityInfo
endDocInfoGroup
PageSetup
Print
endPrintGroup
Recent File (<i>Windows only.</i> <i>Note the space in the name</i>)
endRecentFileGroup (<i>Windows only</i>)
Quit

Table 3 *Edit menu item names*

Undo
endUndoGroup
Cut
Copy
Paste
Clear
SelectAll
endEditGroup
copyFileToClipboard (<i>Windows only</i>)
endOleGroup (<i>Windows only</i>)
Pages
CropPages
RotatePages
endPageOpGroup
InsertPages
ExtractPages
ReplacePages
DeletePages
Bookmarks
NewBookmark
SetBookmarkDest
Thumbs
CreateAllThumbs
DeleteAllThumbs
Notes
ImportNotes
ExportNotes
endObjectsEditGroup
Properties
endPropertiesGroup

Table 3 *Edit menu item names*

ShortLongMenus
Prefs
GeneralPrefs
NotesPrefs
FullScreenPrefs

Table 4 *View menu item names*

ActualSize
FitPage
FitWidth
FitVisible
FullScreen
endZoomTypeGroup
ZoomTo
endZoomGroup
FirstPage
PrevPage
NextPage
LastPage
GoToPage
endPageNavGroup
ArticleThreads
endArticlesGroup
GoBack
GoForward
endGoBackGroup
PageOnly
ShowBookmarks
ShowThumbs

Table 5 *Tool menu item names*

Hand
ZoomIn
ZoomOut
SelectText
SelectGraphics
Note
Link
Thread
endToolsGroup
Find
FindAgain
endFindGroup
FindNextNote
CreateNotesFile

Table 6 *Window menu item names*

ShowHideToolBar
ShowHideClipboard
endShowHideGroup
Cascade
TileHorizontal
TileVertical
CloseAll

A.1.3 Tool names

<i>Tool name</i>	<i>Description</i>
Hand	Hand tool.
Note	Tool for making notes.
Select	Text selection tool.

Zoom	Tool for changing the zoom factor.
Link	Link creation tool.
Thread	Thread creation tool.

A.1.4 Toolbar button names

<i>Button name</i>	<i>Description</i>
UseNone	Displays document only (no bookmarks or thumbnail images).
UseBookmarks	Displays document and bookmarks.
UseThumbs	Displays document and thumbnail images.
endPageModeGroup	Separator not visible in the toolbar
Hand	Allows user to scroll the page.
ZoomIn	Zooms in.
ZoomOut	Zooms out.
Select	Allows user to select text.
Note	Create/select/edit notes.
Link	Creat/select/edit links.
endToolsGroup	Separator not visible in the toolbar
FirstPage	Goes to the document's first page.
PreviousPage	Goes to the previous page in the document.
NextPage	Goes to the next page in the document.
LastPage	Goes to the document's last page.
endPageNavGroup	Separator not visible in the toolbar
GoBack	Goes to the previous view in the view history.
GoForward	Goes to the next view in the view history.
endPageStackGroup	Separator not visible in the toolbar
Zoom100	Sets the zoom factor to 100% (<i>i.e.</i> , actual size)

FitPage	Sets the zoom factor to fit the entire page into the window.
FitVisible	Sets the zoom factor to fit the portion of the page on which drawing appears into the window.
endZoomGroup	Separator not visible in the toolbar
FindDialog	Brings up the Find dialog (not the cross-document search dialog).
endDialogGroup	Separator not visible in the toolbar

A.1.5 Zoom strategies

<i>Zoom type</i>	<i>Description</i>
AVZoomNoVary	no variable zoom (i.e., zoom is a fixed value such as 100%)
AVZoomFitPage	fit page to window
AVZoomFitWidth	fit page width to window
AVZoomFitHeight	fit page height to window

A.1.6 Page rotation

<i>Rotation</i>
pdRotate0
pdRotate90
pdRotate180
pdRotate270

A.1.7 View mode

<i>View mode</i>	<i>Description</i>
PDDontCare	Leave the view mode as it is.

PDUseNone	Display the document, but neither bookmarks nor thumbnail images.
PDUseThumbs	Display the document and thumbnail images.
PDUseBookmarks	Display the document and bookmarks.
PDFullScreen	Display the document in full screen mode.

A.1.8 Preference item names

Table 7 *Preferences file items*

avpPrefsVersion — Int32 (Read only) The preferences file format version number.
avpOpenDialogAtStartup — boolean If true, the “file open” dialog is displayed when the Acrobat viewer is launched without a document to open. If false, it does not.
avpShowSplashAtStartup — boolean If true, the Acrobat viewer splash screen is shown when the Acrobat viewer is launched. If false, it is not.
avpShowToolBar — boolean If true, the Acrobat viewer’s toolbar is displayed. If false, it is not. The toolbar can also be shown and hidden by the user.
avpRememberDialogs — boolean If true, the Acrobat viewer remembers the location of certain dialogs (such as the Find dialog) and displays them in their previous location. If false, they are displayed in a default location.
avpShortMenus — boolean If true, the Acrobat viewer displays short menus. If false, it displays long menus.

Table 7 *Preferences file items*

avpDefaultOverviewType — Int32 Whether thumbnail images, bookmarks, or neither should be shown along with documents by default. Must be one of the following: PDUseNone — Document only. PDUseThumbs — Document plus thumbs. PDUseBookmarks — Document plus bookmarks. PDFullScreen — Full screen mode.
avpDefaultZoomScale — Fixed Default magnification when a document is opened.
avpDefaultZoomType — AVZoomType Default zoom type for a page view. Must be one of the values listed in Section A.1.5.
avpShowLargeImages — boolean If true, the Acrobat viewer displays large images. If false, gray boxes are shown in place of large images, reducing rendering time for pages with large images.
avpGreekText — boolean If true, text smaller than avpGreekLevel is greeked (displayed as gray boxes). If false, all text is drawn, regardless of its size.
avpGreekLevel — Int32 Size, in points, below which text is greeked if avpGreekText is true.
avpSubstituteFontType — Int32 Determines whether sans serif, serif, or both substitution fonts are available when printing. Using only one substitution font type generally reduces the quality of font substitution, but may allow some files that require font substitution to print on PostScript printers that have very little memory. The allowed values are: 0 — Use both sans serif and serif 1 — Use sans serif only 2 — Use serif only
avpDoCalibratedColor — boolean If true, the Acrobat viewer renders using calibrated color.
avpSkipWarnings — boolean If true, a warning dialog box is not displayed when a user deletes notes, bookmarks, links, pages, or thumbnails. If false, the dialog box is displayed.

Table 7 *Preferences file items*

avpPSLevel — Int32
The PostScript language level to use when printing to a PostScript printer. Allowed values are 1 and 2.
avpShrinkToFit — boolean
If true, pages are shrunk to fit the imageable area of the printer when printed. If false, pages are not shrunk to fit.
avpCaseSensitive — boolean
If true, the Acrobat viewer's Find command (not the Search plug-in) performs case-sensitive searches. If false, searches are not case-sensitive.
avpWholeWords — boolean
If true, the Acrobat viewer's Find command (not the Search plug-in) matches only whole words. If false, the partial words are also matched.
avpNoteColor — PDColorValue
Default color to use for new notes.
avpNoteLabel — char *
Default label to use for new notes.
avpMaxThreadZoom — Fixed
The maximum zoom factor that is automatically used when the Acrobat viewer enters "Follow Article" mode. A value of 1.0 corresponds to a zoom factor of 100%.
avpEnablePageCache — boolean
If true, the following page is rendered offscreen while the current page is viewed, improving performance when a document is viewed sequentially. If false, no draw-ahead is used.
avpFullScreencolor — PDColorValue
The background color to use when the Acrobat viewer is in Full Screen mode.
avpMaxPageCacheZoom — Fixed
Maximum zoom factor at which pages will be cached. Pages viewed at a higher zoom factor will not be cached. A value of 1.0 corresponds to a zoom factor of 100%.
avpMinPageCacheTicks — Int32
The minimum number of ticks needed to redraw a page before it will be cached. Pages that can be redrawn in less time will not be cached. A tick is 1/60 of a second.

Table 7 *Preferences file items*

<p>avpMaxPageCacheBytes — Int32</p> <p>The maximum number of bytes the page cache is allowed to occupy.</p>
<p>avpFullScreenChangeTimeDelay — Int32</p> <p>Time (in seconds) to show each page when using automatic page changing in full screen mode.</p>
<p>avpFullScreenLoop — boolean</p> <p>If true, the document's pages are displayed in a loop (rather than just once) when using full screen mode. If false, they are not.</p>
<p>avpThumbViewScale — Fixed</p> <p>The scale at which thumbnail images are created. The Acrobat viewer's default is <code>fixedEighth</code>, creating thumbnail images whose linear dimensions are one-eighth those of the actual page.</p>
<p>avpDestFitType — char *</p> <p>Default destination fit type for creating links and bookmarks.</p>
<p>avpDestZoomInherit — boolean</p> <p>If true, the default for creating new links and bookmarks is "Inherit zoom"</p>
<p>avpHighlightMode — Int32 (Used on Macintosh only)</p> <p>Specifies the way in which highlighted text is to be displayed. Can have one of the following values:</p> <p><code>#define HIGHLIGHT_PAINT_XOR 0</code> — Paint highlight color in XOR mode</p> <p><code>#define HIGHLIGHT_INVERT_MAC 1</code> — Invert in special Macintosh highlight mode</p> <p><code>#define HIGHLIGHT_INVERT_XOR 2</code> — Invert in XOR mode</p> <p>This preference exists because the standard Macintosh highlighting generally works quite well, but can occasionally become invisible when text is on a colored background.</p>
<p>avpDefaultSplitterPos — Int32</p> <p>The default width (in pixels) of the portion of the document window in which bookmarks and thumbnail images are displayed. The actual width can be changed by the user.</p>

APPENDIX B

Apple Event Constants

This appendix contains the values of a number of constants needed to use Apple events from a C language program instead of from AppleScript. Only constants for the Acrobat-specific events are presented and only those constants that do not appear in the Apple event chapter are included. See *The Apple Event Registry: Standard Suites* for a list of the constants for the required and core event suites.

The constants in this appendix are located in the file `AETypes.h`, which is located in the IAC folder.

Note: The prototype for `AEObjectInit()` in the Apple-supplied `AEOObjects.h` is missing a void parameter, so `AETypes.h` contains a private, modified copy.

B.1 Event class

Apple encourages the use of an application's signature as the name of its class for application-specific Apple events, hence `CARO` is the name of the class for Acrobat viewer-specific Apple events.

```
#define kAEAcrobatViewerClass 'CARO'
#define kAEAcrobatViewerSuite kAEAcrobatViewerClass
```

B.2 Objects and properties

The standard objects and properties defined in the *Apple Event Registry: Standard Suites*, and which are used by the Acrobat viewer, are:

- `cDocument` → `AVDoc` object, methods from `AVDoc` and `PDDoc`
- `cMenu` → `AVMenu` object and methods
- `cMenuItem` → `AVMenuItem` object and methods

Note that the Acrobat-specific objects retain their PD/AV/AS prefix (*e.g.*, cAVPageView, cPDPage, etc.), but objects that are also present in an Apple standard object use the Apple name (*e.g.*, cApplication, cDocument, etc.) instead.

B.2.1 cApplication

```
/* cApplication/* embodied in our AVApp object */  
typeApplicationProperty = 'Papp',/* property */
```

B.2.2 cDocument

```
/* cDocument /* physically, our AVDoc object, but  
includes most PDDoc methods as well */  
typeDocumentProperty = 'Pdoc',/* AVDoc property */
```

B.2.3 cMenu

```
/* cMenu /* embodied in our AVMenu object */  
typeMenuProperty = 'Pmnu',/* AVMenu property */
```

B.2.4 cMenuItem

```
/* cMenuItem /* embodied in our AVMenuItem object  
*/  
typeMenuItemProperty = 'Pmen'
```

B.2.5 AVPageView

```
cAVPageView = 'cpgv',/* object */  
typeAVPageViewProperty = 'Ppgv',/* property */
```

B.2.6 PDPage

```
cPDPage = 'cpag',/* object */  
typePDPageProperty = 'Ppag',/* property */
```

B.2.7 PDBookmark

```
cPDBookmark = 'cbkm',/* object */  
typePDBookmarkProperty = 'Pbkm',/* property */
```

B.2.8 PDAnnot

```
cPDAnnot = 'cant',/* object */  
typePDAnnotProperty = 'Pant',/* property */
```


The following are only to be used as parameters to the `kAECreatElements` call.

```
cPDTextAnnot = 'ctan', /* PDTextAnnot object */
cPDLinkAnnot = 'clan' /* PDLinkAnnot object */
```

B.3 Object properties

B.3.1 cApplication

```
#define pActiveDoc 'padc'
#define pActiveToolType 'patl'
#define pToolBarIsVisible 'ptbv'
#define pMenuBarIsVisible 'pmbv'
#define pMenuState 'pmst'
#define pLanguage 'plan'
#define pOpenDialogAtStartup 'pods'
#define pShowSplashAtStartup 'psss'
#define pDefaultZoomFactor 'pdzf'
#define pDefaultZoomType 'pdzt' // an enum
#define kAEZoomNoVary 'pznv'
#define kAEZoomFitPage 'pzfp'
#define kAEZoomFitWidth 'pzfw'
#define kAEZoomFitHeight 'pzfh'
#define kAEZoomFitVisibleWidth 'pzvw'
#define pSkipWarnings 'pskw'
#define pPSLevel 'ppsl'
#define pShrinkToFit 'pstf'
#define pCaseSensitive 'pcas'
#define pWholeWords 'pwwd'
#define pNoteColor 'pntc'
#define pNoteLabel 'pntl'
```

B.3.2 cDocument

```
#define pFileAlias 'pfal'
#define pInstID 'pIID'
#define pPermID 'pPID'
#define pViewMode 'pview' // an enum
#define kAENotVisible 'pnvs'
#define kAEPagesOnly 'pgs'
#define kAEPagesThumbs 'pgtb'
#define kAEPagesBookmarks 'pgbm'
```

B.3.3 cMenu and/or cMenuItem

```
#define pTitle 'ptit'
#define pMarked 'pmrk'
```

B.3.4 cAVPageView

```
#define pZoomFactor'pzmf'
#define pZoomType'pzmt'// an enum
//#define kAEZoomNoVary'pznv'
//#define kAEZoomFitPage'pzfp'
//#define kAEZoomFitWidth'pzfw'
//#define kAEZoomFitHeight'pzfh'
//#define kAEZoomFitVisibleWidth'pzvw'
#define pPageNumber'ppg#'
```

B.3.5 cPDPage

```
//#define pPageNumber'ppg#'
```

B.3.6 cPDBookmark

```
#define pIsValid'pvld'
#define pDestPageNumber'pdp#'
#define pFitType'pftt'
#define kAEFitXYZ'pxyz'
#define kAEFitPage'pzfp'// same as kAEZoomFitPage
#define kAEFitWidth'pzfw'// same as kAEZoomFitWidth
#define kAEFitHeight'pzfh'// same as kAEZoom-
    FitHeight
#define kAEFitRect'pfrt'
#define kAEFitBBox'pfbb'
#define kAEFitBBWidth'pfbw'
#define kAEFitBBHeight'pfbh'
//#define pZoomFactor'pzmf'
#define pDestRect'pdrtr'
```

B.3.7 cPDAnnot

```
#define pSubtype'psub'
#define pOpen'popn'
#define pContents'pcon'
#define pDate'pdte'
//#define pIsValid'pvld'
```

B.4 Data Structures

For the special needs of Acrobat.

```
typedef struct {
    float x;
    float y;
} AEFloatPoint;
```

```

typedef struct {
    short x;
    short y;
} AEInt16Point;

typedef struct{
    Fixed a;
    Fixed b;
    Fixed c;
    Fixed d;
    Fixed h;
    Fixed v;
} AEFixedMatrix;

typedef struct {
    Fixed h;
    Fixed v;
}AEFixedPoint;

typedef struct _t_AEFixedRect {
    Fixed left;
    Fixed top;
    Fixed right;
    Fixed bottom;
} AEFixedRect;

```


APPENDIX C

Acrobat Search Plug-in

The Acrobat Search plug-in adds menus, menu items, toolbar buttons, to the Acrobat viewer. In addition, it supports several Apple events on the Macintosh and DDE messages under Windows. These Apple events and DDE messages allow remote clients to submit search queries and manipulate the list of indices on the shelf.

This appendix contains the names of items added to the Acrobat viewer by the Search plug-in. In addition, it describes the Apple events and DDE messages supported.

C.1 Menu names

<i>Menu name</i>	<i>Description</i>
AcroSrch:ToolsSubMenu	Acrobat Search submenu of Tools menu.

C.2 Menu item names

<i>Menu item name</i>	<i>Description</i>
AcroSrch>About	Displays the About box.
AcroSrch:Prefs	Displays the Search Preferences dialog.
AcroSrch:ToolsSubMenu	Search menu item in the Tools menu.
AcroSrch:Query	Displays the Search dialog.
AcroSrch:Indexes	Displays the Index dialog.
AcroSrch:Results	Displays the Results dialog.
AcroSrch:Assist	Displays the Word Assistant dialog.
AcroSrch:Separator	Separator item in Search submenu.
AcroSrch:PrevDoc	Goes to the previous document in the hit list.
AcroSrch:PrevHit	Goes to the previous hit in the hit list.
AcroSrch:NextHit	Goes to the next hit in the hit list.
AcroSrch:NextDoc	Goes to the next document in the hit list.

C.3 Toolbar button names

<i>Button name</i>	<i>Description</i>
AcroSrch:Separator	Separator not visible in toolbar
AcroSrch:Query	Displays the Query dialog.
AcroSrch:Results	Displays the Results dialog.
AcroSrch:Prev	Goes to the previous hit in the hit list.
AcroSrch:Next	Goes to the next hit in the hit list.

C.4 Apple events

The event class for all Apple events supported by the Search plug-in is:

```
#define kExplorerEvtClass 'Exp!'
```

SearchDoQuery

Description	Executes a specified query, using the set of indices currently on the shelf. The search results are displayed in the Acrobat Search plug-in's Results window.
Apple event ID	kSearchDoQuery ('kwry')
Apple event Parameters	<p>kQueryStringTag ('Qury'), typeChar The query string, a null terminated block of text. Its format is the same as what a user would type into the search Query window, and depends on the search language specified by kParserTag.</p> <p>kParserTag ('Prsr'), typeShortInteger The query parser to use; may be one of (see SrchType.h):</p> <p>kParserSimple 0 — Allows only simple phrase searches; does not allow boolean searching</p> <p>kParserCQL 1 — Allows boolean searches using AND, OR, and NOT, as described in the Acrobat Search plug-in's online help file.</p> <p>kParserBPlus 2 — The Verity BooleanPlus query language. Contact Verity for further information on this language.</p> <p>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, etc.</p> <p>Each string may be any field that appears in the the index, plus Score (which sorts results by relevance ranking). Some common fields are Title, ModificationDate, CreationDate, and Keywords.</p> <p>kWordOptionsTag ('WOpt'), typeLongInteger A bitfield of word options. Must be a logical OR of the values listed in Table 8.</p> <p>The manner in which the options are used depends on the value associated with kOptionsOverrideTag.</p> <p>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.</p> <p>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 must be no greater than 999..</p>
Return Value	None

Table 8 *Search plug-in word options for Apple events (see SearchAE.h)*

kWordOptionCase (1 << 0)
The search is case-sensitive.
kWordOptionStemming (1 << 1)
Find not only the specified word, but other words that have the same stem (<i>e.g.</i> , run and ran have the same stem).
kWordOptionSoundsLike (1 << 2)
Find not only the specified word, but other words that sound like it.
kWordOptionThesaurus (1 << 3)
Find not only the specified word, but other words that have the same meaning.
kWordOptionProximity (1 << 4)
Consider the proximity of results when using the AND operator to look for more than one word in a document. Without kWordOptionProximity, ANDed 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, ANDed terms must be within two or three pages of each other to be found. Also, with kWordOptionProximity, the closer ANDed terms appear together, the higher the relevance ranking of the document that contains them.
kWordOptionRefine (1 << 5)
Do not search the entire list of indices, but only the documents that matched the previous search. This is used to refine the results of the previous search.

SearchGetIndexList

Description	Gets a list of the indices currently on the shelf.
Apple event ID	kSearchGetIndexList (‘gidx’)
Apple event Parameters	None
Return Value	kIndexListTag (‘SilP’), typeLongInteger An opaque void * representing the list of indices currently on the shelf. This value can subsequently be used by other search Apple events to obtain information about a specific index, the number of indices on the shelf, etc.

SearchCountIndexList

Description	Gets the number of indices currently on the shelf.
Apple event ID	kSearchCountIndexList (‘cidx’)
Apple event Parameters	kIndexListTag (‘SilP’), typeLongInteger Opaque void * representing the shelf, obtained from SearchGetIndexList.
Return Value	kIndexListTag (‘SilP’), typeLongInteger Number of indices on the shelf (kIndexListTag here isn’t semantically correct, but works).

SearchGetNthIndex

Description	Gets the n th index on the shelf. The index can be passed to other Search Apple events to remove it from the shelf, obtain its title, etc.
Apple event ID	kSearchGetNthIndex (‘fndx’)
Apple event Parameters	kIndexListTag (‘SilP’), typeLongInteger Opaque void * representing the shelf, obtained from SearchGetIndexList. kNthIndexTag (‘Enth’), typeLongInteger The index to get. The first index on the shelf is index zero.
Return Value	kIndexTag (‘SixP’), typeLongInteger Opaque void * representing an index. Returns NULL if the n th index is gone.

SearchGetIndexByPath

Description	Gets the index that has the specified path. The index must already be on the shelf. The index can be passed to other Search Apple events to remove it from the shelf, obtain its title, etc.
Apple event ID	kSearchGetIndexByPath (‘fpdx’)
Apple event Parameters	kIndexListTag (‘SilP’), typeLongInteger Opaque void * representing the shelf, obtained from SearchGetIndexList. kPathTag (‘Path’), typeChar Macintosh full path representing an index, of the form MyDisk:TopFolder:BottomFolder:Strange.pdx.
Return Value	kIndexTag (‘SixP’), typeLongInteger Opaque void * representing an index. Returns NULL if the specified index is gone.

SearchAddIndex

Description	Adds a specified index to the shelf.
Apple event ID	kSearchAddIndex ('addx')
Apple event Parameters	<p>kIndexListTag ('SilP'), typeLongInteger Opaque void * representing the shelf, obtained from SearchGetIndexList.</p> <p>kPathTag ('Path'), typeChar Macintosh full path representing an index, of the form MyDisk:TopFolder:BottomFolder:Strange.pdx.</p> <p>kFlagTag ('Flag'), typeLongInteger Index flags. See SearchGetIndexFlags for a description of them. The kIndexAvailable flag should always be set.</p>
Return Value	<p>kIndexTag ('SixP'), typeLongInteger Opaque void * representing an index. Returns NULL if failure. Returns</p> <p>#define kIndexExists ((SearchIndexPtr)-1) if the index already exists in the index list. Note: if the index already exists, you can retrieve it using SearchIndexByPath.</p>

SearchRemoveIndex

Description	Removes the specified index from the shelf.
Apple event ID	kSearchRemoveIndex ('rmdx')
Apple event Parameters	<p>kIndexListTag ('SilP'), typeLongInteger Opaque void * representing the shelf, obtained from SearchGetIndexList.</p> <p>kIndexTag ('SixP'), typeLongInteger Opaque void * representing the index to be removed. The index may be obtained using SearchGetIndexByPath, SearchGetNthIndex, or SearchAddIndex.</p>
Return Value	None

SearchGetIndexFlags

Description	Get the flags for an index.
Apple event ID	kSearchGetIndexFlags (‘gfdx’)
Apple event Parameters	kIndexTag (‘SixP’), typeLongInteger Opaque void * representing an index
Return Value	kFlagTag (‘Flag’), typeLongInteger A logical OR of the following: kIndexAvailableFlag (1L << 0) — Set if the index is available for searching. kIndexSelectedFlag (1L << 1) — Set if the index appears with a check mark in the Search plug-in’s user interface. kIndexPtrInvalidFlag (1L << 31) — Set if the index in is not valid or is no longer valid.

SearchSetIndexFlags

Description	Sets the flags for an index.
Apple event ID	kSearchSetIndexFlags (‘sfdx’)
Apple event Parameters	kIndexTag (‘SixP’), typeLongInteger Opaque void * representing an index kFlagTag (‘Flag’), typeLongInteger Index flags. See the description in SearchGetIndexFlags. In practice, kIndexAvailableFlag should always be set.
Return Value	kFlagTag (‘Flag’), typeLongInteger Index flags. See the description in SearchGetIndexFlags. This value is returned because it is possible for a request to set a flag to fail.

SearchGetIndexTitle

Description	Gets the title of an index.
Apple event ID	kSearchGetIndexTitle (‘gtdx’)
Apple event Parameters	kIndexTag (‘SixP’), typeLongInteger Opaque void * representing the index whose title is to be obtained. The index may be obtained using SearchGetIndexByPath, SearchGetNthIndex, or SearchAddIndex.
Return Value	kTitleTag (‘Titl’), typeChar A null-terminated character string representing the title of the index. If there is no title, it will return the index’s path. Returns an empty string if the requested index is not valid.

SearchGetIndexPath

Description	Gets the full path to an index.
Apple event ID	kSearchGetIndexPath ('gpdx')
Apple event Parameters	kIndexTag ('SixP'), typeLongInteger Opaque void * representing the index whose path is to be obtained. The index may be obtained using SearchGetIndexByPath, SearchGetNthIndex, or SearchAddIndex.
Return Value	kPathTag ('Path'), typeChar A null-terminated character string representing the full path of the index. Returns an empty string if the requested index is not valid.

C.5 DDE messages

A client can connect to the Search plug-in via DDE using the service name "Acrobat Search" and the topic name "Acrobat Search".

```
DdeInitialize(&id, &DDE_ProcessMessage,  
    APPCMD_CLIENONLY, 0);  
hszServerName = DdeCreateStringHandle(id,  
    "Acrobat Search", 0);  
hszTopicName = DdeCreateStringHandle(id,  
    "Acrobat Search", 0);  
hConv = DdeConnect(id, hszServerName, hszTopicName,  
    NULL);
```

C.5.1 Running a Query Through DDE

After a connection has been made, a single poke transaction will invoke a search. The item name to use is "Query".

```
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:

```
typedef struct _QueryData
{
    eQLangType qlt;
    boolean bOverrideWordOptions;
    uns32 nWordOptions;
    uns16 nMaxDocs;
    uns16 nQueryOffset;
    uns16 nNumSorts;
    uns16 nSortOffset[QP_MAX_SORT_FIELDS];
    boolean bSortWays[QP_MAX_SORT_FIELDS];
    unsigned char cData[1];
} QueryData;
```

Where:

qlt is the query language type and must be one of the values listed in : Table 9.

bOverrideWordOptions indicates that the client wishes to use different word options than those currently set by the user.

nWordOptions is the word options, and must be an OR of the values listed in Table 10.

If nMaxDocs is non-zero, this indicates that the client wishes to use a different maximum docs limit than the limit currently set by the user.

nSortOffsets is a list of offsets into the cData chunk. Each offset points to a null-terminated string containing the field name.

nQueryOffset is an offset into the cData chunk that points to a null-terminated string containing the query to execute.

nNumSort is the number of fields in the sort spec. If this number is 0 then the current sort spec set by the user is used.

bSortWays is a list of sort order flags, one for each sort field. true indicates an ascending sort, and false indicates a descending sort.

Table 9 *Search plug-in query parser constants for DDE messages (see SEARCHDDE.H)*

QLangType_Simple
Allows only simple phrase searches; does not allow boolean searching
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.

Table 10 *Search plug-in word options for DDE messages (see SEARCHDDE.H)*

QPON_Case
The search is case-sensitive.
QPON_Stemming
Find not only the specified word, but other words that have the same stem (<i>e.g.</i> , 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 QPON_Proximity, ANDed 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 QPON_Proximity, however, ANDed terms must be within two or three pages of each other to be found. Also, with QPON_Proximity, the closer ANDed terms appear together, the higher the relevance ranking of the document that contains them.

Table 10 *Search plug-in word options for DDE messages (see SEARCHDDE.H)*

QPON_Refine

Do not search the entire list of indices, but only the documents that matched the previous search. This is used to refine the results of the previous search.

In order 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:

```
nSize = sizeof(QueryData) + sls + lq;
qd = (QueryData *)malloc(sizeof(QueryData) + sls + lq);
```

E.g., if the query was "Adobe" and the sort spec was "Title" ascending and "Score" descending then the structure would be packed as follows.

```
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");
```

C.5.2 Manipulating Indices Through DDE

After a connection has been made, a single poke transaction can add, delete, add , or remove indices. The item name to use is "Index".

```
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:

```
typedef struct _IndexData
{
    IndexActionType eAction;
    int16 nIndexOffset;
    int16 nTempNameOffset;
    unsigned char cData[1];
} IndexData;
```

Where:

eAction specifies the operation to be performed on the index, and must be one of values listed in Table 11.

nIndexOffset is an offset into the cData chunk that points to a null-terminated string containing the pdx file representing the index.

nTempNameOffset is a temporary name to use when adding an index in case the index is temporarily unavailable.

Table 11 *Search plug-in index operation selectors for DDE messages (see SEARCHDDE.H)*

IndexAction_Add
Adds an index to the shelf
IndexAction_Remove
Removes an index from the shelf
IndexAction_Enable
Enables an index on the shelf
IndexAction_Disable
Disables an index on the shelf

In order 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:

```
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:

```
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");
```

