Adobe Acrobat 7.0.5

Acrobat Interapplication Communication Reference

October 7, 2005

Adobe Solutions Network — http://partners.adobe.com

Contents

OLE Automation

DDE

Plug-Ins

Preface

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

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

Description

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

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

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

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

Audience

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

Prerequisites

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

- Apple events
- AppleScript
- DDE
- OLE

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

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

Related Documents

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

Developer Documentation

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

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

Other Interapplication Communication Documentation

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

Core API Documentation

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

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

File Format Documentation

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

Platform-Specific Documentation

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

AppleScript Language Guide, ISBN 0-201-40735-3, Addison-Wesley. This contains more information on the AppleScript language.

Apple Event Registry: Standard Suites, by Apple Developer Technical Publications, Part number 030-1958-A. This contains more 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. Volume One contains information on OLE 2.0; Volume Two covers OLE Automation.

Conventions Used in This Book

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

Font	Used for	Examples
monospaced	Paths and filenames	`C:\templates\mytmpl.fm`
	Code examples set off from plain text	These are variable declarations: `AVMenu commandMenu,helpMenu;`
monospaced bold	Code items within plain text	The **`GetExtensionID`** method ...
	Parameter names and literal values in reference documents	The enumeration terminates if **`proc`** returns **`false`**.
monospaced italic	Pseudocode	`ACCB1 void ACCB2 ExeProc(void)` `{ do something }`
	Placeholders in code examples	`AFSimple_Calculate(cFunction, cFields)`

Font	Used for	Examples
blue	Live links to Web pages	The Adobe Solutions Network URL is: http://partners.adobe.com/asn/
	Live links to sections within this document	See Using the SDK.
	Live links to code items within this document	Test whether an `ASAtom` exists.
bold	PostScript® language and PDF operators, keywords, dictionary key names	The **setpagedevice** operator
	User interface names	The **File** menu
italic	Document titles that are not live links	*Acrobat Core API Overview*
	New terms	*User space* specifies coordinates for...
	PostScript variables	*filename* **deletefile**

Apple Events

This reference contains the following sections:

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

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

The object and event descriptions have the following conventions.

- Object Descriptions

  The abbreviation *r/o* is used for properties that are read-only.

- Event Descriptions

  All AppleScript examples use the English dialect of AppleScript syntax.

  Optional items are enclosed in square brackets [ ].

  Each AppleScript code sample assumes that it is being executed within an appropriate *tell — end tell* construct, as in this example:

  ```
  tell application "Acrobat 7.0"
  …sample code here…
  end tell
  ```

1 Apple Event Objects

This chapter details Apple Event Objects, with descriptions of each object's elements and properties.

annotation

Description

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

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

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

Elements

None.

Plural form

Annotations

Properties

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

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

Related Methods

delete

perform

application

Description

The Acrobat or Adobe Reader application itself.

Elements

Elements	Can be accessed by
`document`	name, numeric index
`PDF Window`	name, numeric index
`menu`	name, numeric index
`menu item`	name

Properties

Property	Class	Description
`active doc`	reference	The active document.
`active tool`	international text	The type of the currently active tool. See the *Acrobat and PDF Library API Reference* for a list of tool names.
`anti_alias text`	boolean	Determines whether to anti-alias text and monochrome images.
`best type`	type class [r/o]	The best descriptor type.
`case sensitivity`	boolean	Determines whether searches are case-sensitive.
`class`	type class [r/o]	The class.
`default type`	type class [r/o]	The default descriptor type.
`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 (100%).
`default zoom type`	constant	The default zoom type when opening a new document. Valid values are "no vary", "fit page", "fit width", "fit height", and "fit visible width."

Property	Class	Description
download entire file	boolean	Determines whether to download the entire file.
frontmost	boolean	Determines whether Acrobat is the frontmost application. Value can be set to true only.
fullscreen click advances	boolean	Determines whether mouse click advances in fullscreen mode.
fullscreen cursor	boolean	Determines whether to hide the cursor in fullscreen mode.
fullscreen escape	boolean	Determines whether the <Esc> key can be used to exit fullscreen mode.
fullscreen loop	boolean [r/o]	Determines whether the document's pages are displayed in a loop while in fullscreen mode.
fullscreen timer delay	integer	The number of seconds to advance to the next page in fullscreen mode.
fullscreen transition	international text [r/o]	Default fullscreen transition.
highlight color	'RGB '	Color used to highlight selections.
maximum documents	integer [r/o]	Maximum number of open documents.
name	string [r/o]	The application's name.
note color	'RGB '	A list of three values between 0 and 65535 representing the color of the border around text annotations. The following example sets the note color to deep blue: **set the note color to {0, 0, 32768}**.
note font name	international text	**NOTE:** Deprecated.
note font size	integer	**NOTE:** Deprecated.
open in place	boolean	Determines whether to open cross-document links in the same window.
page layout	international text	Default page layout. Values are: **Single Page**, **Continuous**, **Facing**, and **Continuous - Facing**.

Property	Class	Description
`page units`	international text	Default page display units: **Points**, **Inches** or **Millimeters**
`PS level`	integer	NOTE: Deprecated. Set PostScript level when using `save` or `print pages` commands.
`save as linearize`	boolean	Determines whether to save the document as a linearized file. Primarily used to optimize document viewing in a web browser.
`show splash at startup`	boolean	Determines whether the splash screen is shown at startup.
`skip warnings`	boolean	Determines whether to skip warning dialog boxes during program execution.
`shrink to fit`	boolean	NOTE: Deprecated.
`text note label`	international text	The text that will appear in the title bar of all newly created text notes.
`toolbar visibility`	boolean	Determines whether the toolbar is visible.
`UI language`	international text [r/o]	A three-character language code identifying which language is used in the Acrobat user interface. Example: `ENU` represents English.
`use fullscreen timer`	boolean	Determines whether to use a timer to advance pages in fullscreen mode
`version`	string [r/o]	The version number of the application.
`whole word searching`	boolean	Determines whether searches are applied to whole words only.

Related Methods

`close all docs`

`count`

`make`

`open`

`print`

`quit`

`run`

AVPageView

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

bookmark

Description

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

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

Elements

None.

Plural form

Bookmarks.

Properties

Property	Class	Description
best type	type class [r/o]	The best descriptor type.
class	type class [r/o]	The class.
default type	type class [r/o]	The default descriptor type.
destination page number	integer	The page number to which the **PDF Window** goes when the bookmark's action is performed.
destination rectangle	list of small real	Boundary rectangle (specified in user space) for the view of the destination when the bookmark's action is performed. Coordinates are specified in the following order: (left, top, right, bottom). **NOTE:** Set this only after setting **fit type**.

Property	Class	Description
`fit type`	constant	Controls how the destination rectangle is fitted to the window when the bookmark's action is performed. Possible values: `Left Top Zoom` — Sets a specified zoom and a specified location on the page. `Fit Page` — Sets the zoom factor so that the entire page fits into the window. `Fit Width` — Sets the zoom factor so that the width of the page fits into the window. `Fit Height` — Sets the zoom factor so that the height of the page fits into the window. `Fit Rect` — Sets the zoom factor so that the specified rectangle fits into the window. `Fit BBox` — Sets the zoom so that the rectangle enclosing all marks on the page (known as the *bounding box*) fits into the window. `Fit BB Width` — Sets the zoom factor so that the width of the bounding box fits into the window. `Fit BB Height` — Sets the zoom factor so that the height of the bounding box fits into the window.
`index`	integer [r/o]	The bookmark's index within the `document`.
`name`	international text	The bookmark's title.
`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`.

Related Methods

`insert pages`

`perform`

conversion

Description

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

```
get every conversion
```

Properties

Property	Class	Description
`best type`	type class [r/o]	The best descriptor type.
`class`	type class [r/o]	The class.
`default type`	type class [r/o]	The default descriptor type.
`index`	integer [r/o]	The index number of the converter.
`name`	international text	The conversion's description.

Related Methods

`save`

document

Description

Represents a single open document in Acrobat or Adobe Reader.

Elements

Element	Can be accessed
`page`	by numeric index. The first page in a document is page 1.
`bookmark`	by name or numeric index.
`PDF Window`	No document has more than one `PDF Window`, so it may be accessed by using an index of 1 or via the `some` keyword in AppleScript.

Plural form

documents.

Properties

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

Related Methods

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

EPS Conversion

Description

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

Properties

Inherits from `Postcript Conversion`.

Related Methods

`save`

Link Annotation

Description

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

NOTE: This object was formerly known as `PDLinkAnnot`.

Elements

None.

Properties

Inherits from `annotation`.

Related Methods

`delete`

`perform`

menu

Description

A menu in Acrobat or Adobe Reader's menu bar.

Elements

Element	Can be accessed by
`menu item`	name, numeric index.

Properties

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

Related Methods

`execute`

menu item

Description

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

Elements

None.

Properties

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

Related Methods

`execute`

page

Description

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

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

Elements

Element	Can be accessed by
annotation	numeric index.

Plural form

Pages.

Properties

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

Related Methods

delete pages

insert pages

replace pages

goto

move

PDAnnot

NOTE: Deprecated. Use `annotation` now.

PDBookMark

NOTE: Deprecated. Use `bookmark` now.

PDLinkAnnot

NOTE: Deprecated. Use `Link Annotation` now.

PDPage

NOTE: Deprecated. Use `page` now.

PDTextAnnot

NOTE: Deprecated. Use `Text Annotation` now.

PDF Window

Description

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

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

Elements

Element	Can be accessed by...
`page`	numeric index. The first page in a document is page 1.

Properties

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

Related Methods

`go backward`

`go forward`

`goto`

`goto next`

`goto previous`

`read page down`

`read page up`

`scroll`

`select text`

`zoom`

Postcript Conversion

Description

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

Properties

Inherits other properties from `conversion`.

Property	Class	Description
`annotations`	boolean [r/o]	Determines whether to include annotations.
`binary`	boolean [r/o]	Determines whether the output file should be in binary or ASCII text format.
`embedded fonts`	boolean [r/o]	Determines whether to include fonts.
`halftones`	boolean [r/o]	Determines whether to halftone screens.
`images`	boolean [r/o]	Determines whether to include RGB and LAB images.
`postScript level`	integer [r/o]	The PostScript Language level. Only levels 2 and 3 are supported.
`preview`	boolean [r/o]	Determines whether to include preview in output.
`TrueType`	boolean [r/o]	Determines whether to convert TrueType fonts to Type 1.

Related Methods

`save`

Text Annotation

Description

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

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

Elements

None.

Properties

Inherits from **annotation**.

Related Methods

find next note

perform

replace pages

2 Apple Events

Required Suite

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

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

open

Description

Opens a file.

AppleScript Syntax

open reference

AppleScript Parameters

open	The file or files to open.

Return Value

None

print

Description

Prints one or more files.

AppleScript Syntax

`print` reference

AppleScript Parameters

`print`	The file or files to print.

Return Value

None

quit

Description

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

AppleScript Syntax

```
quit
```

AppleScript Parameters

None

Return Value

None

run

Description

Launches the application and invokes its standard startup procedures.

AppleScript Syntax

```
run
```

AppleScript Parameters

None

Return Value

None

Core Suite

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

close

Description

Closes a document.

AppleScript Syntax

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

AppleScript Parameters

`close`	The document to close.
`saving`	Determines whether to save a document that has been modified before quitting. Possible values: **`yes`** — Save the document. **`no`** — Do not save the document. **`ask`** — Ask the user whether to save the document. The default value is **`ask`**.
`linearize`	Determines whether the document should be linearized when saving before closing.

Return Value

None

Related Events

`open`

count

Description

Counts the number of instances of a particular class.

AppleScript Syntax

count type class **of** reference

AppleScript Parameters

count	The class whose instances are to be counted.
each	The class whose instances are to be counted. **NOTE:** The keyword **each** is optional.

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

count reference **each** type class

Return Value

An integer specifying the number of elements.

AppleScript Example

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

delete

Description

Deletes one or more objects.

AppleScript Syntax

`delete` reference

AppleScript Parameters

`delete`	The object to be deleted.

Return Value

None

Related Events

`make`

`exists`

AppleScript Example

```
delete first bookmark of document "test.pdf"
```

exists

Description

Tests whether a specified object exists.

AppleScript Syntax

reference **`exists`**

`exists` reference

AppleScript Parameters

`exists`	Object whose existence is checked.

Return Value

`true` if the object exists, **`false`** otherwise.

AppleScript Example

```
exists second document
second document exists
```

get

Description

Retrieves the value of an object or property.

AppleScript Syntax

[**get**] reference [**as** class]

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

AppleScript Parameters

get	The object or property whose value is returned.
as	The form in which the data is returned.

Return Value

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

Related Events

set

AppleScript Example

```
get the name of last bookmark
get the index of last bookmark as string
```

make

Description

Creates a new object.

AppleScript Syntax

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

AppleScript Parameters

make [new]	The class of the new object.
at	The location at which to insert the new object.
with data	The initial data for the new object.
with properties	The initial values for the properties of the new object.

Return Value

A reference to the newly created object.

Related Events

delete

exists

AppleScript Example

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

move

Description

Moves a **page** object.

AppleScript Syntax

move reference **to** location reference

AppleScript Parameters

move	The page object to move. The first page in a document is page 1.
to	The new location for the page.

Return Value

A reference to the page that is moved.

AppleScript Example

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

open

Description

Opens a document or documents.

AppleScript Syntax

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

AppleScript Parameters

open	The document or documents to open.
invisible	Whether the opened document should be hidden. Default is **false**.
options	Optional parameter string of open actions.

Return Value

None

Related Events

close

quit

Description

Causes the Acrobat application to quit.

AppleScript Syntax

quit [**saving** constant]

AppleScript Parameters

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

Return Value

None

AppleScript Example

```
quit saving yes
```

save

Description

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

AppleScript Syntax

save reference [**to** file specification] [**using** reference] [**linearize** boolean]

AppleScript Parameters

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

Return Value

None

AppleScript Example

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

set

Description

Sets an object's data or properties.

AppleScript Syntax

`set` reference `to` anything

AppleScript Parameters

`set`	The object or property whose value is set.
`to`	The new value.

Return Value

None

Related Events

`get`

AppleScript Example

```
set the name of first bookmark to "Chapter 1"
```

Acrobat application Suite

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

bring to front

Description

Brings the specified document's window to the front.

AppleScript Syntax

`bring to front` reference

AppleScript Parameters

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

Return Value

None

AppleScript Example

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

Apple Event ID

`kAEBringToFront ('bfrt')`

clear selection

Description

Clears the document's current selection, if any.

AppleScript Syntax

`clear selection` reference

AppleScript Parameters

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

Return Value

None

Related Events

`select text`

AppleScript Example

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

Apple Event ID

```
kAEClearSelection ('clsl')
```

close all docs

Description

Closes all documents.

AppleScript Syntax

close all docs [**saving** constant]

AppleScript Parameters

saving	Determines whether to save modified documents before closing. Possible values: **yes** — Save the document. **no** — Do not save the document. **ask** — If the document has been modified, ask the user whether to save it. The default value is **ask.**

Return Value

None

Related Events

open (required suite)

open (core suite)

AppleScript Example

```
close all docs
```

Apple Event ID

```
kAECloseAllDocs ('cldc')
```

create thumbs

Description

Creates thumbnail images for all pages in the document.

AppleScript Syntax

`create thumbs` reference

AppleScript Parameters

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

Return Value

None

Related Events

`delete thumbs`

AppleScript Example

```
create thumbs document "roadmap.pdf"
```

Apple Event ID

`kAECreateThumbs ('crtb')`

delete pages

Description

Deletes the specified pages in the document.

AppleScript Syntax

`delete pages` reference **`first`** integer **`last`** integer

AppleScript Parameters

`delete pages`	The document containing the pages to be deleted.
`first`	The first page to be deleted. The first page in a document is page 1.
`last`	The last page to be deleted.

Return Value

None

Related Events

`insert pages`

`replace pages`

AppleScript Example

```
delete pages document "AppleEvt.pdf" first 1 last 3
```

Apple Event ID

```
kAEDeletePages ('dlpg')
```

Apple Event Parameters

```
keyAEFirstPage ('frpg')
keyAELastPage ('lapg')
```

delete thumbs

Description

Deletes all thumbnails from the document.

AppleScript Syntax

`delete thumbs` reference

AppleScript Parameters

`delete thumbs`	The document from which thumbnails are deleted.

Return Value

None

Related Events

`create thumbs`

AppleScript Example

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

Apple Event ID

`kAEDeleteThumbs ('dltb')`

execute

Description

Executes the specified menu item.

AppleScript Syntax

execute reference

AppleScript Parameters

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

Return Value

None

AppleScript Example

```
activate
execute menu item "Open"
```

Apple Event ID

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

find next note

Description

Finds and selects the next text note in a document.

AppleScript Syntax

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

AppleScript Parameters

`find next note`	The document in which to find the next text note.
`wrap around`	Determines whether to continue the search at the beginning of a document if a note has not been found after the end of the document is reached. If **`true`**, the search wraps around; otherwise it does not. The default value is **`false`**.

Return Value

The text annotation found.

Related Events

`find text`

AppleScript Example

```
find next note document "dev_acro.pdf"
```

Apple Event ID

`kAEFindNextNote ('fnnt')`

Apple Event Parameters

`keyAEWrapAround ('wrar')`

find text

Description

Finds text in a document.

AppleScript Syntax

find text reference **string** international text [**case sensitive** boolean] [**whole words** boolean] [**wrap around** boolean]

AppleScript Parameters

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

Return Value

None

Related Events

find next note

AppleScript Example

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

Apple Event ID

```
kAEFindText ('ftxt')
```

Apple Event Parameters

```
keyAESearchString ('sstr')
keyAECaseSensitive ('case')
keyAEWholeWordsOnly ('whwd')
keyAEWrapAround ('wrar')
```

get info

Description

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

AppleScript Syntax

`get info` reference **`key`** international text

AppleScript Parameters

`get info`	The document from which to obtain the **Info** dictionary entry.
`key`	The case-sensitive **Info** dictionary key whose value is to be obtained. The predefined keys are: **`Creator`**, **`Producer`**, **`CreationDate`**, **`Author`**, **`Title`**, **`Subject`**, and **`Keywords`**. None of these is required in the PDF file.

Return Value

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

AppleScript Example

```
get info document "PLUGINS.PDF" key "CreationDate"
```

Apple Event ID

```
kAEGetInfo ('gnfo')
```

Apple Event Parameters

```
keyAEInfoKey ('inky')
```

go backward

Description

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

AppleScript Syntax

go backward reference

AppleScript Parameters

go backward	A **PDF Window** object

Return Value

None

Related Events

go forward

goto

goto next

goto previous

AppleScript Example

```
go backward first PDF Window
```

Apple Event ID

kAEGoBack ('gbck')

go forward

Description

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

AppleScript Syntax

go forward reference

AppleScript Parameters

go forward	A **PDF Window** object

Return Value

None

Related Events

go backward

goto

goto next

goto previous

AppleScript Example

```
go forward first PDF Window
```

Apple Event ID

kAEGoForward ('gfwd')

goto

Description

Displays the page that has the specified page number.

AppleScript Syntax

goto reference **page** integer

AppleScript Parameters

goto	The **PDF Window** object in which to change the page.
page	The page number of the page to be displayed. The first page in a document is page 1.

Return Value

None

Related Events

go backward

go forward

goto next

goto previous

AppleScript Example

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

Apple Event ID

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

Apple Event Parameters

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

goto next

Description

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

AppleScript Syntax

goto next reference

AppleScript Parameters

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

Return Value

None

Related Events

go backward

go forward

goto

goto previous

AppleScript Example

```
goto next first PDF Window
```

Apple Event ID

```
kAEGotoNextPage ('nxpg')
```

goto previous

Description

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

AppleScript Syntax

goto previous reference

AppleScript Parameters

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

Return Value

None

Related Events

go backward

go forward

goto

goto next

AppleScript Example

```
goto previous first PDF Window
```

Apple Event ID

```
kAEGotoPrevPage ('pvpg')
```

insert pages

Description

Inserts one or more pages from one document into another.

AppleScript Syntax

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

AppleScript Parameters

`insert pages`	The target document in which to insert the page or pages.
`after`	The number of the page after which the pages will be inserted. The first page in a document is page 1.
`from`	The source document containing the page or pages to be inserted.
`starting with`	The first page to be inserted.
`number of pages`	The number of pages to be inserted.
`insert bookmarks`	Determines whether to copy bookmarks that point to the inserted pages. Default is **`true`**.

Return Value

None

Related Events

`delete pages`

AppleScript Example

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

Apple Event ID

```
kAEInsertPages ('inpg')
```

Apple Event Parameters

```
keyAEInsertAfter ('inaf')
keyAESourceDoc ('srdc')
kAESourceStartPage ('stpg')
keyAENumPages ('nmpg')
keyAEInsertBookmarks ('inbm')
```

is toolbutton enabled

Description

Determines whether the specified toolbar button is enabled.

AppleScript Syntax

`is toolbutton enabled named` international text

AppleScript Parameters

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

Return Value

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

Related Events

`remove toolbutton`

AppleScript Example

```
is toolbutton enabled named "AcroSrch:Query"
```

Apple Event ID

`kAEIsToolButtonEnabled ('tben')`

Apple Event Parameters

`keyAEButtonname ('tbnm')`

maximize

Description

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

AppleScript Syntax

`maximize` reference **`max size`** boolean

AppleScript Parameters

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

Return Value

None

AppleScript Example

```
maximize document "AppleEvt.pdf" max size false
```

Apple Event ID

```
kAEMaximize ('maxi')
```

Apple Event Parameters

```
keyAEMaxSize ('mxsz')
```

perform

Description

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

AppleScript Syntax

`perform` reference

AppleScript Parameters

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

Return Value

None

AppleScript Example

```
perform last bookmark
```

Apple Event ID

```
kAEPerform ('prfm')
```

print pages

Description

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

AppleScript Syntax

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

AppleScript Parameters

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

Return Value

None

AppleScript Example

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

Apple Event ID

```
kAEPrintPages ('prpg')
```

Apple Event Parameters

```
keyAEFirstPage ('frpg')
keyAELastPage ('lapg')
keyAEPSLevel ('pslv')
keyAEBinaryOK ('binO')
keyAEShrinkToFit ('s2ft')
```

read page down

Description

Scrolls forward through the document by one screen.

AppleScript Syntax

`read page down` reference

AppleScript Parameters

`read page down`	The **`PDF Window`** object to be scrolled.

Return Value

None

Related Events

`read page up`

`scroll`

AppleScript Example

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

Apple Event ID

```
kAEReadPageDown ('pgdn')
```

read page up

Description

Scrolls backward through the document by one screen.

AppleScript Syntax

`read page up` reference

AppleScript Parameters

`read page up`	The **`PDF Window`** object to be scrolled.

Return Value

None

Related Events

`read page down`

`scroll`

AppleScript Example

```
read page up first PDFPageWindow
```

Apple Event ID

```
kAEReadPageUp ('pgup')
```

remove toolbutton

Description

Removes the specified button from the toolbar.

AppleScript Syntax

`remove toolbutton named` international text

AppleScript Parameters

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

Return Value

None

Related Events

`is toolbutton enabled`

AppleScript Example

```
remove toolbutton named "ZoomIn"
```

Apple Event ID

`kAERemoveToolButton ('rmtb')`

Apple Event Parameters

`keyAEButtonname ('tbnm')`

replace pages

Description

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

AppleScript Syntax

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

AppleScript Parameters

`replace pages`	The target document whose pages are to be replaced.
`over`	The first page to be replaced. The first page in a document is page 1.
`from`	The source document from which the replacement page or pages are obtained.
`starting with`	The first page in the source document to be copied.
`number of pages`	The number of pages to be replaced.
`merge notes`	Determines whether to copy notes from the source document. The default value is **`true`**.

Return Value

None

Related Events

`delete pages`

`insert pages`

AppleScript Example

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

Apple Event ID

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

Apple Event Parameters

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

scroll

Description

Scrolls the view of a page by the specified amount.

AppleScript Syntax

scroll reference **X Amount** integer **Y Amount** integer

AppleScript Parameters

scroll	The **PDF Window** object in which to scroll the view.
X Amount	The amount to scroll in the horizontal direction, in pixels. Positive values move the view to the right.
Y Amount	The amount to scroll in the vertical direction, in pixels. Positive values move the view down.

Return Value

None

Related Events

read page down

read page up

AppleScript Example

```
scroll first PDFWindow X Amount 20 Y Amount 100
```

Apple Event ID

```
kAEScroll ('scrl')
```

Apple Event Parameters

```
keyAEXDelta ('xdlt')
keyAEYDelta ('ydlt')
```

select text

Description

Selects text as specified by either character or word offsets.

AppleScript Syntax

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

AppleScript Parameters

`select text`	The **`PDF Window`** object in which to select text.
`from words`	The words to be selected. This consists of one or more pairs of word offsets from the beginning of the document and word lengths (the number of contiguous words).
`from chars`	Characters to be selected. This consists of one or more pairs of character offsets from the beginning of the document and character lengths (the number of contiguous characters).

Return Value

None

Related Events

`clear selection`

AppleScript Example

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

Apple Event ID

`kAESetTextSelection ('stxs')`

Apple Event Parameters

```
keyAEWordList ('fmwd')
keyAECharList ('fmch')
```

set info

Description

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

AppleScript Syntax

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

AppleScript Parameters

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

Return Value

None

AppleScript Example

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

Apple Event ID

```
kAESetInfo ('snfo')
```

Apple Event Parameters

```
keyAEInfoKey ('inky')
keyAEInfoValue ('invl')
```

zoom

Description

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

AppleScript Syntax

zoom reference **to** small real

AppleScript Parameters

zoom	The **PDF Window** object to be zoomed.
to	The zoom factor specified as a percentage. For example, a value of 100 (100%) displays the document with a magnification of 1.0.

Return Value

None

AppleScript Example

```
zoom first PDFWindow to 150
```

Apple Event ID

kAEZoomTo ('zmto')

Apple Event Parameters

keyAEZoomFactor ('zmft')

Miscellaneous Apple Events

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

do script

Description

Executes the specified Acrobat JavaScript script.

AppleScript Syntax

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

AppleScript Parameters

`do script`	The Acrobat JavaScript script to be executed.
`file`	File holding the JavaScript script to be executed.

Return Value

Result of JavaScript execution as text.

AppleScript Example

```
do script MyJavaScriptFile.js
```

OLE Automation

This reference contains the following sections:

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

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

The syntax used in this section follows that used in Microsoft Visual Basic 3.0.

3 OLE Automation Objects

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

AcroExch.App

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

AcroExch.AVDoc

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

AcroExch.AVPageView

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

AcroExch.Hilite

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

AcroExch.PDAnnot

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

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

AcroExch.PDBookmark

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

AcroExch.PDDoc

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

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

AcroExch.PDPage

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

AcroExch.PDTextSelect

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

AxAcroPDFLib.AxAcroPDF

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

4 OLE Automation Methods

AcroExch.App

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

CloseAllDocs

```
VARIANT_BOOL CloseAllDocs();
```

Description

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

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

Parameters

None

Return Value

-1 if successful, **0** if not.

Related Methods

AVDoc.Close

AVDoc.Open

AVDoc.OpenInWindow

AVDoc.OpenInWindowEx

PDDoc.Close

PDDoc.Open

PDDoc.OpenAVDoc

Exit

```
VARIANT_BOOL Exit();
```

Description

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

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

Parameters

None

Return Value

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

Related Methods

App.CloseAllDocs

GetActiveDoc

```
LPDISPATCH GetActiveDoc();
```

Description

Gets the frontmost document.

Parameters

None

Return Value

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

Related Methods

App.GetAVDoc

GetActiveTool

```
BSTR GetActiveTool();
```

Description

Gets the name of the currently active tool.

Parameters

None

Return Value

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

Related Methods

App.SetActiveTool

GetAVDoc

```
LPDISPATCH GetAVDoc(long nIndex);
```

Description

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

Parameters

nIndex	The index of the document to get.

Return Value

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

Related Methods

App.GetActiveTool

GetFrame

```
LPDISPATCH GetFrame();
```

Description

Gets the window's frame.

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

Parameters

None

Return Value

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

Related Methods

App.Maximize

App.SetFrame

GetInterface

```
LPDISPATCH GetInterface (BSTR szName);
```

Description

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

Parameters

szName	Name of the object.

Return Value

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

Related Methods

None.

GetLanguage

```
BSTR GetLanguage();
```

Description

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

Parameters

None

Return Value

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

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

Related Methods

App.GetPreference

App.SetPreference

GetNumAVDocs

```
long GetNumAVDocs();
```

Description

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

Parameters

None

Return Value

The number of open **AcroExch.AVDoc** objects.

Related Methods

App.**GetActiveDoc**

App.**GetAVDoc**

GetPreference

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

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

Description

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

Parameters

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

Return Value

The value of the specified preference item.

Related Methods

App.GetLanguage

App.SetPreference

GetPreferenceEx

```
VARIANT GetPreferenceEx(short nType);
```

Description

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

Parameters

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

Return Value

The value of the specified preference item.

Related Methods

App.GetLanguage

App.SetPreferenceEx

Hide

```
VARIANT_BOOL Hide();
```

Description

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

Parameters

None

Return Value

-1 if successful, **0** if not.

Related Methods

App.Show

Lock

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

Description

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

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

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

Parameters

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

Return Value

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

Related Methods

App.UnlockEx

Minimize

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

Description

Minimizes the Acrobat application.

Parameters

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

Return Value

-1 if successful, **0** if not.

Related Methods

App.GetFrame

App.SetFrame

Maximize

```
VARIANT_BOOL Maximize(long bMaximize);
```

Description

Maximizes the Acrobat application.

Parameters

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

Return Value

-1 if successful, **0** if not.

Related Methods

App.GetFrame

App.SetFrame

MenuItemExecute

```
VARIANT_BOOL MenuItemExecute(BSTR 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 the *Acrobat And PDF Library API Reference* for a list of menu item names.

Return Value

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

Related Methods

App.MenuItemIsEnabled

App.MenuItemIsMarked

App.MenuItemRemove

MenuItemIsEnabled

```
VARIANT_BOOL MenuItemIsEnabled(BSTR szMenuItemName);
```

Description

Determines whether the specified menu item is enabled.

Parameters

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

Return Value

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

Related Methods

App.**MenuItemExecute**

App.**MenuItemIsMarked**

App.**MenuItemRemove**

MenuItemIsMarked

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

Description

Determines whether the specified menu item is marked.

Parameters

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

Return Value

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

Related Methods

App.**MenuItemExecute**

App.**MenuItemIsEnabled**

App.**MenuItemRemove**

MenuItemRemove

```
VARIANT_BOOL MenuItemRemove(BSTR szMenuItemName);
```

Description

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

Parameters

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

Return Value

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

Related Methods

App.**MenuItemExecute**

App.**MenuItemIsEnabled**

App.**MenuItemIsMarked**

Restore

```
VARIANT_BOOL Restore(long bRestore);
```

Description

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

Parameters

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

Return Value

-1 if successful, **0** if not.

Related Methods

App.GetFrame

App.SetFrame

SetActiveTool

```
VARIANT_BOOL SetActiveTool(BSTR szButtonName,
                           long bPersistent);
```

Description

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

Parameters

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

Return Value

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

Related Methods

App.**GetActiveTool**

App.**ToolButtonIsEnabled**

App.**ToolButtonRemove**

SetFrame

```
VARIANT_BOOL SetFrame(LPDISPATCH iAcroRect);
```

Description

Sets the window's frame to the specified rectangle.

Parameters

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

Return Value

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

Related Methods

App.GetFrame

App.Maximize

SetPreference

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

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

Description

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

Parameters

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

Return Value

-1 if successful, **0** if not.

Related Methods

App.GetLanguage

App.GetPreferenceEx

SetPreferenceEx

```
VARIANT_BOOL SetPreferenceEx(short nType, VARIANT* pVal);
```

Description

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

Parameters

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

Return Value

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

Related Methods

App.GetLanguage

App.GetPreferenceEx

Show

```
VARIANT_BOOL Show();
```

Description

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

Parameters

None

Return Value

-1 if successful, **0** if not.

Related Methods

App.Hide

ToolButtonIsEnabled

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

Description

Determines whether the specified toolbar button is enabled.

Parameters

szButtonName	The name of the button whose enabled state is checked. See the *Acrobat And PDF Library API Reference* for a list of toolbar button names.

Return Value

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

Related Methods

App.GetActiveTool

App.SetActiveTool

App.ToolButtonRemove

ToolButtonRemove

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

Description

Removes the specified button from the toolbar.

Parameters

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

Return Value

-1 if the button was removed, **0** otherwise.

Related Methods

App.GetActiveTool

App.SetActiveTool

App.ToolButtonIsEnabled

Unlock

```
VARIANT_BOOL Unlock();
```

Description

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

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

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

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

Parameters

None

Return Value

-1 if successful, **0** if not.

Related Methods

App.Lock

App.UnlockEx

UnlockEx

```
VARIANT_BOOL UnlockEx (BSTR szLockedBy);
```

Description

Unlocks the Acrobat application if it was previously locked.

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

Parameters

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

Return Value

-1 if successful, **0** if not.

Related Methods

App.Lock

Parameters

AcroExch.AVDoc

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

BringToFront

```
VARIANT_BOOL BringToFront();
```

Description

Brings the window to the front.

Parameters

None

Return Value

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

Related Methods

None

ClearSelection

```
VARIANT_BOOL ClearSelection();
```

Description

Clears the current selection.

Parameters

None

Return Value

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

Related Methods

AVDoc.SetTextSelection

AVDoc.ShowTextSelect

PDDoc.CreateTextSelect

PDPage.CreatePageHilite

PDPage.CreateWordHilite

PDTextSelect.Destroy

PDTextSelect.GetBoundingRect

PDTextSelect.GetNumText

PDTextSelect.GetPage

PDTextSelect.GetText

Close

```
VARIANT_BOOL Close(long bNoSave);
```

Description

Closes a document. You can close all open **AVDoc** objects by calling **App**.**CloseAllDocs**.

To reuse an **AVDoc** object, close it with **AVDoc**.**Close**, then use the **AVDoc** object's **LPDISPATCH** for **AVDoc**.**OpenInWindow**.

Parameters

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

Return Value

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

Related Methods

App.**CloseAllDocs**

AVDoc.**Open**

AVDoc.**OpenInWindow**

AVDoc.**OpenInWindowEx**

PDDoc.**Close**

PDDoc.**Open**

PDDoc.**OpenAVDoc**

FindText

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

Description

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

Parameters

szText	The text to be found.
bCaseSensitive	If a positive number, the search is case-sensitive. If **0**, it is case-insensitive.
bWholeWordsOnly	If a positive number, the search matches only whole words. If **0**, it matches partial words.
bReset	If a positive number, the search begins on the first page of the document. If **0**, it begins on the current page.

Return Value

-1 if the text was found, **0** otherwise.

Related Methods

None

GetAVPageView

```
LPDISPATCH GetAVPageView();
```

Description

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

Parameters

None

Return Value

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

Related Methods

AVDoc.GetPDDoc

AVDoc.SetViewMode

AVPageView.GetAVDoc

AVPageView.GetDoc

GetFrame

```
LPDISPATCH GetFrame();
```

Description

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

Parameters

None

Return Value

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

Related Methods

AVDoc.SetFrame

GetPDDoc

```
LPDISPATCH GetPDDoc();
```

Description

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

Parameters

None

Return Value

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

Related Methods

AVDoc.GetAVPageView

AVPageView.GetAVDoc

AVPageView.GetDoc

GetTitle

```
BSTR GetTitle();
```

Description

Gets the window's title.

Parameters

None

Return Value

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

Related Methods

AVDoc.Open

AVDoc.SetTitle

PDDoc.OpenAVDoc

GetViewMode

```
long GetViewMode();
```

Description

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

Parameters

None

Return Value

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

Related Methods

AVDoc.GetAVPageView

AVDoc.SetViewMode

IsValid

```
VARIANT_BOOL IsValid();
```

Description

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

Parameters

None

Return Value

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

Related Methods

App.GetAVDoc

AVPageView.GetAVDoc

Maximize

```
VARIANT_BOOL Maximize(long bMaxSize);
```

Description

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

Parameters

bMaxSize	Indicates whether window should be maximized.

Return Value

-1 if a document is open, **0** otherwise.

Related Methods

AVDoc.GetFrame

AVDoc.SetFrame

Open

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

Description

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

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

Parameters

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

Return Value

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

Related Methods

App.CloseAllDocs

AVDoc.Close

AVDoc.GetTitle

AVDoc.OpenInWindow

AVDoc.OpenInWindowEx

AVDoc.SetTitle

PDDoc.Close

PDDoc.Open

PDDoc.OpenAVDoc

OpenInWindow

```
VARIANT_BOOL OpenInWindow(BSTR fileName, short hWnd);
```

Description

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

Parameters

`fileName`	The full pathname of the file to open.
`hWnd`	Handle for the window in which the file is displayed.

Return Value

`-1`.

Related Methods

App.`CloseAllDocs`

AVDoc.`Close`

AVDoc.`Open`

AVDoc.`OpenInWindowEx`

PDDoc.`Close`

PDDoc.`Open`

PDDoc.`OpenAVDoc`

OpenInWindowEx

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

Description

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

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

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

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

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

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

Parameters

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

Return Value

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

Related Methods

App.CloseAllDocs

AVDoc.Close

AVDoc.Open

AVDoc.OpenInWindow

PDDoc.Close

PDDoc.Open

PDDoc.OpenAVDoc

PrintPages

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

Description

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

Parameters

nFirstPage	The first page to be printed. The first page in a **PDDoc** object is page 0.
nLastPage	The last page to be printed.
nPSLevel	Valid values are 2 and 3. If 2, PostScript Level 2 operators are used. If 3, PostScript Language Level 3 operators are also used.
bBinaryOk	If a positive number, binary data may be included in the PostScript program. If **0**, all data is encoded as 7-bit ASCII.
bShrinkToFit	If a positive number, the page is shrunk (if necessary) to fit within the imageable area of the printed page. If **0**, it is not.

Return Value

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

Related Methods

AVDoc.PrintPagesEx

AVDoc.PrintPagesSilent

AVDoc.PrintPagesSilentEx

PrintPagesEx

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

Description

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

Parameters

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

Return Value

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

Related Methods

AVDoc.PrintPages

AVDocPrintPagesSilent

AVDoc..PrintPagesSilentEx

PrintPagesSilent

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

Description

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

Parameters

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

Return Value

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

Related Methods

AVDoc.PrintPages

AVDoc.PrintPagesEx

AVDoc..PrintPagesSilentEx

PrintPagesSilentEx

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

Description

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

Parameters

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

Return Value

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

Related Methods

AVDoc.**PrintPages**

AVDoc.**PrintPagesEx**

AVDoc.**PrintPagesSilentEx**

SetFrame

```
VARIANT_BOOL SetFrame(LPDISPATCH iAcroRect);
```

Description

Sets the window's size and location.

Parameters

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

Return Value

Always returns **-1**.

Related Methods

AVDoc.GetFrame

SetTextSelection

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

Description

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

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

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

Parameters

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

Return Value

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

Related Methods

AVDoc.ClearSelection

AVDoc.ShowTextSelect

PDDoc.CreateTextSelect

PDPage.CreatePageHilite

PDPage.CreateWordHilite

PDTextSelect.Destroy

PDTextSelect.GetBoundingRect

PDTextSelect.GetNumText

PDTextSelect.GetPage

PDTextSelect.GetText

SetTitle

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

Description

Sets the window's title.

Parameters

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

Return Value

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

Related Methods

AVDoc.GetTitle

AVDoc.Open

PDDoc.OpenAVDoc

SetViewMode

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

Description

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

Parameters

nType	The view mode to be set. Must be one of the values (except for **PDFullScreen***)* listed in View Mode (see document in Chapter 1, "Apple Event Objects" for an explanation of the View Mode property). Possible values: **PDDontCare** (**0** - leave the view mode as it is) **PDUseNone** (**1**- display without bookmarks or thumbnails) **PDUseThumbs** (**2** - display using thumbnails) **PDUseBookmarks** (**3** - display using bookmarks) **PDFullScreen** (**4** - display in full screen mode) **NOTE:** Do not set the view mode with **AVDoc.SetViewMode**.

Return Value

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

Related Methods

AVDoc.GetAVPageView

AVDoc.GetViewMode

ShowTextSelect

```
VARIANT_BOOL ShowTextSelect();
```

Description

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

Parameters

None

Return Value

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

Related Methods

AVDoc.ClearSelection

AVDoc.SetTextSelection

PDDoc.CreateTextSelect

PDPage.CreatePageHilite

PDPage.CreateWordHilite

PDTextSelect.Destroy

PDTextSelect.GetBoundingRect

PDTextSelect.GetNumText

PDTextSelect.GetPage

PDTextSelect.GetText

AcroExch.AVPageView

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

DevicePointToPage

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

Description

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

Parameters

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

Return Value

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

Related Methods

AVPageView.PointToDevice

DoGoBack

```
VARIANT_BOOL DoGoBack();
```

Description

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

Parameters

None

Return Value

Always returns **-1**.

Related Methods

AVPageView.DoGoForward

DoGoForward

```
VARIANT_BOOL DoGoForward();
```

Description

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

Parameters

None

Return Value

Always returns **-1**.

Related Methods

AVPageView.DoGoBack

GetAperture

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

Description

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

Parameters

None

Return Value

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

Related Methods

AVDoc.GetAVPageView

AVPageView.GetAVDoc

AVPageView.GetDoc

AVPageView.GetPage

AVPageView.GetZoomType

GetAVDoc

```
LPDISPATCH GetAVDoc();
```

Description

Gets the **AcroExch.AVDoc** associated with the current page.

Parameters

None

Return Value

The **LPDISPATCH** for the **AcroExch.AVDoc**.

Related Methods

AVDoc.**GetAVPageView**

AVDoc.GetPDDoc

AVPageView.GetDoc

GetDoc

```
LPDISPATCH GetDoc();
```

Description

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

Parameters

None

Return Value

The **LPDISPATCH** for the **AcroExch.PDDoc**.

Related Methods

AVDoc.GetAVPageView

AVDoc.GetPDDoc

AVPageView.GetAVDoc

GetPage

```
LPDISPATCH GetPage();
```

Description

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

Parameters

None

Return Value

The **LPDISPATCH** for the **AcroExch.PDPage**.

Related Methods

AVPageView.GetPageNum

PDDoc.AcquirePage

PDDoc.GetNumPages

PDPage.GetDoc

PDPage.GetNumber

PDPage.GetRotate

PDPage.GetSize

PDTextSelect.GetPage

GetPageNum

```
long GetPageNum();
```

Description

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

Parameters

None

Return Value

The current page's page number.

Related Methods

AVPageView.GetPage

PDDoc.AcquirePage

PDDoc.GetNumPages

PDPage.GetDoc

PDPage.GetNumber

PDPage.GetRotate

PDPage.GetSize

PDTextSelect.GetPage

GetZoom

```
long GetZoom();
```

Description

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

Parameters

None

Return Value

The current zoom factor.

Related Methods

App.GetPreference

AVPageView.GetZoomType

AVPageView.ZoomTo

GetZoomType

```
short GetZoomType();
```

Description

Gets the current zoom type.

Parameters

None

Return Value

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

Related Methods

App.GetPreference

AVPageView.GetZoomType

AVPageView.ZoomTo

Goto

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

Description

Goes to the specified page.

Parameters

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

Return Value

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

Related Methods

AVPageView.DoGoBack

AVPageView.DoGoForward

AVPageView.ReadPageDown

AVPageView.ReadPageUp

AVPageView.ScrollTo

AVPageView.ZoomTo

PointToDevice

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

Description

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

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

Parameters

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

Return Value

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

Related Methods

AVPageView.DevicePointToPage

ReadPageDown

```
VARIANT_BOOL ReadPageDown();
```

Description

Scrolls forward through the document by one screen area.

Parameters

None

Return Value

Always returns **-1**.

Related Methods

AVPageView.DoGoBack

AVPageView.DoGoForward

AVPageView.Goto

AVPageView.ReadPageUp

AVPageView.ScrollTo

AVPageView.ZoomTo

ReadPageUp

```
VARIANT_BOOL ReadPageUp();
```

Description

Scrolls backward through the document by one screen area.

Parameters

None

Return Value

Always returns **-1**.

Related Methods

AVPageView.DoGoBack

AVPageView.DoGoForward

AVPageView.Goto

AVPageView.ReadPageDown

AVPageView.ScrollTo

AVPageView.ZoomTo

ScrollTo

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

Description

Scrolls to the specified location on the current page.

Parameters

nX	*x*–coordinate of the destination.
nY	*y*–coordinate of the destination.

Return Value

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

Related Methods

AVPageView.DoGoBack

AVPageView.DoGoForward

AVPageView.Goto

AVPageView.ReadPageDown

AVPageView.ReadPageUp

AVPageView.ZoomTo

ZoomTo

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

Description

Zooms to the specified magnification.

Parameters

nType	Zoom type. See zoom type (in document in Chapter 1, "Apple Event Objects) for a list of zoom types.
nScale	The desired zoom factor, expressed as a percentage (for example, 100 is a magnification of 1.0).

Return Value

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

Related Methods

AVPageView.GetZoomType

AVPageView.Goto

AVPageView.ScrollTo

AcroExch.HiliteList

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

Add

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

Description

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

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

Parameters

nOffset	Offset of the first word or character to be highlighted, the first of which has an offset of zero.
nLength	The number of consecutive words or characters to be highlighted.

Return Value

Always returns **-1**.

Related Methods

PDPage.CreatePageHilite

PDPage.CreateWordHilite

AcroExch.PDAnnot

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

GetColor

```
long GetColor();
```

Description

Gets an annotation's color.

Parameters

None

Return Value

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

Related Methods

`PDAnnot`.`SetColor`

GetContents

```
BSTR GetContents();
```

Description

Gets a text annotation's contents.

Parameters

None

Return Value

The annotation's contents.

Related Methods

PDAnnot.SetContents

PDAnnot.GetDate

PDAnnot.GetRect

PDAnnot.GetSubtype

PDAnnot.GetTitle

GetDate

```
LPDISPATCH GetDate();
```

Description

Gets an annotation's date.

Parameters

None

Return Value

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

Related Methods

PDAnnot.GetContents

PDAnnot.GetRect

PDAnnot.GetSubtype

PDAnnot.GetTitle

PDAnnot.SetDate

GetRect

```
LPDISPATCH GetRect();
```

Description

Gets an annotation's bounding rectangle.

Parameters

None

Return Value

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

Related Methods

PDAnnot.GetContents

PDAnnot.GetDate

PDAnnot.GetSubtype

PDAnnot.GetTitle

PDAnnot.SetRect

GetSubtype

```
BSTR GetSubtype();
```

Description

Gets an annotation's subtype.

Parameters

None

Return Value

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

Related Methods

PDAnnot.GetContents

PDAnnot.GetDate

PDAnnot.GetRect

PDAnnot.GetTitle

GetTitle

```
BSTR GetTitle();
```

Description

Gets a text annotation's title.

Parameters

None

Return Value

The annotation's title.

Related Methods

PDAnnot.GetContents

PDAnnot.GetDate

PDAnnot.GetRect

PDAnnot.GetSubtype

PDAnnot.SetTitle

IsEqual

```
VARIANT_BOOL IsEqual(LPDISPATCH PDAnnot);
```

Description

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

Parameters

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

Return Value

-1 if the annotations are the same, **0** otherwise.

Related Methods

PDAnnot.GetContents

PDAnnot.GetDate

PDAnnot.GetRect

PDAnnot.GetSubtype

PDAnnot.GetTitle

PDAnnot.IsOpen

PDAnnot.IsValid

IsOpen

```
VARIANT_BOOL IsOpen();
```

Description

Tests whether a text annotation is open.

Parameters

None

Return Value

-1 if open, **0** otherwise.

Related Methods

PDAnnot.GetContents

PDAnnot.GetDate

PDAnnot.GetRect

PDAnnot.GetSubtype

PDAnnot.GetTitle

PDAnnot.IsEqual

PDAnnot.IsValid

PDAnnot.SetOpen

IsValid

```
VARIANT_BOOL IsValid();
```

Description

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

Parameters

None

Return Value

-1 if the annotation is valid, **0** otherwise.

Related Methods

PDAnnot.GetContents

PDAnnot.GetDate

PDAnnot.GetRect

PDAnnot.GetSubtype

PDAnnot.GetTitle

PDAnnot.IsEqual

PDAnnot.IsOpen

Perform

```
VARIANT_BOOL Perform(LPDISPATCH iAcroAVDoc);
```

Description

Performs a link annotation's action.

Parameters

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

Return Value

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

Related Methods

PDAnnot.IsValid

SetColor

```
VARIANT_BOOL SetColor(long nRGBColor);
```

Description

Sets an annotation's color.

Parameters

nRGBColor	The color to use for the annotation.

Return Value

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

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

Related Methods

PDAnnot.GetColor

PDAnnot.SetContents

PDAnnot.SetDate

PDAnnot.SetOpen

PDAnnot.SetRect

PDAnnot.SetTitle

SetContents

```
VARIANT_BOOL SetContents(BSTR szContents);
```

Description

Sets a text annotation's contents.

Parameters

szContents	The contents to use for the annotation.

Return Value

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

Related Methods

PDAnnot.GetContents

PDAnnot.SetColor

PDAnnot.SetDate

PDAnnot.SetOpen

PDAnnot.SetRect

PDAnnot.SetTitle

SetDate

```
VARIANT_BOOL SetDate(LPDISPATCH iAcroTime);
```

Description

Sets an annotation's date.

Parameters

iAcroTime	The **LPDISPATCH** for the date and time to use for the annotation. **iAcroTime's** instance variable **m_lpDispatch** contains this **LPDISPATCH**.

Return Value

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

Related Methods

PDAnnot.GetTitle

PDAnnot.SetColor

PDAnnot.SetContents

PDAnnot.SetOpen

PDAnnot.SetRect

PDAnnot.SetTitle

SetOpen

```
VARIANT_BOOL SetOpen(long bIsOpen);
```

Description

Opens or closes a text annotation.

Parameters

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

Return Value

Always returns **-1**.

Related Methods

PDAnnot.IsOpen

PDAnnot.SetColor

PDAnnot.SetContents

PDAnnot.SetDate

PDAnnot.SetRect

PDAnnot.SetTitle

SetRect

```
VARIANT_BOOL SetRect(LPDISPATCH iAcroRect);
```

Description

Sets an annotation's bounding rectangle.

Parameters

iAcroRect	The **LPDISPATCH** for the bounding rectangle (**AcroExch.Rect**) to set. **iAcroRect** contains the instance variable **m_lpDispatch**, which contains the **LPDISPATCH**.

Return Value

-1 if a rectangle was supplied, **0** otherwise.

Related Methods

PDAnnot.GetRect

PDAnnot.SetColor

PDAnnot.SetContents

PDAnnot.SetDate

PDAnnot.SetOpen

PDAnnot.SetTitle

SetTitle

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

Description

Sets a text annotation's title.

Parameters

szTitle	The title to use.

Return Value

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

Related Methods

PDAnnot.GetByTitle

PDAnnot.SetColor

PDAnnot.SetContents

PDAnnot.SetDate

PDAnnot.SetOpen

PDAnnot.SetRect

AcroExch.PDBookmark

The methods in this section relate to Acrobat bookmarks.

Destroy

```
VARIANT_BOOL Destroy();
```

Description

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

Parameters

None

Return Value

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

Related Methods

PDBookmark.IsValid

GetByTitle

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

Description

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

Parameters

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

Return Value

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

Related Methods

PDBookmark.GetTitle

PDBookmark.SetTitle

Example

```
long b;
CAcroPDBoookmark;
bookmark = new AcroPDBookmark;
COleException e;

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

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

GetTitle

```
BSTR GetTitle();
```

Description

Gets a bookmark's title.

Parameters

None

Return Value

The title.

Related Methods

PDBookmark.GetByTitle

PDBookmark.SetTitle

IsValid

```
VARIANT_BOOL IsValid();
```

Description

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

Parameters

None

Return Value

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

Related Methods

PDBookmark.Destroy

Perform

```
VARIANT_BOOL Perform(LPDISPATCH iAcroAVDoc);
```

Description

Performs a bookmark's action.

Parameters

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

Return Value

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

Related Methods

PDBookmark.IsValid

SetTitle

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

Description

Sets a bookmark's title.

Parameters

szNewTitle	The title to set.

Return Value

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

Related Methods

PDBookmark.GetByTitle

PDBookmark.GetTitle

AcroExch.PDDoc

The methods in this section work with documents.

AcquirePage

```
LPDISPATCH AcquirePage(long nPage);
```

Description

Acquires the specified page.

Parameters

nPage	The number of the page to acquire. The first page is page 0.

Return Value

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

Related Methods

AVPageView.GetPage

AVPageView.GetPageNum

PDDoc.GetNumPages

PDPage.GetDoc

PDPage.GetNumber

PDPage.GetRotate

PDPage.GetSize

PDTextSelect.GetPage

ClearFlags

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

Description

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

Parameters

nFlags	Flags to be cleared. See **PDDoc.GetFlags** for a description of the flags. The flags **PDDocWasRepaired**, **PDDocNewMajorVersion**, **PDDocNewMinorVersion**, **and PDDocOldVersion** are read-only and cannot be cleared.

Return Value

Always returns **-1**.

Related Methods

PDDoc.GetFlags

PDDoc.SetFlags

Close

```
VARIANT_BOOL Close();
```

Description

Closes a file.

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

Parameters

None

Return Value

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

Related Methods

App.CloseAllDocs

AVDoc.Close

AVDoc.Open

AVDoc.OpenInWindow

AVDoc.OpenInWindowEx

PDDoc.Open

PDDoc.OpenAVDoc

Create

```
VARIANT_BOOL Create();
```

Description

Creates a new **AcroExch.PDDoc**.

Parameters

None

Return Value

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

Related Methods

None

CreateTextSelect

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

Description

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

Parameters

nPage	The page on which the selection is created. The first page in a **PDDoc** object is page 0.
iAcroRect	The **LPDISPATCH** for the **AcroExch.Rect** enclosing the region to select. **iAcroRect** contains the instance variable **m_lpDispatch**, which contains the **LPDISPATCH**.

Return Value

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

Related Methods

AVDoc.ClearSelection

AVDoc.SetTextSelection

AVDoc.ShowTextSelect

PDPage.CreatePageHilite

PDPage.CreateWordHilite

PDTextSelect.Destroy

PDTextSelect.GetBoundingRect

PDTextSelect.GetNumText

PDTextSelect.GetPage

PDTextSelect.GetText

CreateThumbs

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

Description

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

Parameters

nFirstPage	First page for which thumbnail images are created. The first page in a **PDDoc** object is page 0.
nLastPage	Last page for which thumbnail images are created.

Return Value

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

Related Methods

PDDoc.DeleteThumbs

CropPages

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

Description

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

Parameters

nStartPage	First page that is cropped. The first page in a **PDDoc** object is page 0.
nEndPage	Last page that is cropped.
nEvenOrOddPagesOnly	Value indicating which pages in the range are cropped. Must be one of the following: ● 0 means crop all pages in the range ● 1 means crop only odd pages in the range ● 2 means crop only even pages in the range
iAcroRect	An **LPDISPATCH** for a **CAcroRect** specifying the cropping rectangle, which is specified in user space.

Return Value

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

Related Methods

PDPage.CropPages

DeletePages

```
VARIANT_BOOL DeletePages(long nStartPage, long nEndPage);
```

Description

Deletes pages from a file.

Parameters

nStartPage	The first page to be deleted. The first page in a **PDDoc** object is page 0.
nEndPage	The last page to be deleted.

Return Value

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

Related Methods

PDDoc.AcquirePage

PDDoc.DeletePages

PDDoc.GetNumPages

PDDoc.InsertPages

PDDoc.MovePage

PDDoc.ReplacePages

DeleteThumbs

```
VARIANT_BOOL DeleteThumbs(long nStartPage, long nEndPage);
```

Description

Deletes thumbnail images from the specified pages in a document.

Parameters

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

Return Value

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

Related Methods

PDDoc.CreateThumbs

GetFileName

```
BSTR GetFileName();
```

Description

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

Parameters

None

Return Value

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

Related Methods

PDDoc.Save

GetFlags

```
long GetFlags();
```

Description

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

Parameters

None

Return Value

The document's flags, containing an OR of the following:

Flag	Description
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.

Related Methods

PDDoc.ClearFlags

PDDoc.SetFlags

GetInfo

```
BSTR GetInfo(BSTR szInfoKey);
```

Description

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

Parameters

szInfoKey	The key whose value is obtained.

Return Value

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

Related Methods

PDDoc.SetInfo

GetInstanceID

```
BSTR GetInstanceID();
```

Description

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

Parameters

None

Return Value

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

Related Methods

PDDoc.GetPermanentID

GetJSObject

```
LDispatch* GetJSObject();
```

Description

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

Parameters

None

Return Value

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

Related Methods

None.

GetNumPages

```
long GetNumPages();
```

Description

Gets the number of pages in a file.

Parameters

None

Return Value

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

Related Methods

AVPageView.GetPage

AVPageView.GetPageNum

PDDoc.AcquirePage

PDPage.GetNumber

PDTextSelect.GetPage

GetPageMode

```
long GetPageMode();
```

Description

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

Parameters

None

Return Value

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

Related Methods

PDDoc.SetPageMode

GetPermanentID

```
BSTR GetPermanentID();
```

Description

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

Parameters

None

Return Value

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

Related Methods

PDDoc.GetInstanceID

InsertPages

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

Description

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

Parameters

nInsertPageAfter	The page in the current document after which pages from the source document are inserted. The first page in a **PDDoc** object is page 0.
iPDDocSource	The **LPDISPATCH** for the **AcroExch.PDDoc** containing the pages to insert. **iPDDocSource** contains the instance variable **m_lpDispatch**, which contains the **LPDISPATCH**.
nStartPage	The first page in **iPDDocSource** to be inserted into the current document.
nNumPages	The number of pages to be inserted.
bBookmarks	If a positive number, bookmarks are copied from the source document. If **0**, they are not.

Return Value

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

Related Methods

PDDoc.AcquirePage

PDDoc.DeletePages

PDDoc.GetNumPages

PDDoc.MovePage

PDDoc.ReplacePages

MovePage

```
VARIANT_BOOL MovePage(long nMoveAfterThisPage,
                      long nPageToMove);
```

Description

Moves a page to another location within the same document.

Parameters

nMoveAfterThisPage	The page being moved is placed after this page number. The first page in a **PDDoc** object is page 0.
nPageToMove	Page number of the page to be moved.

Return Value

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

Related Methods

PDDoc.AcquirePage

PDDoc.DeletePages

PDDoc.GetNumPages

PDDoc.InsertPages

PDDoc.ReplacePages

Open

```
VARIANT_BOOL Open(BSTR szFullPath);
```

Description

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

Parameters

szFullPath	Full pathname of the file to be opened.

Return Value

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

Related Methods

App.CloseAllDocs

AVDoc.Close

AVDoc.Open

AVDoc.OpenInWindow

AVDoc.OpenInWindowEx

PDDoc.Close

PDDoc.OpenAVDoc

OpenAVDoc

```
LPDISPATCH OpenAVDoc(BSTR szTitle);
```

Description

Opens a window and displays the document in it.

Parameters

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

Return Value

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

Related Methods

App.CloseAllDocs

AVDoc.Close

AVDoc.GetTitle

AVDoc.Open

AVDoc.OpenInWindow

AVDoc.OpenInWindowEx

AVDoc.SetTitle

PDDoc.Close

PDDoc.Open

ReplacePages

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

Description

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

Parameters

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

Return Value

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

Related Methods

PDDoc.AcquirePage

PDDoc.DeletePages

PDDoc.GetNumPages

PDDoc.InsertPages

PDDoc.MovePage

Save

```
VARIANT_BOOL Save(short nType, BSTR szFullPath);
```

Description

Saves a document.

Parameters

`nType`	Specifies the way in which the file should be saved. **NOTE:** A file may be saved as a linearized file using the **PDSaveLinearized** flag, but the following sequence MUST be observed: 1. Open the PDF file with **PDDoc**.**Open**. 2. Call **PDDoc**.**Save** using the **PDSaveLinearized** flag. 3. Call **PDDoc**.**Close**. This allows batch linearizing of files. **nType** is a logical OR of one or more of the following flags: • **PDSaveIncremental:** Write changes only, not the complete file. This will always result in a larger file, even if objects have been deleted. • **PDSaveFull:** Write the entire file to the filename specified by **szFullPath**. • **PDSaveCopy:** Write a copy of the file into the file specified by **szFullPath**, but keep using the old file. This flag can only be specified if **PDSaveFull** is also used. • **PDSaveCollectGarbage:** Remove unreferenced objects; this often reduces the file size, and its usage is encouraged. This flag can only be specified if **PDSaveFull** is also used. • **PDSaveLinearized:** Save the file in a linearized fashion, providing hint tables. This allows the PDF file to be byte-served. This flag can only be specified if **PDSaveFull** is also used.
`szFullPath`	The new pathname to the file, if any.

Return Value

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

Related Methods

PDDoc.**GetFileName**

SetFlags

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

Description

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

Parameters

nFlags	Flags to be set. See **PDDoc**.**GetFlags** for a description of the flags. The flags **PDDocWasRepaired**, **PDDocNewMajorVersion**, **PDDocNewMinorVersion**, **and PDDocOldVersion** are read-only and cannot be set.

Return Value

Always returns **-1**.

Related Methods

PDDoc.**ClearFlags**

PDDoc.**GetFlags**

SetInfo

```
VARIANT_BOOL SetInfo(BSTR szInfoKey, BSTR szBuffer);
```

Description

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

Parameters

szInfoKey	The key whose value is set.
szBuffer	The value to be assigned to the key.

Return Value

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

Related Methods

PDDoc.GetInfo

SetPageMode

```
VARIANT_BOOL SetPageMode(long nPageMode);
```

Description

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

Parameters

nPageMode	The page mode to be set. Must be one of the values listed in View Mode (see document in Chapter 1, "Apple Event Objects" for an explanation of the View Mode property).

Return Value

Always returns **-1**.

Related Methods

PDDoc.GetPageMode

PDDoc.SetPageMode

AcroExch.PDPage

The methods in this section work with pages.

AddAnnot

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

Description

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

Parameters

nIndexAddAfter	Location in the page's annotation array to add the annotation. The first annotation on a page has an index of zero.
iPDAnnot	The **LPDISPATCH** for the **AcroExch.PDAnnot** to add. **iPDAnnot** contains the instance variable **m_lpDispatch**, which contains the **LPDISPATCH**.

Return Value

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

Related Methods

PDPage.AddNewAnnot

PDPage.RemoveAnnot

AddNewAnnot

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

Description

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

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

Parameters

nIndexAddAfter	Location in the page's annotation array after which to add the annotation. The first annotation on a page has an index of zero.
szSubType	Subtype of the annotation to be created. Must be text.
iAcroRect	The **LPDISPATCH** for the **AcroExch.Rect** bounding the annotation's location on the page. **iAcroRect** contains the instance variable **m_lpDispatch**, which contains the **LPDISPATCH**.

Return Value

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

Related Methods

PDAnnot.SetContents

PDPage.AddAnnot

PDPage.RemoveAnnot

CopyToClipboard

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

Description

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

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

Parameters

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

Return Value

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

Related Methods

PDPage.DrawEx

CreatePageHilite

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

Description

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

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

Parameters

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

Return Value

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

Related Methods

AVDoc.ClearSelection

AVDoc.SetTextSelection

AVDoc.ShowTextSelect

HiliteList.Add

PDDoc.CreateTextSelect

PDPage.CreateWordHilite

PDTextSelect.Destroy

PDTextSelect.GetBoundingRect

PDTextSelect.GetNumText

PDTextSelect.GetPage

PDTextSelect.GetText

CreateWordHilite

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

Description

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

Parameters

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

Return Value

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

Related Methods

AVDoc.ClearSelection

AVDoc.SetTextSelection

AVDoc.ShowTextSelect

HiliteList.Add

PDDoc.CreateTextSelect

PDPage.CreatePageHilite

PDTextSelect.Destroy

PDTextSelect.GetBoundingRect

PDTextSelect.GetNumText

PDTextSelect.GetPage

PDTextSelect.GetText

CropPage

```
VARIANT_BOOL CropPage(LPDISPATCH iAcroRect);
```

Description

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

Parameters

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

Return Value

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

Related Methods

PDDoc.CropPages

Draw

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

Description

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

Parameters

window	**HWND** into which the page is to be drawn.
displayContext	**hDC** to use for drawing. If **NULL**, the **HDC** for **window** is used. **NOTE:** **displayContext** cannot be reliably used as the **hDC** for a printer device. In particular, Visual Basic applications cannot use **Draw** to print.
XOrigin	*x*–coordinate of the portion of the page to be drawn.
YOrigin	*y*–coordinate of the portion of the page to be drawn.
zoom	Zoom factor at which the page is to be drawn, specified as a percent (for example, 100 corresponds to a magnification of 1.0).

Return Value

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

Related Methods

PDPage.CopyToClipboard

PDPage.DrawEx

DrawEx

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

Description

Draws page contents into a specified window.

You can use **PDPage**.**CopyToClipboard** to copy page contents to the clipboard without an **hWnd** or **hDC** from the client.

Parameters

window	Handle for the window (**HWND**) into which the page is drawn.
displayContext	This parameter is invalid and should not be used. It should be assigned a **NULL** value. If it is not assigned a **NULL** value , an exception will be thrown. **NOTE:** **displayContext** cannot be reliably used as the **hDC** for a printer device. In particular, Visual Basic applications cannot use **DrawEx** to print.

updateRect	**LPDISPATCH** for an **AcroExch.Rect** to be drawn with *user space* coordinates. **updateRect** contains the instance variable **m_lpDispatch**, which contains the **LPDISPATCH**. Any objects outside of **updateRect** are not drawn. All objects are drawn if **updateRect** is **NULL**.
	NOTE: In previous specifications, this rectangle was in device space coordinates.
	Use methods in the **CAcroRect** class to set the size of the rectangle. For example:
	`CAcroRect* rect = new CAcroRect;`
	
	`rect->CreateDispatch("AcroExch.Rect", &e);`
	`if (rect) {`
	`/* Set values for rect - increases from right to left and bottom to top */`
	`rect->SetLeft(100);`
	`rect->SetTop(400);`
	`rect->SetRight(400);`
	`rect->SetBottom(100);`
	`}`

xOrigin	*x*–coordinate of the portion of the page to be drawn.
yOrigin	*y*–coordinate of the portion of the page to be drawn.
zoom	Zoom factor at which the page is drawn, specified as a percent (for example, 100 corresponds to a magnification of 1.0).

Return Value

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

Related Methods

PDPage.CopyToClipboard

GetAnnot

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

Description

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

Parameters

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

Return Value

The **LPDISPATCH** for the **AcroExch.PDAnnot** object.

Related Methods

PDPage.GetAnnotIndex

PDPage.GetNumAnnots

GetAnnotIndex

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

Description

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

Parameters

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

Return Value

The annotation's index.

Related Methods

PDPage.GetAnnot

PDPage.GetNumAnnots

GetDoc

```
LPDISPATCH GetDoc();
```

Description

Gets the **AcroExch.PDDoc** associated with the page.

Parameters

None

Return Value

The **LPDISPATCH** for the page's **AcroExch.PDDoc**.

Related Methods

AVPageView.GetPage

AVPageView.GetPageNum

PDDoc.AcquirePage

PDDoc.GetNumPages

PDPage.GetNumber

PDPage.GetRotate

PDPage.GetSize

PDTextSelect.GetPage

GetNumAnnots

```
long GetNumAnnots();
```

Description

Gets the number of annotations on the page.

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

Parameters

None

Return Value

The number of annotations on the page.

Related Methods

PDPage.GetAnnot

PDPage.GetAnnotIndex

GetNumber

```
long GetNumber();
```

Description

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

Parameters

None

Return Value

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

Related Methods

AVPageView.GetPage

AVPageView.GetPageNum

PDDoc.AcquirePage

PDDoc.GetNumPages

PDPage.GetDoc

PDPage.GetRotate

PDPage.GetSize

PDTextSelect.GetPage

GetRotate

```
short GetRotate();
```

Description

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

Parameters

None

Return Value

Rotation value. See **rotation** in page (located in Chapter 1, "Apple Event Objects") for a list of rotation values.

Related Methods

AVPageView.GetPage

AVPageView.GetPageNum

PDDoc.AcquirePage

PDPage.GetNumber

PDPage.GetSize

PDPage.SetRotate

PDTextSelect.GetPage

GetSize

```
LPDISPATCH GetSize();
```

Description

Gets a page's width and height in points.

Parameters

None

Return Value

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

Related Methods

AVPageView.GetPage

AVPageView.GetPageNum

PDDoc.AcquirePage

PDPage.GetNumber

PDPage.GetRotate

PDTextSelect.GetPage

RemoveAnnot

```
VARIANT_BOOL RemoveAnnot(long nIndex);
```

Description

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

Parameters

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

Return Value

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

Related Methods

PDPage.AddAnnot

PDPage.AddNewAnnot

PDPage.GetAnnotIndex

SetRotate

```
VARIANT_BOOL SetRotate(short nRotate);
```

Description

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

Parameters

nRotate	Rotation value. See **rotation** in page (in Chapter 1, "Apple Event Objects") for a list of rotation values.

Return Value

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

Related Methods

PDPage.GetRotate

AcroExch.PDTextSelect

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

Destroy

```
VARIANT_BOOL Destroy();
```

Description

Destroys a text selection object.

Parameters

None

Return Value

Always returns **-1**.

Related Methods

AVDoc.ClearSelection

AVDoc.SetTextSelection

AVDoc.ShowTextSelect

PDDoc.CreateTextSelect

PDPage.CreatePageHilite

PDPage.CreateWordHilite

PDTextSelect.GetBoundingRect

PDTextSelect.GetNumText

PDTextSelect.GetPage

PDTextSelect.GetText

GetBoundingRect

```
LPDISPATCH GetBoundingRect();
```

Description

Gets a text selection's bounding rectangle.

Parameters

None

Return Value

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

Related Methods

AVDoc.ClearSelection

AVDoc.SetTextSelection

AVDoc.ShowTextSelect

PDDoc.CreateTextSelect

PDPage.CreatePageHilite

PDPage.CreateWordHilite

PDTextSelect.Destroy

PDTextSelect.GetNumText

PDTextSelect.GetPage

PDTextSelect.GetText

GetNumText

```
long GetNumText();
```

Description

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

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

Parameters

None

Return Value

The number of elements in the text selection.

Related Methods

AVDoc.**ClearSelection**

AVDoc.**SetTextSelection**

AVDoc.**ShowTextSelect**

PDDoc.**CreateTextSelect**

PDPage.**CreatePageHilite**

PDPage.**CreateWordHilite**

PDTextSelect.**Destroy**

PDTextSelect.**GetBoundingRect**

PDTextSelect.**GetPage**

PDTextSelect.**GetText**

GetPage

```
long GetPage();
```

Description

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

Parameters

None

Return Value

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

Related Methods

AVDoc.ClearSelection

AVDoc.SetTextSelection

AVDoc.ShowTextSelect

AVPageView.GetPage

AVPageView.GetPageNum

PDDoc.CreateTextSelect

PDDoc.GetNumPages

PDPage.CreatePageHilite

PDPage.CreateWordHilite

PDPage.GetNumber

PDTextSelect.Destroy

PDTextSelect.GetBoundingRect

PDTextSelect.GetNumText

PDTextSelect.GetText

GetText

```
BSTR GetText(long nTextIndex);
```

Description

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

Parameters

nTextIndex	The element of the text selection to get.

Return Value

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

Related Methods

AVDoc.ClearSelection

AVDoc.SetTextSelection

AVDoc.ShowTextSelect

PDPage.CreatePageHilite

PDDoc.CreateTextSelect

PDPage.CreateWordHilite

PDTextSelect.Destroy

PDTextSelect.GetBoundingRect

PDTextSelect.GetNumText

PDTextSelect.GetPage

AxAcroPDFLib.AxAcroPDF

The methods in this section work with PDF browser controls.

GetVersions

```
VARIANT GetVersions();
```

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

GoBackwardStack

```
void GoBackwardStack();
```

Description

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

Parameters

None

Return Value

None.

Related Methods

AcroPDF.GoForwardStack

GoForwardStack

```
void GoForwardStack();
```

Description

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

Parameters

None

Return Value

None.

Related Methods

AcroPDF.GoBackwardStack

GotoFirstPage

```
void gotoFirstPage();
```

Description

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

Parameters

None

Return Value

None.

Related Methods

AcroPDF.GotoLastPage

AcroPDF.GotoNextPage

AcroPDF.GotoPreviousPage

AcroPDF.SetCurrentPage

GotoLastPage

```
void gotoLastPage();
```

Description

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

Parameters

None

Return Value

None.

Related Methods

AcroPDF.GotoFirstPage

AcroPDF.GotoNextPage

AcroPDF.GotoPreviousPage

AcroPDF.SetCurrentPage

GotoNextPage

```
void gotoNextPage();
```

Description

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

Parameters

None

Return Value

None.

Related Methods

AcroPDF.GotoFirstPage

AcroPDF.GotoLastPage

AcroPDF.GotoPreviousPage

AcroPDF.SetCurrentPage

GotoPreviousPage

```
void gotoPreviousPage();
```

Description

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

Parameters

None

Return Value

None.

Related Methods

AcroPDF.GotoFirstPage

AcroPDF.GotoLastPage

AcroPDF.GotoNextPage

AcroPDF.SetCurrentPage

LoadFile

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

Description

Opens and displays the specified document within the browser.

Parameters

fileName	The pathname of the file to be opened.

Return Value

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

Related Methods

None

Print

```
void Print();
```

Description

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

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

Parameters

None

Return Value

None.

Related Methods

AcroPDF.PrintAll

AcroPDF.PrintAllFit

AcroPDF.PrintPages

AcroPDF.PrintPagesFit

AcroPDF.PrintWithDialog

PrintAll

```
void printAll();
```

Description

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

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

Parameters

None

Return Value

None.

Related Methods

AcroPDF.Print

AcroPDF.PrintAllFit

AcroPDF.PrintPages

AcroPDF.PrintPagesFit

AcroPDF.PrintWithDialog

PrintAllFit

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

Description

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

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

Parameters

bOn	Determines whether to scale the imageable area when printing the document. A value of **0** indicates that no scaling should be used, and a positive value indicates that the pages are shrunk, if necessary, to fit into the imageable area of a page in the printer.

Return Value

None.

Related Methods

AcroPDF.Print

AcroPDF.PrintAll

AcroPDF.PrintPages

AcroPDF.PrintPagesFit

AcroPDF.PrintWithDialog

PrintPages

```
void printPages( Long nFrom, Long nTo);
```

Description

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

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

Parameters

nFrom	The page number of the first page to be printed. The first page in a document is page 0.
nTo	The page number of the last page to be printed.

Return Value

None.

Related Methods

AcroPDF.Print

AcroPDF.PrintAll

AcroPDF.PrintAllFit

AcroPDF.PrintPagesFit

AcroPDF.PrintWithDialog

PrintPagesFit

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

Description

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

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

Parameters

nFrom	The page number of the first page to be printed. The first page in a document is page 0.
nTo	The page number of the last page to be printed.
bShrinkToFit	Specifies whether the pages will be shrunk, if necessary, to fit into the imageable area of a page in the printer.

Return Value

None.

Related Methods

AcroPDF.Print

AcroPDF.PrintAll

AcroPDF.PrintAllFit

AcroPDF.PrintPages

AcroPDF.PrintWithDialog

PrintWithDialog

```
void printWithDialog();
```

Description

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

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

Parameters

None

Return Value

None.

Related Methods

AcroPDF.Print

AcroPDF.PrintAll

AcroPDF.PrintAllFit

AcroPDF.PrintPages

AcroPDF.PrintPagesFit

SetCurrentHighlight

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

Description

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

Parameters

nLeft	The distance in points from the left side of the page.
nTop	The distance in points from the top of the page.
nRight	The width of the bounding rectangle.
nBottom	The height of the bounding rectangle.

Return Value

None.

Related Methods

None

SetCurrentPage

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

Description

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

Parameters

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

Return Value

None.

Related Methods

AcroPDF.GotoFirstPage

AcroPDF.GotoLastPage

AcroPDF.GotoNextPage

AcroPDF.GotoPreviousPage

SetLayoutMode

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

Description

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

Parameters

szLayoutMode	Possible values: **"DontCare":** use the current user preference **"SinglePage":** use single page mode (as it would have appeared in pre-Acrobat 3.0 viewers) **"OneColumn":** use one-column continuous mode **"TwoColumnLeft":** use two-column continuous mode with the first page on the left **"TwoColumnRight":** use two-column continuous mode with the first page on the right

Return Value

None.

Related Methods

AcroPDF.SetNamedDest

AcroPDF.SetView

AcroPDF.SetViewRect

AcroPDF.SetViewScroll

SetNamedDest

```
void setNamedDest(BSTR szNamedDest);
```

Description

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

Parameters

szNamedDest	The named destination to which the viewer will go.

Return Value

None.

Related Methods

AcroPDF.SetLayoutMode

AcroPDF.SetView

AcroPDF.SetViewRect

AcroPDF.SetViewScroll

SetPageMode

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

Description

Sets the page mode according to the specified string.

Parameters

szPageMode	Possible values: **"none":** displays the document, but does not display bookmarks or thumbnails (default) **"bookmarks":** displays the document and bookmarks **"thumbs":** displays the document and thumbnails

Return Value

None.

Related Methods

AcroPDF.SetShowScrollbars

AcroPDF.SetShowToolbar

SetShowScrollbars

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

Description

Determines whether scrollbars will appear in the document view.

Parameters

bOn	A positive value indicates that scrollbars will appear, **0** indicates that they will not.

Return Value

None.

Related Methods

AcroPDF.SetPageMode

AcroPDF.SetShowToolbar

SetShowToolbar

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

Description

Determines whether a toolbar will appear in the viewer.

Parameters

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

Return Value

None.

Related Methods

AcroPDF.SetPageMode

AcroPDF.SetShowScrollbars

SetView

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

Description

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

Parameters

szViewMode	Possible values: **"Fit":** fits the entire page within the window both vertically and horizontally. **"FitH":** fits the entire width of the page within the window. **"FitV":** fits the entire height of the page within the window. **"FitB":** fits the bounding box within the window both vertically and horizontally. **"FitBH":** fits the entire width of the bounding box within the window. **"FitBV":** fits the entire height of the bounding box within the window.

Return Value

None.

Related Methods

AcroPDF.SetLayoutMode

AcroPDF.SetNamedDest

AcroPDF.SetViewRect

AcroPDF.SetViewScroll

SetViewRect

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

Description

Sets the view rectangle according to the specified coordinates.

Parameters

left	The upper left horizontal coordinate.
top	The vertical coordinate in the upper left corner.
width	The horizontal width of the rectangle.
height	The vertical height of the rectangle.

Return Value

None.

Related Methods

AcroPDF.SetLayoutMode

AcroPDF.SetNamedDest

AcroPDF.SetView

AcroPDF.SetViewScroll

SetViewScroll

```
void setViewRect(BSTR szViewMode, FLOAT offset);
```

Description

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

Parameters

szViewMode	Possible values: **"Fit":** fits the entire page within the window both vertically and horizontally. **"FitH":** fits the entire width of the page within the window. **"FitV":** fits the entire height of the page within the window. **"FitB":** fits the bounding box within the window both vertically and horizontally. **"FitBH":** fits the entire width of the bounding box within the window. **"FitBV":** fits the entire height of the bounding box within the window.
offset	The horizontal or vertical coordinate positioned either at the left or top edge.

Return Value

None.

Related Methods

AcroPDF.SetLayoutMode

AcroPDF.SetNamedDest

AcroPDF.SetView

AcroPDF.SetViewRect

SetZoom

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

Description

Sets the magnification according to the specified value.

Parameters

`percent`	The desired zoom factor, expressed as a percentage (for example, 1.0 represents a magnification of 100%).

Return Value

None.

Related Methods

`AcroPDF.``SetZoomScroll`

SetZoomScroll

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

Description

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

Parameters

percent	The desired zoom factor, expressed as a percentage (for example, 1.0 represents a magnification of 100%).
left	The horizontal coordinate positioned at the left edge.
top	The vertical coordinate positioned at the top edge.

Return Value

None.

Related Methods

AcroPDF.SetZoom

5 OLE Automation Properties

AcroExch.Point

X

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

Description

Gets or sets the *x*-coordinate of an **`AcroPoint`**.

Return Value

The *x*-coordinate of the **`AcroPoint`**.

Y

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

Description

Gets or sets the *y*-coordinate of an **`AcroPoint`**.

Return Value

The *y*-coordinate of the **`AcroPoint`**.

AcroExch.Rect

Bottom

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

Description

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

Return Value

The *y*-coordinate of the bottom of the **`AcroRect`**.

Left

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

Description

Gets or sets left *x*-coordinate of an **`AcroRect`**.

Return Value

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

Right

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

Description

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

Return Value

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

Top

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

Description

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

Return Value

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

AcroExch.Time

Date

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

Description

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

Return Value

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

Hour

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

Description

Gets or sets the hour from an **`AcroTime`**.

Return Value

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

Millisecond

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

Description

Gets or sets the milliseconds from an **`AcroTime`**.

Return Value

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

Minute

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

Description

Gets or sets the minutes from an **`AcroTime`**.

Return Value

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

Month

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

Description

Gets or sets the month from an **`AcroTime`**.

Return Value

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

Second

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

Description

Gets or sets the seconds from an **`AcroTime`**.

Return Value

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

Year

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

Description

Gets or sets the year from an **`AcroTime`**.

Return Value

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

AxAcroPDFLib.AxAcroPDF

Src

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

Description

Gets or sets the URL for the document.

Return Value

The URL for the document, formatted as a string.

DDE

This reference contains the following section:

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

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

8 DDE Messages

AppExit

```
[AppExit()]
```

Description

Exits the Acrobat application.

`AppExit` is also supported in Adobe Reader.

Parameters

None

Return Value

`true` if the Acrobat application exited successfully, **`false`** otherwise.

Related Methods

`AppHide`

`AppShow`

AppHide

[AppHide()]

Description

Iconifies or hides the Acrobat application.

Parameters

None

Return Value

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

Related Methods

AppExit

AppShow

AppShow

[AppShow ( )]

Description

Shows the Acrobat application.

Parameters

None

Return Value

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

Related Methods

AppExit

AppHide

CloseAllDocs

```
[CloseAllDocs()]
```

Description

Closes all open documents.

CloseAllDocs is also supported in Adobe Reader.

Parameters

None

Return Value

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

Related Methods

DocClose

DocOpen

FileOpen

DocClose

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

Description

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

DocClose is also supported in Adobe Reader.

Parameters

fullPath	The full pathname of the file to be closed.

Return Value

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

Related Methods

CloseAllDocs

DocOpen

FileOpen

DocDeletePages

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

Description

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

Parameters

fullPath	The full pathname of the document.
fromPage	The page number of the first page to be deleted.
toPage	The page number of the last page to be deleted.

Return Value

true if the pages were deleted successfully. Returns **false** if the document specified by **fullPath** does not exist, if the request was to delete all the document's pages, or if the pages were not deleted successfully.

Related Methods

DocInsertPages

DocReplacePages

DocFind

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

Description

Finds a string in a specified file. This does not use the cross-document search present in versions 2.0 and later of Acrobat, but performs a page-by-page search of the specified file.

Parameters

fullPath	The full pathname of the file to be searched.
string	The string to be found.
caseSensitive	**true** if the search is case-sensitive, **false** otherwise.
wholeWords	**true** if the search will only match whole words, **false** otherwise.
bReset	**true** if the search begins on the first page of the document, **false** if the search begins on the current page.

Return Value

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

DocGoTo

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

Description

Goes to the specified page.

DocGoTo is also supported in Adobe Reader.

Parameters

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.

DocGoToNameDest

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

Description

Goes to the specified named destination.

DocGoToNameDest is also supported in Adobe Reader.

Parameters

fullPath	The full pathname of the file.
nameDest	The named destination.

Return Value

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

DocInsertPages

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

Description

Inserts pages from one file into another.

Parameters

fullPath	The full pathname of the target document, which must already be open in the Acrobat application.
insertAfterPage	The page number after which pages are being inserted. Possible values may be a page number, or one of the following enumerations: **PDBeforeFirstPage** — Pages are inserted at the beginning of the document. **PDLastPage** — Pages are inserted at the end of the document.
sourcePath	The full pathname of the source document. This file need not be open in the Acrobat application.

Return Value

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

Related Methods

DocDeletePages

DocReplacePages

DocOpen

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

Description

Opens a document and adds it to the list of documents known to DDE, allowing it to be manipulated by other DDE messages (see **FileOpen**).

DocOpen is also supported in Adobe Reader.

Parameters

fullPath	The full pathname of the file to be opened.

Return Value

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

Related Methods

CloseAllDocs

DocClose

FileOpen

DocPageDown

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

Description

Scrolls forward through the document by one screen area.

Parameters

fullPath	The full pathname of the document.

Return Value

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

Related Methods

DocPageLeft

DocPageRight

DocPageUp

DocScrollTo

DocPageLeft

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

Description

Scrolls to the left by a small amount.

Parameters

fullPath	The full pathname of the document.

Return Value

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

Related Methods

DocPageDown

DocPageRight

DocPageUp

DocScrollTo

DocPageRight

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

Description

Scrolls to the right by a small amount.

Parameters

fullPath	The full pathname of the document.

Return Value

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

Related Methods

DocPageDown

DocPageLeft

DocPageUp

DocScrollTo

DocPageUp

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

Description

Scrolls backward through the document by one screen area.

Parameters

fullPath	The full pathname of the document.

Return Value

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

Related Methods

DocPageDown

DocPageLeft

DocPageRight

DocScrollTo

DocPrint

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

Description

Prints a specified range of pages from a document, without displaying any modal Print dialog box to the user. For PostScript printing, only Level 1 operators are used, only ASCII data is generated, and the document's pages are not shrunk to fit into the imageable area of the printed page.

Parameters

fullPath	The full pathname of document.
startPage	The page number of the first page to be printed.
endPage	The page number of the last page to be printed.

Return Value

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

Related Methods

FilePrint

FilePrintSilent

FilePrintTo

DocReplacePages

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

Description

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

Parameters

fullPath	The full pathname of the target document. This file must already be open in the Acrobat application.
startDestPage	The page number of the first page in the target document to be replaced.
sourcePath	The full pathname of the source document. This file does not have to be already open in the Acrobat application.
startSourcePage	The page number of the first page in the source document to use as a replacement page.
endSourcePage	The page number of the last page in the source document 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.

Related Methods

DocDeletePages

DocInsertPages

DocSave

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

Description

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

Parameters

fullPath	The full pathname of the file to be saved.

Return Value

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

Related Methods

DocSaveAs

DocSaveAs

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

Description

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

Parameters

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, **false** if the document does not exist or was not saved successfully.

Related Methods

DocSave

DocScrollTo

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

Description

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

Parameters

fullPath	The full pathname of the document.
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.

Related Methods

DocPageDown

DocPageLeft

DocPageRight

DocPageUp

DocSetViewMode

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

Description

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

Parameters

fullPath	The full pathname of the document.
viewType	The view mode to be used. 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, **false** if the document specified by **fullPath** does not exist or an unknown view mode is specified.

Related Methods

FullMenus

ShortMenus

DocZoomTo

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

Description

Sets the zoom for a specified document.

Parameters

fullPath	The full pathname of the file whose zoom to 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 specified as a percent (for example, 100 corresponds to a magnification of 1.0). **scale** is used only when **zoomType** is **AVZoomNoVary**.

Return Value

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

FileOpen

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

Description

Opens and displays the specified document. If the file is already open, it becomes the active document and appears in the front. This DDE message does not add the document to the list that can be manipulated using DDE messages; use **DocOpen** to do that.

FileOpen is also supported in Adobe Reader.

Parameters

fullPath	The full pathname of the file to be opened.

Return Value

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

Related Methods

CloseAllDocs

DocClose

DocOpen

FileOpenEx

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

Description

Opens and displays a file. If the file is already open, it becomes the active document and appears in the front. This DDE message does not add the document to the list that can be manipulated using DDE messages; use **DocOpen** to do that.

This method allows documents that either take a long time to open or are password-protected to open without stopping the flow of DDE messages. Documents opened with **FileOpenEx** are opened during an idle period. This is useful in situations in which several DDE messages are sent at once, such as a multiple file select from Windows Explorer.

FileOpenEx is also supported in Adobe Reader.

Parameters

fullPath	The full pathname of the file to be opened.

Return Value

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

Related Methods

FileOpen

CloseAllDocs

DocClose

DocOpen

FilePrint

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

Description

Prints all pages in a document, displaying a modal print dialog box to the user. For PostScript printing, only Level 1 operators are used, only ASCII data is generated, and the document's pages are not shrunk to fit into the imageable area of the printed page.

FilePrint is also supported in Adobe Reader.

Parameters

fullPath	The full pathname of the file to be printed.

Return Value

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

Related Methods

DocPrint

FilePrintSilent

FilePrintTo

FilePrintEx

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

Description

Prints all pages in a document, displaying a modal print dialog box to the user. For PostScript printing, only Level 1 operators are used, only ASCII data is generated, and the document's pages are not shrunk to fit into the imageable area of the printed page.

Similar to **FileOpenEx**, this is a special DDE command that returns **true** right away and performs the action during idle periods. This ensures that no DDE commands are dropped when printing a large number of files simultaneously.

FilePrintEx is also supported in Adobe Reader.

Parameters

fullPath	The full pathname of the file to print.

Return Value

true is always returned.

Related Methods

DocPrint

FileOpenEx

FilePrint

FilePrintSilent

FilePrintSilentEx

FilePrintTo

FilePrintToEx

FilePrintSilent

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

Description

Prints all pages in a document, without displaying a print dialog box to the user. For PostScript printing, only Level 1 operators are used, only ASCII data is generated, and the document's pages are not shrunk to fit into the imageable area of the printed page.

FilePrintSilent is also supported in Adobe Reader.

Parameters

fullPath	The full pathname of the file to be printed.

Return Value

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

Related Methods

DocPrint

FilePrint

FilePrintTo

FilePrintSilentEx

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

Description

Prints all pages in a document, without displaying a print dialog box to the user. For PostScript printing, only Level 1 operators are used, only ASCII data is generated, and the document's pages are not shrunk to fit into the imageable area of the printed page.

Similar to **FileOpenEx**, this is a special DDE command that returns **true** right away and does the action during idle periods. This is to ensure that no DDE commands are dropped when printing a large number of files simultaneously.

FilePrintSilentEx is also supported in Adobe Reader.

Parameters

fullPath	The full pathname of the file to be printed.

Return Value

true is always returned.

Related Methods

DocPrint

FileOpenEx

FilePrintEx

FilePrintSilent

FilePrintTo

FilePrintToEx

FilePrintTo

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

Description

Prints all pages in a document to a specified printer, using a specified driver and port, displaying a modal print dialog box to the user. For PostScript printing, only ASCII data is generated, and the document's pages are not shrunk to fit into the imageable area of the printed page.

FilePrintTo is also supported in Adobe Reader.

Parameters

fullPath	The full pathname of the file to be printed.
printName	(*Required for Windows 95 and higher*) The name of the printer.
driverName	Printer driver name.
portName	(*Required for Windows NT*) Port name.

Return Value

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

Related Methods

DocPrint

FilePrint

FilePrintSilent

FilePrintToEx

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

Description

Prints all pages in a document to a specified printer, using a specified driver and port, displaying a modal print dialog box to the user. For PostScript printing, only ASCII data is generated, and the document's pages are not shrunk to fit into the imageable area of the printed page.

Similar to **FileOpenEx**, this is a special DDE command that returns **true** right away and does the action during idle periods. This is to ensure that no DDE commands are dropped when printing a large number of files simultaneously.

FilePrintToEx is also supported in Adobe Reader.

Parameters

fullPath	The full pathname of the file to be printed.
printName	(*Required for Windows 95 and higher*) The name of the printer.
driverName	Printer driver name.
portName	(*Required for Windows NT*) Port name.

Return Value

true is always returned.

Related Methods

DocPrint

FileOpenEx

FilePrintEx

FilePrintSilentEx

FilePrintTo

FilePrintToEx

FullMenus

```
[FullMenus()]
```

Description

Displays full menus, and sets this option in the Acrobat application's preferences file.

With Acrobat 3.0 or later, all menus are displayed, and this function is ignored.

Parameters

None

Return Value

`true` if full menus were set successfully, **`false`** otherwise.

Related Methods

`DocSetViewMode`

`ShortMenus`

HideToolbar

```
[HideToolbar()]
```

Description

Hides the toolbar.

Parameters

None

Return Value

`true` if the toolbar was hidden successfully, **`false`** otherwise.

Related Methods

`ShowToolbar`

MenuitemExecute

```
[MenuitemExecute(char* menuItemName)]
```

Description

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

Parameters

menuItemName	The language-independent name of the menu item to execute. See the *Acrobat And PDF Library API Reference* for a list of menu item names.

Return Value

None

ShortMenus

[ShortMenus()]

Description

Displays short menus, and sets this option in the Acrobat application's preferences file. With Acrobat 3.0 or later, all menus are displayed, and this function is ignored.

Parameters

None

Return Value

true if short menus were set successfully, **false** otherwise.

Related Methods

DocSetViewMode

FullMenus

ShowToolbar

[ShowToolbar()]

Description

Shows the toolbar.

Parameters

None

Return Value

true if the toolbar was shown successfully, **false** otherwise.

Related Methods

HideToolbar

Plug-Ins

This portion of the reference contains descriptions of the **Catalog**, **AcroForm**, and **Search** plug-ins.

7 Acrobat Catalog

Acrobat Catalog is a plug-in that allows you to create a full-text index of a set of PDF documents. A full-text index is a searchable database of all the text in the documents. After building an index, you can use the **Edit > Search** command to search the entire library quickly. Searches of full-text indexes created using Catalog are faster and more convenient than using the **Edit > Find** command.

For more information, see the *Acrobat and PDF Library API Overview*.

Contents

This part of the document describes the methods available when using DDE.

Catalog Windows Messages

Catalog broadcasts a set of Windows Messages when certain operations occur. These messages are broadcast whether the operations are initiated from the user interface, HFT methods, or DDE methods.

- **`AcrobatCatalogBuildSuccess`**: On every successful build.
- **`AcrobatCatalogBuildFail`**: On every failed build.
- **`AcrobatCatalogBuildStopped`**: When a build has stopped.

8 Catalog DDE Methods

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

AppExit

```
[AppExit()]
```

Description

Exits Acrobat Catalog.

Parameters

None.

Return Value

If **`true`**, Catalog exited successfully, otherwise **`false`**.

AppFront

```
[AppExit()]
```

Description

Brings Catalog to the front.

Parameters

None.

Return Value

None.

FileBuild

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

Description

Builds an index based on the specified index definition file.

Parameters

fullPath	The full pathname of the file to be opened (including the **.pdx** extension).

Return Value

If **true**, the file opened successfully, otherwise **false**.

FileOpen

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

Description

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

Parameters

`fullPath`	The full pathname of the file to be opened (including the **`.pdx`** extension).

Return Value

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

FilePurge

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

Description

Purges an index definition file.

Parameters

`fullPath`	The full pathname of the file to be purged (including the **`.pdx`** extension).

Return Value

`true` if the file was successfully purged, otherwise **`false`**.

9 Acrobat Forms

The Acrobat Forms plug-in allows a Portable Document Format (PDF) document to act as a form; that is, the Acrobat equivalent of a paper form with fields. It is now faster and easier to exchange information either in familiar paper or electronic forms converted to PDF files with Acrobat, or as dynamic interactive database templates.

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

The Forms plug-in for Acrobat (versions 4.0 and above) allows users to author form fields. For Adobe Reader, the Forms plug-in does not allow form authoring, but allows users to fill in data and print Acrobat forms. The Reader Forms plug-in also does not allow users to save data to the local hard disk. Both Acrobat and Reader allow Web designers to send data from the form back to a Web server.

Contents

This section of the document is a reference for developers who want to take advantage of the Forms API. It describes the OLE Automation methods exported by the Acrobat AcroForm plug-in.

Other Useful Documentation

You should be familiar with the Acrobat core API. For more information, see the following documents:

- *Acrobat and PDF Library API Overview*
- *Acrobat and PDF Library API Reference*
- *Acrobat SDK Plug-in Guide*
- *PDF Reference*

10 Acroform OLE Automation

Description

The Acrobat Forms plug-in has been enhanced to work as an Automation server in the Win32 environment. Since the automation capabilities have been added to a plug-in, rather than an executable that can be directly launched, the following steps are necessary to access them from an Automation controller:

First instantiate the Acrobat application by using the VB **`CreateObject`** method. For example:

```
CreateObject("AcroExch.App")
```

This causes the Acrobat Forms plug-in to run, at which time it registers its class object with OLE. Then its main exposed object can be instantiated, that is:

```
CreateObject("AFormAut.App")
```

Presently, registration in the Windows registry (which is different from the class object registration described above) happens every time Acrobat loads the plug-in. Therefore, you must run Acrobat at least once with the `AForm32.api` file in the `plug-ins` folder before its type library can be found for object browsing within the Microsoft Visual Studio environment. This is also necessary in order to allow early binding. Declare the program variables as objects of the corresponding classes in **`AFORMAUTLib`**, and not simply as **`Object`**.

The object model was designed in accordance with the applicable standards and guidelines for document-centric applications from the *OLE Programmer's Reference*. That manual uses the term *document* to describe whatever an application uses as a file or an individual entity of data (in our case a field).

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

Contents

This portion of the document contains the following sections:

- Acroform OLE Automation Objects. This section describes the Acrobat objects represented as OLE objects.
- Acroform OLE Automation Methods. This section describes each OLE method, including its parameters, return value, and related methods.
- Acroform Automation Properties. This section details the properties that can be set in the various objects. Each property describes the key, the property type (for example, read-only), and the semantics.

Conventions

The syntax used in this reference follows that used in Microsoft Visual Basic references.

Exceptions

All methods and properties may return an exception. These may include standard OLE exceptions, such as:

- **`E_OUTOFMEMORY`** (0x8007000E)
- **`E_INVALIDARG`** (0x80070057)

These exceptions are not specifically listed in the descriptions of the methods and properties that appear below. Others are AcroForm-specific, and are listed in the following table.

The actual numeric value of the returned exception is assembled as an **`HRESULT`**, uses the **`FACILITY_ITF`**, and starts with decimal 512 (hex 0x0200), as recommended by Microsoft. For example, the numeric value of the exception **`AutErcNoForm`** is 0x80040201. The important part is the right-most (0x201), which is the first error in the enumeration below.

Exception Name	Numeric Value	Description
`AutErcNoDoc`	1	No document is currently open in the Acrobat application.
`AutErcNotTerminal`	2	This property or method applies to terminal fields or their annotations.
`AutErcNotToThisFieldType`	3	This property or method is not applicable to this type of field.

11 Acroform OLE Automation Objects

AFormApp

AFormApp is the only object the controller can externally instantiate (that is, using **CreateObject**). All other objects must be created by navigating down the hierarchy with the methods and properties described in this section.

Field

A field in the document that is currently active in Acrobat.

Fields

A collection of all the fields in the document that are currently active in Acrobat at the time **Fields** is instantiated.

The Fields collection includes both *terminal* and *non-terminal* fields. A terminal field is one that either does not have children, or if it does, they are simply multiple appearances (that is, child annotations) of the field in question.

NOTE: If you instantiate a **Fields** object, and subsequently fields are manually added or removed using the Forms tool in Acrobat, the **Fields** object will no longer be in sync with the document. You must re-instantiate the **Fields** object.

12 Acroform OLE Automation Methods

Field

PopulateListOrComboBox

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

Description

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

Parameters

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

Return Value

None

Exceptions

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

Related Methods

Add

SetBackgroundColor

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

Description

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

Parameters

bstrColorSpace	Values are defined by using a **transparent**, **gray**, **RGB** or **CMYK** color space. Valid strings include: • **"T"** • **"G"** • **"RGB"** • **"CMYK"**
GorRorC	Used if **bstrColorSpace** is set to **T**, **G**, or **RGB**. A float range between zero and one inclusive.
GorM	Used if **bstrColorSpace** is set to **G**. A float range between zero and one inclusive.
BorY	Used if **bstrColorSpace** is set to **RGB**. A float range between zero and one inclusive.
K	Used if **bstrColorSpace** is set to **CMYK**. A float range between zero and one inclusive.

Return Value

None

Related Methods

SetBorderColor
SetForegroundColor

Example

```
Field.SetBackgroundColor "RGB", 0.7, 0.3, 0.6, 0
```

SetBorderColor

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

Description

Specifies the border color for a field. The border color is used to stroke the field's rectangle with a line as large as the border width. The new border color is propagated to any child annotations underneath, so the field may be non-terminal.

Parameters

bstrColorSpace	A string specifying the color space. Valid strings include: • **"T"** • **"G"** • **"RGB"** • **"CMYK"**
GorRorC	Used if **bstrColorSpace** is set to **T**, **G**, or **RGB**. A float range between zero and one inclusive.
GorM	Used if **bstrColorSpace** is set to **G**. A float range between zero and one inclusive.
BorY	Used if **bstrColorSpace** is set to **RGB**. A float range between zero and one inclusive.
K	Used if **bstrColorSpace** is set to **CMYK**. A float range between zero and one inclusive.

Return Value

None

Related Methods

SetBackgroundColor
SetForegroundColor

Example

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

SetButtonCaption

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

Description

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

Parameters

bstrFace	A string that specifies the face for which the caption will be used. Valid strings include: • **N** is the Normal appearance • **D** is the Down appearance • **R** is the appearance for Rollover
bstrCaption	The caption for the button. **NOTE:** If a button's layout is of type **icon** only, the caption is not used in generating its appearance. In addition, only the **Normal** face is displayed, unless the **Highlight** is of type **push**.

Return Value

None

Exceptions

Raises **AutErcNotToThisFieldType** if the field is not of type **button**. The new appearance is propagated to any child annotations underneath; the field may be non-terminal.

Related Methods

SetButtonIcon

Example

```
Field.SetButtonCaption "D", "Submit Form"
```

SetButtonIcon

```
void SetButtonIcon (LPCTSTR bstrFace, LPCTSTR bstrFullPath,
short pageNum);
```

Description

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

Parameters

bstrFace	A string that specifies the face for which the icon will be used. Valid strings include: • **N** is the Normal appearance • **D** is the Down appearance • **R** is the appearance for Rollover
bstrFullPath	The full pathname of the PDF file to be used as the source of the appearance.
pageNum	Used to select the page inside that PDF file (zero-based). NOTE: If a button's layout is of type **icon** only, the caption is not used in generating its appearance. In addition, only the **Normal** face is displayed, unless the **Highlight** is of type **push**.

Return Value

None

Exceptions

Raises **AutErcNotToThisFieldType** if the field is not of type **button**. The new appearance is propagated to any child annotations underneath, so it is OK if the field is non-terminal.

Related Methods

SetButtonCaption

Example

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

SetExportValues

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

Description

Sets the export values for each of the annotations of a field of type radio button and checkbox.

For radio button fields, this is necessary to make the field work properly as a group. One button is checked at any given time, giving its value to the field as a whole.

For checkbox fields, unless an export value is specified, the default is used when the field checked is **Yes**. When it is unchecked, its value is **Off** (this is also true for a radio button field when none of its buttons are checked).

Parameters

arrExportVal	An array of strings, which is expected to have as many elements as there are annotations in the field. The elements of the array are distributed among the individual annotations comprising the field, using their tab order.

Return Value

None

Exceptions

Raises **AutErcNotToThisFieldType** if the field is not of type radio button or checkbox.

Related Methods

Add

Example

```
Dim arrExp(1) As String
arrExp(0) = "Visa"
arrExp(1) = "Mastercard"
Field.SetExportValues arrExp
```

SetForegroundColor

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

Description

Specifies the foreground color for a field. It represents the text color for text, button, combobox, or listbox fields and the check color for checkbox or radio button fields.

NOTE: Parameters are similar to **SetBorderColor** and **SetBackgroundColor**, except that the **transparent** color space is not allowed.

Parameters

bstrColorSpace	A string specifying the color space. Valid strings include: • **"T"** • **"G"** • **"RGB"** • **"CMYK"**
GorRorC	Used if **bstrColorSpace** is set to **T**, **G**, or **RGB**. A float range between zero and one inclusive.
GorM	Used if **bstrColorSpace** is set to **G**. A float range between zero and one inclusive.
BorY	Used if **bstrColorSpace** is set to **RGB**. A float range between zero and one inclusive.
K	Used if **bstrColorSpace** is set to **CMYK**. A float range between zero and one inclusive.

Return Value

None

Related Methods

SetBackgroundColor
SetBorderColor

Example

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

SetJavaScriptAction

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

Description

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

Parameters

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

bstrTheScript	The script itself. **NOTE:** If the trigger is **calculate**, an entry is added at the end of the calculation order array (see the **CalcOrderIndex** property).

Return Value

None

Related Methods

None

Calculate Notes

There is a simple calculate script supplied with Acrobat.

`AFSimple_Calculate(`*`cFunction, cFields`*`)`

- *cFunction* is one of **"AVG"**, **"SUM"**, **"PRD"**, **"MIN"**, **"MAX"**
- *cFields* is the list of the fields to use in the calculation.

Formatting Notes

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

`AFDate_KeystrokeEx(cFormat)`
`AFDate_Format(cFormat)`

– *`cFormat`* is one of:

"m/d", "m/d/yy", "mm/dd/yy", "mm/yy", "d-mmm", "d-mmm-yy", "dd-mmm-yy", "yy-mm-dd", "mmm-yy", "mmmm-yy", "mmm d, yyyy", "mmmm d, yyyy", "m/d/yy h:MM tt", "m/d/yy HH:MM"

`AFTime_Keystroke(ptf)`
`AFTime_Format(ptf)`

– *`ptf`* is the time format:

0 = 24HR_MM [14:30]
1 = 12HR_MM [2:30 PM]
2 = 24HR_MM_SS [14:30:15]
3 = 12HR_MM_SS [2:30:15 PM]

`AFPercent_Keystroke(nDec, sepStyle)`
`AFPercent_Format(nDec, sepStyle)`

– *`nDec`* is the number of places after the decimal point.
– *`sepStyle`* is an integer denoting whether to use a separator. If *`sepStyle`* is **0**, use commas. If *`sepStyle`* is **1**, do not separate.

`AFSpecial_Keystroke(psf)`
`AFSpecial_Format(psf)`

– *`psf`* is the type of formatting to use:

0 = zip code
1 = zip + 4
2 = phone
3 = SSN

`AFNumber_Format(nDec, sepStyle, negStyle, currStyle, strCurrency, bCurrencyPrepend)`
`AFNumber_Keystroke(nDec, sepStyle, negStyle, currStyle, strCurrency, bCurrencyPrepend)`

- *nDec* is the number of places after the decimal point.
- *sepStyle* is an integer denoting whether to use a separator. If *sepStyle* is **0**, use commas. If *sepStyle* is **1**, do not separate.
- *sepStyle* is the formatting used for negative numbers:

  0 = MinusBlack
  1 = Red
  2 = ParensBlack
  3 = ParensRed
- *currStyle* is the currency style - not used.
- *strCurrency* is the currency symbol.
- *bCurrencyPrepend* is **true** to prepend the currency symbol; **false** to display on the end of the number.

SetResetFormAction

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

Description

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

Parameters

`bstrTrigger`	A string that specifies which trigger is used for the action. Valid strings include: • **`"up"`** (Mouse Up) • **`"down"`** (Mouse Down) • **`"enter"`** (Mouse Enter) • **`"exit"`** (Mouse Exit)
`theFlags`	When **0** (**`Include`**), **`arrFields`** specifies which fields to *include* in the reset operation. When non-zero (**`Exclude`**), **`arrFields`** specifies which fields to *exclude* from the reset operation.
`arrFields`	(*Optional*) An array of strings for the fully-qualified names of the fields. Depending on the value of **`theFlags`**, these fields are included in or excluded from the reset operation. When the fields are included, the set can include the names of non-terminal fields, which is a fast and easy way to cause all their children to be included in the action. When not supplied, all fields are reset.

Return Value

None

Related Methods

None

SetSubmitFormAction

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

Description

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

Parameters

bstrTrigger	A string that specifies which trigger is used for the action. Valid strings include: • **"up"** (Mouse Up) • **"down"** (Mouse Down) • **"enter"** (Mouse Enter) • **"exit"** (Mouse Exit)
bstrTheURL	A string containing the URL.
theFlags	A collection of flags that define various characteristics of the action. **NOTE:** See the *PDF Reference* to learn how the binary value of this **long** is interpreted.
arrFields	(*Optional*) If passed, represents an array of strings for the fully-qualified names of the fields to submit when the action is executed. If the array is interpreted as fields to submit (as opposed to fields excluded from the submission, depending on the least-significant bit in the flags), then it may include the names of non-terminal fields, which is a fast and easy way to cause all their children to be included in the submission. If not passed, then the created action will not include a **/Fields** key.

Return Value

None

Related Methods

None

Fields

Add

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

Description

Dynamically adds a new field to the Acrobat form and to the **Fields** collection.

Returns the newly-created **Field** object. You can pass the name of an existing field as a parameter, as long as that field is of the same type as the one being created.

This is useful in the following circumstances:

- For radio buttons to use the **SetExportValues** method to make the radio buttons mutually exclusive.
- For fields that should have multiple appearances (that is, child annotations) in the document.

Parameters

bstrFieldName	The fully-qualified name of the field.
bstrFieldType	Field type for the newly created field. Valid types are: • **"text"** • **"button"** • **"combobox"** • **"listbox"** • **"checkbox"** • **"radio button"** • **"signature"** You must use the quotation marks. See the sample code below. **NOTE:** When creating list or combo boxes, there is a limit of 64K for string data on Windows platforms. MacOS systems have a limit of 200 entries for the list or combo boxes. Using more than the limit degrades performance. You populate the fields of the list and combo boxes using the **PopulateListOrComboBox** method.
pageNum	The page number (zero-based).

`left, top, right, bottom`	These parameters are floats representing the left, top, right, and bottom coordinates of the field rectangle, measured in *rotated page space*; that is, [0,0] is always at the left bottom corner regardless of page rotation.

Return Value

The newly-created **`Field`** object.

Related Methods

`PopulateListOrComboBox`
`Remove`

Example

```
Set Field = Fields.Add("payment”,_ "radiobutton”, 0, 100, 600, 130, 570)
```

AddDocJavascript

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

Description

Adds a document-level JavaScript function to the PDF file. When using **AddDocJavascript**, within Visual Basic, you can use **Chr(13)** to add a <CR>, and **Chr(9)** for tabs, so that the function is nicely formatted.

Parameters

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

Return Value

None

Related Methods

ExecuteThisJavascript

Example

```
'Adding a document-level JavaScript
'function, to compute factorials:
Fields.AddDocJavaScript "Fact", _
"function Fact(n)" & Chr(13) & _
"{" & Chr(13) & _
Chr(9) & "if (n <= 0)" & Chr(13) & _
Chr(9) & Chr(9) & "return 1;" & Chr(13) & _
Chr(9) & "else" & Chr(13) & _
Chr(9) & Chr(9) & "return n * Fact(n - 1);" & Chr(13) & _
"}"
```

ExecuteThisJavascript

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

Description

Executes the specified JavaScript script.

Parameters

bstrTheScript	A string containing a JavaScript script, which is executed by Acrobat in the context of the currently active document. **NOTE:** See the *Acrobat JavaScript Scripting Reference* for information on event level values.

Return Value

Returns a result by assigning it to event value.

Related Methods

AddDocJavascript

Example

```
Fields.ExecuteThisJavaScript "var f =_ this.getField(""myButton"");
f.delay =_ false;"
```

To get the return value in Visual Basic:

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

ExportAsFDF

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

Description

Exports the data as FDF from an Acrobat form.

Parameters

bstrFullPath	A full pathname of the file to which the produced FDF file will be saved.
bstrSubmitButton	The name of an existing form field of type **button** (in case you want to include it in the FDF file, as if it had been used to trigger a **SubmitForm** action). You may pass an empty string.
bEmptyFields	A boolean to indicate whether fields with no value should be included in the produced FDF file.
arrFields	(*Optional*) An array of strings representing the fully-qualified names of the fields to include in the FDF file. This array may include the names of non-terminal fields, which is a fast and easy way to cause all their children to be included in the FDF file.

Return Value

None

Related Methods

ImportAnFDF
ExportAsHtml

Example

```
Dim arrFields(1) As String
arrFields(0) = "name”
arrFields(1) = "address”
‘This will create an FDF that includes
‘name.last, name.first, address.street,
‘etc., but only if they have a value
‘(since we are passing False for the
‘ "bEmptyFields” parameter.
Fields.ExportAsFDF "C:\Temp\out.fdf”, "”, False, arrFields
```

ExportAsHtml

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

Description

Exports the data as HTML from an Acrobat form. This method is similar to **ExportAsFDF**. The only difference is that the form data is exported in URL-encoded format.

Parameters

bstrFullPath	A full pathname of the file to which the produced FDF file will be saved.
bstrSubmitButton	The name of an existing form field of type **button** (in case you want to include it in the FDF file, as if it had been used to trigger a **SubmitForm** action). You may pass an empty string.
bEmptyFields	A boolean to indicate whether fields with no value should be included in the produced FDF file.
arrFields	(Optional) An array of strings representing the fully-qualified names of the fields to include in the FDF file. This array may include the names of non-terminal fields, which is a fast and easy way to cause all their children to be included in the FDF file.

Return Value

None

Related Methods

ExportAsFDF

ImportAnFDF

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

Description

Imports the FDF file into an Acrobat form.

Parameters

bstrFullPath	The full pathname of the file containing the FDF file to be imported.

Return Value

None

Related Methods

ExportAsFDF

Remove

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

Description

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

Parameters

bstrFieldName	The fully-qualified name of the field to be removed from the Acrobat form. If the field has multiple child annotations, all of them are removed. If multiple fields have the same name, all are removed.

Return Value

None

Related Methods

Add

Example

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

13 Acroform Automation Properties

Field

Alignment

```
[get/set] String
```

Description

The text alignment of a text field. Valid alignments are (must be spelled exactly as shown):

```
left
center
right
```

Return Value

If the field is terminal and has multiple child annotations, a get returns the alignment for the first child, whichever annotation that happens to be.

On a set, the property is propagated to any child annotations underneath, so the field may be non-terminal.

Exceptions

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

On a get, if the field is non-terminal, an exception `AutErcNotTerminal` is returned.

Example

```
Field.Alignment = left
```

BorderStyle

```
[get/set] String
```

Description

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

Return Value

If it is terminal and has multiple child annotations, a get returns the value of the border style for the first child, whichever annotation that happens to be.

On a set, the property is propagated to any child annotations underneath, so the field may be non-terminal.

Exceptions

On a get, raises **AutErcNotTerminal** if the field is non-terminal, an exception is returned.

Example

```
Field.BorderStyle = "beveled"
```

BorderWidth

```
[get/set] short
```

Description

The thickness of the border when stroking the perimeter of a field's rectangle. If the border color is transparent, this property has no effect except in the case of a beveled border. The value **0** represents no border, and the value **3** represents a thick border.

Return Value

If it is terminal and has multiple child annotations, a get returns the value of the border width for the first child, whichever annotation that happens to be.

On a set, the property is propagated to any child annotations underneath, so the field may be non-terminal.

Exceptions

On a get, if the field is non-terminal, an exception **`AutErcNotTerminal`** is returned.

Example

```
Field.BorderWidth = 1
```

ButtonLayout

```
[get/set] short
```

Description

The layout appearance of a button. Valid values include:

- `0` - text only; the button has a caption but no icon.
- `1`- icon only; the button has an icon but no caption.
- `2`- icon over text; the icon should appear on top of the caption.
- `3`- text over icon; the text should appear on top of the icon.
- `4`- icon then text; the icon should appear to the left of the caption.
- `5`- text then icon; the icon should appear to the right of the caption.
- `6`- text over icon; the text should be overlaid on top of the icon.

If it is terminal and has multiple child annotations, a get returns the layout for the first child, whichever annotation that happens to be.

On a set, the property is propagated to any child annotations underneath, so it is OK if the field is non-terminal.

Exceptions

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

On a get, if the field is non-terminal, an exception **`AutErcNotTerminal`** is returned.

Example

```
Field.ButtonLayout = 2
```

CalcOrderIndex

```
[get/set] short
```

Description

The zero-based calculation order of fields in the document. If you want the calculation for a field **f2** to be performed after that for field **f1**, you need only set the **CalcOrderIndex** for **f2** to **f1**'s **CalcOrderIndex** + 1. The elements in the calculation order array are shifted to make room for the insertion (but the first calculation is still at index 0).

For more information, see the *Acrobat JavaScript Scripting Reference*.

Example

```
Set F1 = Fields("SubTotal")
Set F2 = Fields("Total")
F2.CalcOrderIndex = F1.CalcOrderIndex + 1
```

CharLimit

```
[get/set] short
```

Description

The limit on the number of characters that a user can type into a text field.

On a set, the property is propagated to any child annotations underneath, if any.

Exceptions

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

DefaultValue

```
[get/set] String
```

Description

The default value of the field. It returns the empty string if the field has no default value. If the field is non-terminal, an exception **`AutErcNotTerminal`** is returned.

See also **`Value`**.

Editable

```
[get/set] Boolean
```

Description

Determines whether the user can type in a selection or must choose one of the provided selections. Comboboxes can be editable; that is, the user can type in a selection.

On a set, the property is propagated to any child annotations underneath, if any.

Exceptions

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

Example

```
Field.Editable = False
```

Highlight

```
[get/set] String
```

Description

Defines how a button reacts when a user clicks it. The four highlight modes supported are:

```
none
invert
push
outline
```

If it is terminal and has multiple child annotations, a get returns the highlight for the first child, whichever annotation that happens to be.

On a set, the property is propagated to any child annotations underneath, so the field may be non-terminal.

Exceptions

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

On a get, if the field is non-terminal, an exception **`AutErcNotTerminal`** is returned.

Example

```
Field.Highlight = "invert"
```

IsHidden

```
[get/set] Boolean
```

Description

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

During get operations, if the field is non-terminal, an exception **`AutErcNotTerminal`** is returned. If it is terminal, and has multiple child annotations, a get returns the value of the hidden flag for the first child, whichever annotation that happens to be.

During set operations, the property is propagated to any child annotations underneath, so it is OK if the field is non-terminal.

Example

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

IsMultiline

```
[get/set] Boolean
```

Description

Determines whether the text field is multi-line or single-line.

On a set, the property is propagated to any child annotations underneath, if any.

Exceptions

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

Example

```
Field.IsMultiline = True
```

IsPassword

```
[get/set] Boolean
```

Description

Determines whether the field will display asterisks for the data entered. Upon submission, the actual data entered is sent. Fields that have the password attribute set will not have the data in the field saved when the document is saved to disk.

On a set, the property is propagated to any child annotations underneath, if any.

Exceptions

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

Example

```
Field.IsPassword = True
```

IsReadOnly

```
[get/set] Boolean
```

Description

The read-only characteristic of a field. When a field is read-only, the user can see the field but cannot change it. If a button is read-only, the user cannot press it to execute an action.

Because this is a field flag and not an annotation flag, both a get and a set of this property are allowed regardless of whether the field is terminal or non-terminal.

- A get on a non-terminal field retrieves that field's flag.
- A set changes the flag on all its terminal children.

IsRequired

```
[get/set] Boolean
```

Description

The required characteristic of a field. When a field is required, its value must be non-**NULL** when the user clicks a submit button that causes the value of the field to be sent to the web. If the field value is **NULL**, the user receives a warning message and the submit does not occur.

Since this is a field flag and not an annotation flag, both a get and a set of this property are allowed, regardless of whether the field is terminal or non-terminal.

A get on a non-terminal field retrieves that field's flag.

A set changes the flag on all its terminal children.

IsTerminal

```
[read-only] Boolean
```

Description

`true` if the field is terminal, otherwise **`false`**.

Example

```
Dim Field As AFORMAUTLib.Field
Dim bTerminal As Boolean

'bTerminal should be True
bTerminal = Field.IsTerminal
```

Name

```
[read-only] String
```

Description

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

NoViewFlag

```
[get/set] Boolean
```

Description

Determines whether a given field prints but doesn't display on the screen.

Set the **`NoViewFlag`** property to **`true`** to allow the field to appear when the user prints the document but not when it displays on the screen; set it to **`false`** to allow both printing and displaying.

On a get, if the field is non-terminal, an exception **`AutErcNotTerminal`** is returned. If it is terminal, and has multiple child annotations, a get returns the value of the no-view flag for the first child, whichever annotation that happens to be.

On a set, the property is propagated to any child annotations underneath, so the field may be non-terminal.

PrintFlag

```
[get/set] Boolean
```

Description

Determines whether a field prints. Set the **`PrintFlag`** property to **`true`** to allow the field to appear when the user prints the document, set it to **`false`** to prevent printing.

On a get, if the field is non-terminal, an exception **`AutErcNotTerminal`** is returned. If it is terminal, and has multiple child annotations, a get returns the value of the print flag for the first child, whichever annotation that happens to be.

On a set, the property is propagated to any child annotations underneath, so the field may be non-terminal.

Style

```
[get/set] String
```

Description

The style of a checkbox or a radio button (the glyph used to indicate that the check box or radio button has been selected).

Valid styles include:

```
check
cross
diamond
circle
star
square
```

If it is terminal and has multiple child annotations, a get returns the style for the first child, whichever annotation that happens to be.

On a set, the property is propagated to any child annotations underneath, so it is OK if the field is non-terminal.

Exceptions

During set, if the field is not of type checkbox or radio button, an exception **`AutErcNotToThisFieldType`** is returned.

On a get, if the field is non-terminal, an exception **`AutErcNotTerminal`** is returned.

Example

```
Field.Style = "star"
```

TextFont

```
[get/set] String
```

Description

The text font used when laying out the field. Valid fonts include:

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

On a set, the property is propagated to any child annotations underneath, if any.

Example

```
Field.TextFont = "Times-BoldItalic"
```

TextSize

```
[get/set] short
```

Description

The text points size used in the field. In combobox and radio button fields, the text size determines the size of the check. Valid text sizes include zero and the range from 4 to 144 inclusive.

A text size of zero means that the largest point size that can still fit in the field's rectangle should be used (in multi-line text fields and buttons this is always 12 points).

On a set, the property is propagated to any child annotations underneath, if any.

Example

```
Field.TextSize = 18
```

Type

```
[read-only] String
```

Description

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

```
text
button
combobox
listbox
checkbox
radiobutton
signature
```

Example

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

Value

```
[get/set] String
```

Description

A string that represents the value of the field. Returns the empty string if the field has no value. If the field is non-terminal, an exception **AutErcNotTerminal** is returned.

For fields of type checkbox, the value **Off** represents the unchecked state. The checked state is represented using the export value. This is also true for radio buttons (where each individual button in a group should have a different export value; see **SetExportValues**). For fields of type listbox or combobox, if an export value is defined, then that represents the value, otherwise the item name is used.

These remarks apply also to **DefaultValue**.

Example

```
Dim arrExp(1) As String
arrExp(0) = "Visa"
arrExp(1) = "Mastercard"
Field.SetExportValues arrExp
Field.Value = arrExp(0)
```

Fields

Count

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

Description

The number of items in the collection.

Example

```
Dim Field As AFORMAUTLib.Field
Dim nFields As Long

nFields = Fields.Count

For Each Field In Fields
If Field.IsTerminal Then
print Field.Value
End If
Next Field
```

Item

```
[read-only] IDispatch*
```

Description

Takes the fully qualified name of the field (for example, "name.last") as a parameter, and returns the **Field** object for it. It is the default member of the **Fields** interface (that is, the property invoked if the object name is specified by itself without a property or a method in the controller script).

Example

```
Dim Field As AFORMAUTLib.Field
Dim nFields As Long

Set Field = Fields.Item("name.last")
'Since Item is the default_ property:
Set Field = Fields("name.last")
```

_NewEnum

```
[read-only] IUnknown*
```

Description

The **`IEnumVariant`** enumerator for the collection.

NOTE: You do not need to call this property directly. Visual Basic calls it in the background whenever the code contains a "For Each Field In Fields" loop. For example:

```
For Each Field in Fields
If Field.IsTerminal
print Field.Value
End If
Next Field
```

14 Acrobat Search

The Acrobat Search plug-in allows users to perform text searches in PDF documents. It adds menus, menu items, toolbar buttons, and a Search panel to the Acrobat application.

The Search plug-in exports a Host Function Table (HFT) containing several methods that can be used by other plug-ins.

Search supports interapplication communication (IAC) in the form of Apple events on the Macintosh and DDE messages under the Windows operating system. These Apple events and DDE messages allow remote clients to submit search queries and manipulate a list of indexes (the list of indexes is referred to as the *shelf*).

This section describes the HFT and IAC APIs supported by the Acrobat Search plug-in. It also contains the names of items added to the Acrobat application user interface by the Search plug-in.

For more information, see the *Acrobat and PDF Library API Overview*.

Contents

This portion of the document contains the following sections:

- Search Apple Events describes in detail each Apple event, including parameters and return value.
- DDE Messages describes in each DDE message and its arguments.
- Search Lists describes Search dialog boxes, menu item names, and toolbar button names.

15 Search 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_CLIENTONLY, 0);
hszServerName = DdeCreateStringHandle(id, "Acrobat Search", 0);
hszTopicName = DdeCreateStringHandle(id, "Acrobat Search", 0);
hConv = DdeConnect(id, hszServerName, hszTopicName, NULL);
```

After a connection has been made, a single poke transaction will submit a search query. Two types of queries are supported: simple query and query.

Simple Query Item

A simple query has the item name “SimpleQuery”. When using a simple query, pass only a string that contains the query, using the ASQL query parser’s format (see **QLangType_CQL** in Table 1). It is not possible to choose another parser or to set word options using the simple query item.

Query Item

Query has the item name "Query". When using query, a **QueryData** structure is passed. This structure contains the query, as well as specifying the query parser to use and additional options.

```
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; //deprecated in Acrobat 6.0
     uns16 nSortOffset[QP_MAX_SORT_FIELDS]; //deprecated in Acrobat 6.0
     boolean bSortWays[QP_MAX_SORT_FIELDS]; //deprecated in Acrobat 6.0
     unsigned char cData[1];
} QueryData;
```

qlt	The query language type. Must be one of the values shown in Table 1.
bOverrideWordOptions	Indicates that the client wishes to use different word options than those currently set by the user.
nWordOptions	The word options. Must be an **OR** of the values shown in Table 2.
nMaxDocs	If non-zero, the client wishes to use a different limit for the maximum number of documents than the limit currently set by the user.
nSortOffsets	A list of offsets into the **cData** chunk. Each offset points to a **NULL**-terminated string containing the field name. **NOTE:** This value has no effect in Acrobat 6.0 or later, because sort options are not valid.
nQueryOffset	An offset into the **cData** chunk that points to a **NULL**-terminated string containing the query to execute.

nNumSorts	The number of fields in the sort spec. If this number is **0**, the plug-in uses the current sort spec set by the user. **NOTE:** This value has no effect in Acrobat 6.0 or later, because sort options are not valid.
bSortWays	A list of sort order flags, one for each sort field. **true** indicates an ascending sort, and **false** indicates a descending sort. **NOTE:** This value has no effect in Acrobat 6.0 or later, because sort options are not valid.

TABLE 1 Query language type constants

QLangType_Simple	Allows only simple phrase searches; does not allow boolean searching. **NOTE:** This query type does not work in the DDE interface of the Search plug-in shipped with version 2.0 of Acrobat.
QLangType_CQL	Allows boolean searches using **AND**, **OR**, and **NOT**, as described in the Acrobat Search plug-in's online help file.
QLangType_Passthrough	The Verity BooleanPlus® query language. Contact Verity for further information on this language.

TABLE 2 Word option bit-flag constants

QPON_Case	The search is case-sensitive.
QPON_Stemming	Find not only the specified word, but other words that have the same stem (for example, run and ran have the same stem).
QPON_SoundsLike	Find not only the specified word, but other words that sound like it.
QPON_Thesaurus	Find not only the specified word, but other words that have the same meaning.

TABLE 2 ***Word option bit-flag constants***

QPON_Proximity	Consider the proximity of results when using the **AND** operator to look for more than one word in a document. Without this option, **AND** terms can be anywhere in a document. Searching for "red" and "blue," for example, finds a document where "red" is the first word on the first page and where "blue" is the last word on the last page. With this option, however, **AND** terms must be within two or three pages of each other to be found. Also, the closer **AND** terms appear together, the higher the relevance ranking of the document that contains them.
QPON_Refine	Do not search the entire list of indices, but only the documents that matched the previous search. This is used to refine the results of the previous search.

To create and populate this structure correctly, the client must know the sum of the lengths of each sort field (**sls**), the length of the query (**lq**), and the size of the **QueryData** structure. The client then allocates memory as follows:

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

For example, if the query was "Adobe" and the sort spec was "Title" ascending and "Score" descending then the structure would be packed as follows:

```
memset(qd, 0, nSize);
qd->nQueryOffset = 0;
strcpy(&cData[0], "Adobe");
qd->nNumSort = 2;
qd->nSortOffset[0] = strlen("Adobe") + 1;
qd->bSortWays[0] = TRUE;
strcpy(&cData[qd->nSortOffset[0]], "Title");
qd->bSortWays[1] = FALSE;
qd->nSortOffset[1] = qd->nSortOffset[0] + strlen("Title") + 1;
strcpy(&cData[qd->nSortOffset[1]], "Score");
```

Manipulating 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;
```

eAction	The operation to be performed on the index, and must be one of values listed in Table 3.
nIndexOffset	An offset into the **cData** chunk that points to a **NULL**-terminated string containing the **.PDX** file representing the index.
nTempNameOffset	An offset into **cData**. It points to a temporary name that is displayed by the Search plug-in when the index is unavailable. This field must specify an offset either to an empty string (**\0**) or to a non-empty C string.

TABLE 3 Search plug-in index operation selectors for DDE messages

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.

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");
```

16 Search Apple Events

SearchAddIndex

Description

Adds a specified index to the shelf.

Apple Event ID

`kSearchAddIndex ('addx')`

Apple Event Parameters

kIndexListTag ('SilP'), typeLongInteger	An 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**
kFlagTag ('Flag'), typeLongInteger	Index flags. See **SearchGetIndexFlags** for a description of them. The **kIndexAvailable** flag should always be set.

Return Value

`kIndexTag ('SixP'), typeLongInteger`

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

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

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

SearchCountIndexList

Description

Gets the number of indices currently on the shelf.

Apple Event ID

`kSearchCountIndexList ('cidx')`

Apple Event Parameters

`kIndexListTag ('SilP'), typeLongInteger`	An opaque **`void*`** representing the shelf, obtained from **`SearchGetIndexList`**.

Return Value

`kIndexListTag ('SilP'), typeLongIneger`

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

SearchDoQuery

Description

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

Apple Event ID

`kSearchDoQuery ('kwry')`

Apple Event Parameters

kQueryStringTag ('Qury'), typeChar	The query string, a **NULL**-terminated block of text. Its format is the same as what a user would type into the search Query window, and depends on the search language specified by **kParserTag**.
kParserTag ('Prsr'), typeShortInteger	The query parser to use; may be one of (see `SrchType.h`): • **kParserSimple** 0 — Allows only simple phrase searches; does not allow boolean searching. • **kParserCQL** 1 — Allows boolean searches using **AND**, **OR**, and **NOT**, as described in the Acrobat Search plug-in's online help file. • **kParserBPlus** 2 — The Verity BooleanPlus query language. Contact Verity for further information on this language.
kSortSpecTag ('Sort'), typeAEList	A list of C strings representing fields to sort by. The first element is the first level sort, the second is the second level sort, and so forth. Each string may be any field that appears in the index, plus **Score** (which sorts results by relevance ranking). Some common fields are Title, ModificationDate, CreationDate, and Keywords.
kWordOptionsTag ('WOpt'), typeLongInteger	A bit field of word options. Must be a logical OR of the values listed below in Search Plug-in Word Options For Apple Events. The manner in which the options are used depends on the value associated with **kOptionsOverrideTag**.
kOptionsOverrideTag ('WOer'), typeShortInteger	Flag that indicates whether the word options are **OR**'ed with the search options set in the user interface, or used instead of them. If **0**, the word options are **OR**'ed with the user interface search options, and the resulting value is used. If non-zero, the word options are used instead of the user interface search options.

`kMaxDocsTag ('MaxD'), typeShortInteger`	The maximum number of documents to display in the Results window. If more documents than this have hits, only the first **`maxDocs`** are displayed. **`maxDocs`** cannot be greater than 999.

Return Value

None

Search Plug-in Word Options For Apple Events

`kWordOptionCase`	The search is case-sensitive.
`kWordOptionStemming`	Find not only the specified word, but other words that have the same stem (for example, run and ran have the same stem).
`kWordOptionSoundsLike`	Find not only the specified word, but other words that sound like it.
`kWordOptionThesaurus`	Find not only the specified word, but other words that have the same meaning.
`kWordOptionProximity`	Consider the proximity of results when using the **AND** operator to look for more than one word in a document. Without **`kWordOptionProximity`**, **AND** terms can be anywhere in a document. Searching for "red" and "blue," for example, finds a document where "red" is the first word on the first page and where "blue" is the last word on the last page. With **`kWordOptionProximity`**, however, **AND** terms must be within two or three pages of each other to be found. Also, with **`kWordOptionProximity`**, the closer **AND** terms appear together, the higher the relevance ranking of the document that contains them.
`kWordOptionRefine`	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.

SearchGetIndexByPath

Description

Gets the index that has the specified path. The index must already be on the shelf. The index can be passed to other Search Apple events to remove it from the shelf, obtain its title, and so forth.

Apple Event ID

`kSearchGetIndexByPath ('fpdx')`

Apple Event Parameters

kIndexListTag ('SilP'), typeLongInteger	An 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

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

SearchGetIndexFlags

Description

Get the flags for an index.

Apple Event ID

`kSearchGetIndexFlags ('gfdx')`

Apple Event Parameters

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

Return Value

`kFlagTag ('Flag'), typeLongInteger`

A logical **OR** of the following:

`kIndexAvailableFlag (1L << 0)` — Set if the index is available for searching.

`kIndexSelectedFlag (1L << 1)` — Set if the index appears with a check mark in the Search plug-in's user interface.

`kIndexPtrInvalidFlag (1L << 31)` — Set if the index is not valid or is no longer valid.

SearchGetIndexList

Description

Gets a list of the indices currently on the shelf.

Apple Event ID

`kSearchGetIndexList ('gidx')`

Apple Event Parameters

None

Return Value

`kIndexListTag ('SilP'), typeLongInteger`

An opaque **`void*`** representing the list of indices currently on the shelf. This value can subsequently be used by other search Apple events to obtain information about a specific index, the number of indices on the shelf, and so forth.

SearchGetIndexPath

Description

Gets the full path to an index.

Apple Event ID

`kSearchGetIndexPath ('gpdx')`

Apple Event Parameters

kIndexTag ('SixP'), typeLongInteger	An opaque **void*** representing the index whose path is to be obtained. The index may be obtained using **SearchGetIndexByPath**, **SearchGetNthIndex**, or **SearchAddIndex**.

Return Value

kPathTag ('Path'), typeChar

A **NULL**-terminated character string representing the full path of the index. Returns an empty string if the requested index is not valid.

SearchGetIndexTitle

Description

Gets the title of an index.

Apple Event ID

`kSearchGetIndexTitle ('gtdx')`

Apple Event Parameters

kIndexTag ('SixP'), typeLongInteger	An opaque **void*** representing the index whose title is to be obtained. The index may be obtained using **SearchGetIndexByPath**, **SearchGetNthIndex**, or **SearchAddIndex**.

Return Value

kTitleTag ('Titl'), typeChar

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

SearchGetNthIndex

Description

Gets the nth index on the shelf. The index can be passed to other Search Apple events to remove it from the shelf, obtain its title, and so forth.

Apple Event ID

`kSearchGetNthIndex ('fndx')`

Apple Event Parameters

kIndexListTag ('SilP'), typeLongInteger	An opaque **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

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

SearchRemoveIndex

Description

Removes the specified index from the shelf.

Apple Event ID

`kSearchRemoveIndex ('rmdx')`

Apple Event Parameters

kIndexListTag ('SilP'), typeLongInteger	An opaque **void*** representing the shelf, obtained from **SearchGetIndexList**.
kIndexTag ('SixP'), typeLongInteger	An opaque **void*** representing the index to be removed. The index may be obtained using **SearchGetIndexByPath**, **SearchGetNthIndex**, or **SearchAddIndex**.

Return Value

None

SearchSetIndexFlags

Description

Sets the flags for an index.

Apple Event ID

`kSearchSetIndexFlags ('sfdx')`

Apple Event Parameters

`kIndexTag ('SixP'), typeLongInteger`	An opaque **`void*`** representing an index.
`kFlagTag ('Flag'), typeLongInteger`	Index flags. See the description in **`SearchGetIndexFlags`**. In practice, **`kIndexAvailableFlag`** should always be set.

Return Value

`kFlagTag ('Flag'), typeLongInteger`

Index flags. See the description in **`SearchGetIndexFlags`**. This value is returned because it is possible for a request to set a flag to fail.

17 Search Lists

The Search plug-in adds a new menu, menu items, and toolbar buttons to the Acrobat application.

Menu Names

The Search plug-in adds the following menu to Acrobat.

Menu name	Description
AcroSrch:ToolsSubMenu	Acrobat Search submenu of Edit menu.

Menu Item Names

The Search plug-in adds the following menu items to Acrobat.

Menu item name	Description
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	A separator item in the Search tools menu.
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.

Toolbar Button Names

The Search plug-in adds the following buttons to the Acrobat toolbar

Button name	Description
AcroSrch:Separator	Separator (not visible).
AcroSrch:Query	Displays the Acrobat Search plug-in's query dialog.
AcroSrch:Results	Displays the Acrobat Search plug-in's search results dialog.
AcroSrch:Prev	Goes to the previous hit in the Acrobat Search plug-in's results list.
AcroSrch:Next	Goes to the next hit in the Acrobat Search plug-in's results list.