

ACCESSING PDF (PORTABLE DOCUMENT FORMAT) 4

WHAT IS AROBAT?.....	4
IMPORTANT!!	4
ACCESSIBILITY FOR PDF	4
WHAT IS THE AROBAT SDK?	5
OLE AUTOMATION OBJECTS	5
HELLO AROBAT.....	5
OLE AUTOMATION DOM	6
ACROEXCH.POINT OBJECT.....	6
AcroExch.Point.X Property	7
AcroExch.Point.Y Property	7
ACROEXCH.RECT OBJECT.....	7
AcroExch.Rect.Bottom Property	7
AcroExch.Rect.Left Property	7
AcroExch.Rect.Right Property.....	8
AcroExch.Rect.Top Property	8
ACROEXCH.TIME OBJECT	8
AcroExch.Time.Date Property	8
AcroExch.Time.Hour Property	8
AcroExch.Time.Millisecond Property	9
AcroExch.Time.Minute Property	9
AcroExch.Time.Month Property.....	9
AcroExch.Time.Second Property.....	9
AcroExch.Time.Year Property.....	9
ACROEXCH.APP OBJECT	10
AcroExch.App.CloseAllDocs() Method	10
AcroExch.App.Exit() Method.....	11
AcroExch.App.GetActiveDoc() Method.....	11
AcroExch.App.GetActiveTool() Method.....	12
AcroExch.App.GetAVDoc() Method	13
AcroExch.App.GetFrame() Method.....	14
AcroExch.App.GetNumAVDocs() Method	14
AcroExch.App.GetLanguage() Method	15
AcroExch.App.GetPreferenceEx() Method	15
AcroExch.App.Hide() Method.....	15
AcroExch.App.Lock() Method	16
AcroExch.App.Minimize() Method	16
AcroExch.App.Maximize() Method	17
AcroExch.App.MenuItemExecute() Method	17
AcroExch.App.MenuItemIsEnabled() Method.....	18
AcroExch.App.MenuItemIsMarked() Method.....	19
AcroExch.App.MenuItemRemove() Method.....	19
AcroExch.App.SetActiveTool() Method	20
AcroExch.App.SetFrame() Method	20
AcroExch.App.SetPreference() Method	21
AcroExch.App.Show() Method	21
AcroExch.App.ToolButtonIsEnabled() Method	21
AcroExch.App.ToolButtonRemove() Method.....	23

AcroExch.App.Unlock() Method.....	23
ACROEXCH.AVDOC OBJECT	24
AcroExch.AVDoc.BringToFront() Method.....	24
AcroExch.AVDoc.ClearSelection() Method.....	24
AcroExch.AVDoc.Close() Method.....	24
AcroExch.AVDoc.FindText() Method	25
AcroExch.AVDoc.GetAVPageView() Method	26
AcroExch.AVDoc.GetFrame () Method.....	26
AcroExch.AVDoc.GetPDDoc () Method.....	27
AcroExch.AVDoc.GetTitle () Method.....	27
AcroExch.AVDoc.GetViewMode () Method	27
AcroExch.AVDoc.IsValid () Method	29
AcroExch.AVDoc.Maximize() Method.....	29
AcroExch.AVDoc.Open() Method	29
AcroExch.AVDoc.OpenInWindowEx() Method.....	30
AcroExch.AVDoc.PrintPages() and PrintPagesSilent() Methods.....	31
AcroExch.AVDoc.SetFrame() Method.....	31
AcroExch.AVDoc.SetTextSelection() Method.....	32
AcroExch.AVDoc.SetTitle() Method	32
AcroExch.AVDoc.SetViewMode() Method	32
AcroExch.AVDoc.ShowTextSelect () Method.....	33
ACROEXCH.AVPAGEVIEW OBJECT	33
AcroExch.AVPageView.DevicePointToPage () Method	34
AcroExch.AVPageView.DoGoBack () Method	34
AcroExch.AVPageView.DoGoForward() Method	35
AcroExch.AVPageView.GetAVDoc() Method	35
AcroExch.AVPageView.GetDoc() Method.....	35
AcroExch.AVPageView.GetPage() Method.....	36
AcroExch.AVPageView.GetPageNum() Method.....	36
AcroExch.AVPageView.GetZoom() Method.....	37
AcroExch.AVPageView.GetZoomType() Method.....	37
AcroExch.AVPageView.Goto() Method	37
AcroExch.AVPageView.ReadPageDown() and ReadPageUp() Methods.....	38
AcroExch.AVPageView.ScrollTo() Method	38
AcroExch.AVPageView.ZoomTo() Method	39
ACROEXCH.HILITELIST OBJECT	39
AcroExch.HiliteList.Add() Method	39
ACROEXCH.PDANNOT OBJECT.....	40
ACROEXCH.PDBOOKMARK OBJECT	40
AcroExch.PDBookmark.Destroy () Method.....	40
AcroExch.PDBookmark.GetByTitle() Method.....	41
AcroExch.PDBookmark.GetTitle() Method	42
AcroExch.PDBookmark.IsValid() Method.....	42
AcroExch.PDBookmark.Perform() Method	42
AcroExch.PDBookmark.SetTitle() Method.....	43
ACROEXCH.PDDOC OBJECT	43
AcroExch.PDDoc.AcquirePage () Method	43
AcroExch.PDDoc.ClearFlags () Method	44
AcroExch.PDDoc.Close () Method.....	45
AcroExch.PDDoc.Create () Method	45
AcroExch.PDDoc.CreateTextSelect () Method	45

AcroExch.PDDoc.CreateThumbs () Method	46
AcroExch.PDDoc.CropPages () Method	46
AcroExch.PDDoc.DeletePages () Method	47
AcroExch.PDDoc.DeleteThumbs () Method	48
AcroExch.PDDoc.GetFileName () Method	48
AcroExch.PDDoc.GetFlags () Method	48
AcroExch.PDDoc.GetInfo () Method	49
AcroExch.PDDoc.GetInstanceID () Method	50
AcroExch.PDDoc.GetJSObject () Method.....	50
AcroExch.PDDoc.GetNumPages () Method.....	51
AcroExch.PDDoc.GetPageMode () Method.....	52
AcroExch.PDDoc.GetPermanentID () Method.....	52
AcroExch.PDDoc.InsertPages () Method	52
AcroExch.PDDoc.MovePage () Method.....	53
AcroExch.PDDoc.Open () Method	53
AcroExch.PDDoc.OpenAVDoc () Method.....	53
AcroExch.PDDoc.ReplacePages () Method.....	54
AcroExch.PDDoc.Save () Method	54
AcroExch.PDDoc.SetFlags () Method.....	55
AcroExch.PDDoc.SetInfo () Method.....	55
AcroExch.PDDoc.SetPageMode () Method.....	55
ACROEXCH.PDPAGE OBJECT.....	55
AcroExch.PDPage.AddAnnot () Method.....	55
AcroExch.PDPage.AddNewAnnot () Method	56
AcroExch.PDPage.CreatePageHilite () Method	56
AcroExch.PDPage.CreateWordHilite () Method	58
AcroExch.PDPage.GetAnnotIndex() Method.....	60
AcroExch.PDPage.GetDoc() Method	61
AcroExch.PDPage.GetNumAnnots() Method	61
AcroExch.PDPage.GetNumber() Method.....	62
AcroExch.PDPage.GetRotate() Method	62
AcroExch.PDPage.GetSize() Method.....	62
AcroExch.PDPage.RemoveAnnot() Method	63
AcroExch.PDPage.SetRotate () Method	63
ACROEXCH.PDTEXTSELECT OBJECT	65
AcroExch.PDTextSelect.Destroy () Method.....	65
AcroExch.PDTextSelect.GetBoundingRect () Method	65
AcroExch.PDTextSelect.GetNumText () Method	66
AcroExch.PDTextSelect.GetPage () Method.....	68
AcroExch.PDTextSelect.GetText () Method	69
ACROFORM OLE AUTOMATION	69
AFORMAUT.APP OBJECT	70
AFORMAUT.FIELD OBJECT	70
AFORMAUT.FIELDS OBJECT	70
DEVELOPER FAQ	70
USER AND DEVELOPER RESOURCES	70
What Do I Need to Download from the Web to Get the Acrobat SDK?	70
Where Can I Get Help With Acrobat Product Issues?	71
What Are the API Differences Between Acrobat and Adobe Reader?.....	71
What API Methods Are Available to Modify PDF Documents?.....	71
How Can I Extract Text From PDF Documents Using the Acrobat SDK?	72

How Do I Use Command Lines with Acrobat and Adobe Reader on Windows?	72
PDF TASKS	72
Find Text in all PDF's File Open.....	72
Invoking AVCommands Programmatically.....	73
Searching words in PDF.....	74
APPENDIX 15	77
View Mode Enumeration	77
Toolbar Items Names	78
Zoom Strategy Enumeration	78
Page Rotation Enumeration.....	79
Document Flags Enumeration.....	79

Accessing PDF (Portable Document Format)

What Is Acrobat?

Adobe® Acrobat® consists of a family of products for creating, modifying, indexing, searching, displaying, and manipulating **PDF** (Portable Document Format) files

Adobe **Reader** for viewing, navigating, and printing **PDF** documents. Adobe Reader is free software that lets anyone view and print Adobe PDF files on all major computer platforms, as well as fill in and submit Adobe **PDF** forms. Adobe has distributed more than 500 million copies of the software worldwide.

Important!!

The Microsoft Windows version of Acrobat is an OLE Automation server. In order to use the OLE objects made available by Acrobat, you must have the full Acrobat product installed on your system!!

Accessibility for PDF

The **Adobe Portable Document Format (PDF)** is a file format for representing documents in a manner independent of the application software, hardware, and operating system used to create them, as well as of the output device on which they are to be displayed or printed. PDF files specify the appearance of pages in a document in a reliable, device independent manner

Adobe provides methods to make the content of a **PDF** file available to assistive technology such as screen readers.

On The Microsoft Windows operating system, Adobe Acrobat and Adobe Reader export **PDF** content as **COM** objects. applications can interface with Acrobat or Adobe Reader in two Ways :

1. Through Microsoft's Active Accessibility(**MSAA**) interface, using **MSAA** objects that Acrobat or Adobe Reader exports.
2. Directly through exported **COM** objects that allow access to the **PDF** document's internal structure, called the Document Object Model(**DOM**).

For Unix Platforms, Adobe Reader supports the Gnome accessibility architecture.

What Is the Acrobat SDK?

The Acrobat **SDK** is a set of tools that help you develop software that interacts with Acrobat technology. The SDK contains header files, type libraries, simple utilities, sample code, and documentation. Using the Acrobat SDK, you can develop software that integrates with Acrobat and Adobe Reader

For more information about acrobat SDK go to : <http://www.adobe.com/devnet/acrobat/>

The Adobe Acrobat 8.1 SDK is now available free of charge to all users. Developers can use the SDK to create software and plug-ins to interact and customize Adobe Acrobat and Adobe Reader. To download Acrobat SDK : <http://www.adobe.com/devnet/acrobat/?tab:downloads=1>

OLE Automation Objects

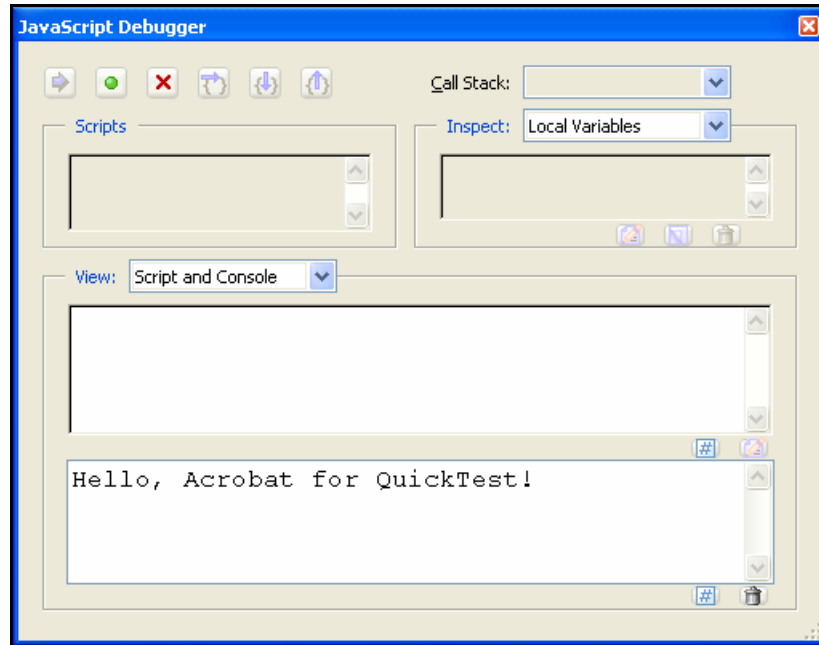
All the samples will be on the QTTutorial.pdf file (QuickTest Tutorial pdf File) located in your Mercury Interactive Installation Folder /help/ QTTutorial.pdf

Hello Acrobat

```
Option Explicit

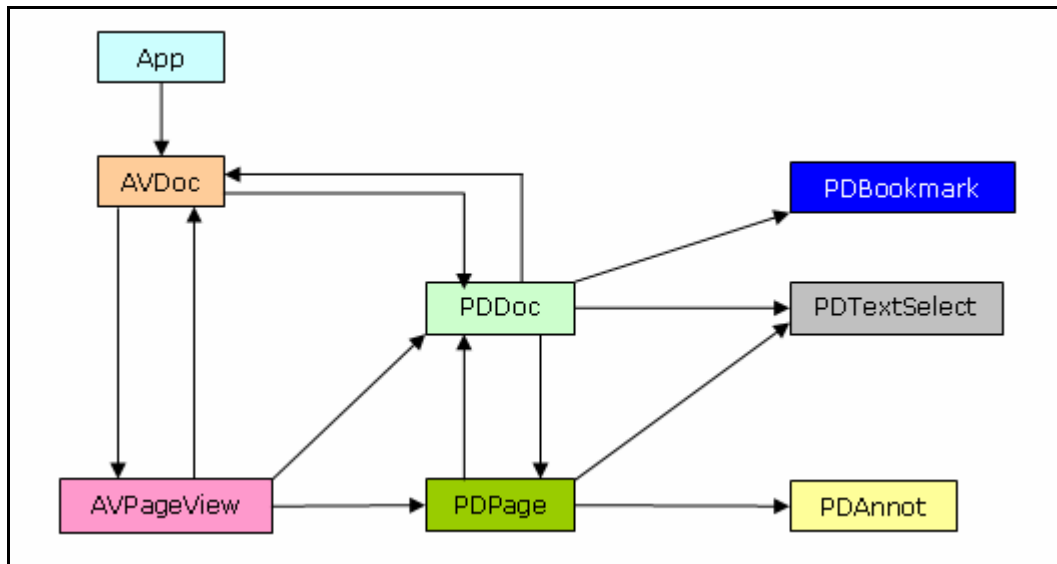
Dim gApp, gPDDoc, jso

Set gApp = CreateObject( "AcroExch.App" )
Set gPDDoc = CreateObject( "AcroExch.PDDoc" )
If gPDDoc.Open( "c:\MyPdfFile.pdf" ) Then
    Set jso = gPDDoc.GetJSObject()
    jso.console.Show
    jso.console.Clear
    jso.console.println ( "Hello, Acrobat for QuickTest!" )
    gApp.Show
End If
Set jso = Nothing : Set gPDDoc = Nothing : Set gApp = Nothing
```



OLE Automation DOM

All the samples will be on the QTTutorial.pdf file (QuickTest Tutorial pdf File) located in your Mercury Interactive Installation Folder /help/ QTTutorial.pdf



AcroExch.Point Object

A point, specified by its x- and y-coordinates.

AcroExch.Point.X Property

Description

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

Syntax

```
returnValue = object.X
```

Return Value

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

AcroExch.Point.Y Property

Description

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

Syntax

```
returnValue = object.Y
```

Return Value

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

AcroExch.Rect Object

A rectangle, specified by the top left and bottom right points.

AcroExch.Rect.Bottom Property

Description

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

Syntax

```
returnValue = object.Bottom
```

Return Value

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

AcroExch.Rect.Left Property

Description

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

Syntax

```
returnValue = object.Left
```

Return Value

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

AcroExch.Rect.Right Property

Description

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

Syntax

```
returnValue = object.Right
```

Return Value

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

AcroExch.Rect.Top Property

Description

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

Syntax

```
returnValue = object.Top
```

Return Value

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

AcroExch.Time Object

A specified time, accurate to the millisecond, including the day of the week.

AcroExch.Time.Date Property

Description

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

Syntax

```
returnValue = object.Date
```

Return Value

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

AcroExch.Time.Hour Property

Description

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

Syntax

```
returnValue = object.Hour
```

Return Value

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

AcroExch.Time.Millisecond Property

Description

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

Syntax

```
returnValue = object.Millisecond
```

Return Value

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

AcroExch.Time.Minute Property

Description

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

Syntax

```
returnValue = object.Minute
```

Return Value

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

AcroExch.Time.Month Property

Description

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

Syntax

```
returnValue = object.Month
```

Return Value

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

AcroExch.Time.Second Property

Description

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

Syntax

```
returnValue = object.Second
```

Return Value

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

AcroExch.Time.Year Property

Description

Gets or sets the year from an AcroTime.

Syntax

```
returnValue = object.Year
```

Return Value

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

AcroExch.App Object

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.App.CloseAllDocs() Method

Description

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

Syntax

```
returnValue = object.CloseAllDocs
```

Return Value

-1 if successful, 0 if not.

Note

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

Related Methods

- **AVDoc.Close()**
- **AVDoc.Open()**
- **AVDoc.OpenInWindow()**
- **PDDoc.Close()**
- **PDDoc.Open()**
- **PDDoc.OpenAVDoc()**

Example

```
Option Explicit
Dim AcroApp, returnValue

Set AcroApp = CreateObject( "AcroExch.App" )
If AcroApp.GetNumAVDocs > 0 Then
    returnValue = AcroApp.CloseAllDocs()
End If
Print "AcroApp.CloseAllDocs returned with code ---> " & Cstr( returnValue )
Set AcroApp = Nothing
```

AcroExch.App.Exit() Method

Description

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

Syntax

```
returnValue = object.Exit
```

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().

Note

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

Example

```
Option Explicit
Dim AcroApp, returnValue

Set AcroApp = CreateObject( "AcroExch.App" )
If AcroApp.GetNumAVDocs > 0 Then
    returnValue = AcroApp.CloseAllDocs()
End If
Print "AcroApp.CloseAllDocs returned with code ---> " & Cstr( returnValue )
AcroApp.Exit()
Set AcroApp = Nothing
```

AcroExch.App.GetActiveDoc() Method

Description

Gets the frontmost document.

Syntax

```
Set avDoc = object.GetActiveDoc
```

Return Value

An AcroExch.AVDoc that represents the frontmost AcroExch.AVDoc object. If there are no documents open, it will return **Nothing**.

Related Methods

- App.GetAVDoc()

Example

```
Option Explicit
Dim AcroApp, AcroAvDoc

Set AcroApp = CreateObject( "AcroExch.App" )
```

```

Set AcroAVDoc = AcroApp.GetActiveDoc()
If Not AcroAVDoc Is Nothing Then
    returnValue = AcroApp.CloseAllDocs()
End If
AcroApp.Exit()Set AcroApp = Nothing

```

AcroExch.App.GetActiveTool() Method

Description

Gets the name of the currently active tool.

Syntax

```
returnValue = object.GetActiveTool
```

Return Value

Returns **Nothing** if there is no active tool. Returns the name of the currently active tool otherwise.

Tool Name	UI Label
Movie	"Movie Tool (M)"
Link	"Link Tool (L)"
Thread	"Article Tool (A)"
Crop	"Crop Tool (C)"
Widget	"Form Tool (F)"
Touch-Up Text Selection	"TouchUp Text Tool (T)"
Object Selection Type	"TouchUp Object Tool (T)" / "TouchUp Order Tool (T)"
Hand	"Hand Tool (H)"
Zoom	"Zoom In Tool (Z)" / "Zoom Out Tool (Z)"
Select	"Text Select Tool (V)" / "Column Select Tool (V)"
BCLC:Table/Formatted_TextZoneTool	"Table/Formatted Text Select Tool (V)"
SelectGraphics	"Graphics Select Tool (G)"
Text	"Note Tool (S)"
FreeText	"FreeText Tool (S)"
Sound	"Sound Attachment Tool (S)"
Stamp	"Stamp Tool (S)"
FileAttachment	"File Attachment Tool (S)"
Ink	"Pencil Tool (N)"
Square	"Square Tool (N)"
Circle	"Circle Tool (N)"
Line	"Line Tool (N)"
Highlight	"Highlight Tool (U)"

StrikeOut	"Strikeout Tool (U)"
Underline	"Underline Tool (U)"
Squiggly	Not exposed through UI.
DigSigTool	"Digital Signature Tool (D)"

Related Methods

- `App.SetActiveTool()`

Example

```
Option Explicit
Dim AcroApp, AcroAvDoc

Set AcroApp = CreateObject( "AcroExch.App" )
Set AcroAVDoc = AcroApp.GetActiveDoc()
If Not AcroAVDoc Is Nothing Then
    returnValue = AcroApp.CloseAllDocs()
End If
AcroApp.Exit()Set AcroApp = Nothing
```



AcroExch.App.GetAVDoc() Method

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.

Syntax

```
Set avDoc = object.GetAVDoc( nIndex )
```

Return Value

An AVDoc object for the specified AcroExch.AVDoc document, or **Nothing** if nIndex is greater than the number of open documents.

Arguments

Parameter	Description
<i>nIndex</i>	The index of the document to get.

Related Methods

- `App.GetActiveDoc()`

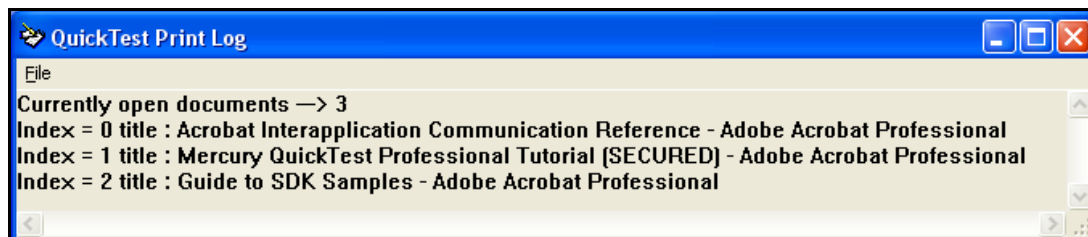
Example

```
Option Explicit
Dim AcroApp, AcroAvDoc
Dim nDoc
```

```

Set AcroApp = CreateObject( "AcroExch.App" )
If AcroApp.GetNumAVDocs > 0 Then
    Print "Currently open documents ----> " & AcroApp.GetNumAVDocs
    For nDoc = 0 To AcroApp.GetNumAVDocs - 1
        Set AcroAvDoc = AcroApp.GetAVDoc( nDoc )
        Print "Index = " & nDoc & " title : " & AcroAvDoc.GetTitle()
    Next
Else
    Print "No open pdf documents found."
End If
AcroApp.Exit()
Set AcroApp = Nothing : Set AcroAvDoc = Nothing

```



AcroExch.App.GetFrame() Method

Description

Gets the window's frame.

Syntax

```
Set rect = object.GetFrame()
```

Return Value

The AcroExch.Rect object that represents the window's frame, specified as an AcroExch.Rect

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.

Related Methods

- App.Maximize()
- App.SetFrame()

AcroExch.App.GetNumAVDocs() Method

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`.

Syntax

```
count = object.GetNumAVDocs()
```

Return Value

The number of open AcroExch.AVDoc objects.

Related Methods

- App.GetActiveDoc()
- App.GetAVDoc()

AcroExch.App.GetLanguage() Method

Description

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

Syntax

```
lang = object.GetLanguage()
```

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, CZE – Czech, SUO – Finnish, JPN – Japanese
- More ...

Related Methods

- App.GetPreference()
- App.SetPreference()

AcroExch.App.GetPreferenceEx() Method

Description

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

Syntax

```
returnValue = object.GetPreference( nType )
```

Arguments

Parameter	Description
<i>nType</i>	The name of the preferences item whose value is obtained.

Return Value

The value of the specified preference item.

AcroExch.App.Hide() Method

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.

Syntax

```
returnValue = object.Hide()
```

Return Value

True if successful, **false** otherwise.

Related Methods

- App.Show()

AcroExch.App.Lock() Method

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.

Syntax

```
returnValue = object.Lock( lockedBy )
```

Arguments

Parameter	Description
<i>lockedBy</i>	A string that is used as the name of the application that has locked the Acrobat application.

Return Value

true if the Acrobat application was locked successfully, **false** otherwise. Locking will fail if the Acrobat application is visible.

Related Methods

- App.Unlock()

AcroExch.App.Minimize() Method

Description

Minimizes the Acrobat application.

Syntax


```
returnValue = object.Minimize( bMinimize )
```

Arguments

Parameter	Description
<i>bMinimize</i>	If a positive number, the Acrobat application is minimized. If 0, the Acrobat application is returned to its normal state.

Return Value

True if successful, **false** otherwise.

AcroExch.App.Maximize() Method

Description

Maximizes the Acrobat application.

Syntax

```
returnValue = object.Maximize( bMaximize )
```

Arguments

Parameter	Description
<i>bMaximize</i>	If a positive number, the Acrobat application is maximized. If 0, the Acrobat application is returned to its normal state.

Return Value

True if successful, **false** otherwise.

Related Methods

- App.GetFrame()
- App.SetFrame()

AcroExch.App.MenuItemExecute() Method

Description

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

Syntax

```
returnValue = object.MenuItemExecute( menuItemName )
```

Arguments

Parameter	Description
<i>menuItemName</i>	The language-independent name of the menu item to execute. <ul style="list-style-type: none"> ■ ShowHideToolBar ■ ShowHideMenuBar (new in Acrobat 3.0) ■ ShowHideClipboard ■ endShowHideGroup ■ Cascade

	<ul style="list-style-type: none"> ■ TileHorizontal ■ TileVertical ■ CloseAll
--	--

Return Value

Returns **true** if the menu item executes successfully, **false** if the menu item is missing or is not enabled.

Related Methods

- App.MenuItemIsEnabled()
- App.MenuItemIsMarked()
- App.MenuItemRemove()

Example

```
Option Explicit

Dim AcroApp, AcroAVDoc
Dim gPDFPath

gPDFPath = "C:\QTTutorial.pdf"
' ** Initialize Acrobat by creating App object
Set AcroApp = CreateObject( "AcroExch.App" )
' ** show Acrobat
AcroApp.Show()
' ** Set AVDoc object
Set AcroAVDoc = CreateObject( "AcroExch.AVDoc" )
' ** open the PDF
If AcroAVDoc.Open( gPDFPath, "" ) Then
    wait 1
    acroApp.MenuItemExecute ( "ShowHideToolBar" )
    MsgBox "toolbar should be hidden"
    acroApp.MenuItemExecute ( "ShowHideToolBar" )
    wait 1
End If
AcroApp.CloseAllDocs()
AcroApp.Exit()
Set AcroApp = Nothing : Set AcroAVDoc = Nothing
```

AcroExch.App.MenuItemIsEnabled() Method

Description

Determines whether the specified menu item is enabled.

Syntax

```
returnValue = object.MenuItemIsEnabled( menuItemName )
```

Arguments

Parameter	Description
menuItemName	The language-independent name of the menu item whose enabled state is obtained.

Return Value

True if the menu item is enabled, **False** if it is disabled or does not exist.

Related Methods

- `App.MenuItemExecute()`
- `App.MenuItemIsMarked()`
- `App.MenuItemRemove()`

AcroExch.App.MenuItemIsMarked() Method

Description

Determines whether the specified menu item is marked.

Syntax

```
returnValue = object.MenuItemIsMarked( menuItemName )
```

Arguments

Parameter	Description
menuItemName	The language-independent name of the menu item whose marked state is obtained.

Return Value

True if the menu item is enabled, **False** if it is disabled or does not exist.

Related Methods

- `App.MenuItemExecute()`
- `App.MenuItemIsEnabled()`
- `App.MenuItemRemove()`

AcroExch.App.MenuItemRemove() Method

Description

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

Syntax

```
returnValue = object.MenuItemRemove( menuItemName )
```

Arguments

Parameter	Description
menuItemName	The language-independent name of the menu item to remove.

Return Value

True if the menu item was removed, **False** if the menu item does not exist.

Related Methods

- `App.MenuItemExecute()`
- `App.MenuItemIsEnabled()`

- `App.MenuItemsMarked()`

AcroExch.App.SetActiveTool() Method

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).

Syntax

```
returnValue = object.SetActiveTool( buttonName, persistent )
```

Arguments

Parameter	Description
buttonName	The name of the tool to set as the active tool. <ul style="list-style-type: none"> ■ Hand - Hand tool. ■ Note - Tool for making notes. (not in LE 2.0 or Reader) ■ Select - Text selection tool. ■ SelectGraphics - Graphics selection tool. (new in 2.0) ■ Zoom - Tool for changing the zoom factor. ■ Link - Link creation tool. (not in LE 2.0 or Reader) ■ Thread - Thread creation tool. (not in LE 2.0 or Reader)
persistent	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

True if successful, **false** if not.

Related Methods

- `App.GetActiveTool()`
- `App.ToolButtonsEnabled()`
- `App.ToolButtonRemove()`

AcroExch.App.SetFrame() Method

Description

Sets the window's frame to the specified rectangle.

Syntax

```
returnValue = object.SetFrame( iAcroRect )
```

Arguments

Parameter	Description
iAcroRect	An AcroExch.Rect specifying the window frame.

Return Value

true if the frame was set, **false** if iAcroRect is not of type AcroExch.Rect.

Related Methods

- App.GetFrame()
- App.Maximize()

AcroExch.App.SetPreference() Method

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, e.g., 100 is automatically converted to 1.0. Colors (used in avpNoteColor) are automatically converted from RGB values to the representation used in the preferences file

Syntax

```
returnValue = object.SetPreference( nType, nValue )
```

Arguments

Parameter	Description
nType	The preferences item whose value is set. section lists the preferences items.
nValue	The value to set.

Return Value

Always returns **true**.

Related Methods

- App.GetLanguage()
- App.GetPreference()

AcroExch.App.Show() Method

Description

Shows the Acrobat viewer. When the viewer is shown, the user is in control, and the Acrobat viewer does not automatically exit when the last automation object is destroyed.

Syntax

```
returnValue = object.Show()
```

Return Value

Always returns **true**.

Related Methods

- App.Hide()

AcroExch.App.ToolButtonIsEnabled() Method

Description

Is the specified toolbar button enabled?

Syntax

```
returnValue = object.ToolButtonIsEnabled( buttonName )
```

Arguments

Parameter	Description
buttonName	The name of the button whose enabled state is obtained. A list a buttons can be seen in Appendix15, Table2 - Toolbar button names

Return Value

true if the button is enabled, **false** if it is not enabled or does not exist.

Related Methods

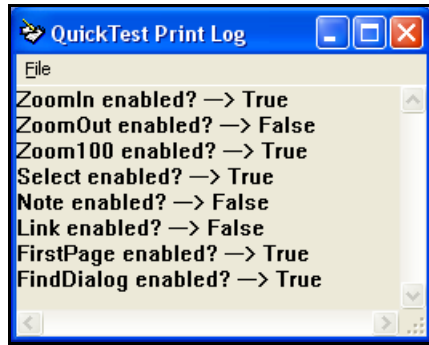
- App.GetActiveTool()
- App.SetActiveTool()
- App.ToolButtonRemove()

Example

```
Option Explicit

Dim AcroApp, AcroAVDoc
Dim gPDFPath

gPDFPath = "C:\QTTutorial.pdf"
' ** Initialize Acrobat by creating App object
Set AcroApp = CreateObject( "AcroExch.App" )
' ** show Acrobat
AcroApp.Show()
' ** Set AVDoc object
Set AcroAVDoc = CreateObject( "AcroExch.AVDoc" )
' ** open the PDF
If AcroAVDoc.Open( gPDFPath, "" ) Then
    Print "ZoomIn enabled? ---> " & acroApp.ToolButtonIsEnabled( "ZoomIn" )
    Print "ZoomOut enabled? ---> " & acroApp.ToolButtonIsEnabled( "ZoomOut" )
    Print "Zoom100 enabled? ---> " & acroApp.ToolButtonIsEnabled( "Zoom100" )
    Print "Select enabled? ---> " & acroApp.ToolButtonIsEnabled( "Select" )
    Print "Note enabled? ---> " & acroApp.ToolButtonIsEnabled( "Note" )
    Print "Link enabled? ---> " & acroApp.ToolButtonIsEnabled( "Link" )
    Print "FirstPage enabled? ---> " & acroApp.ToolButtonIsEnabled( "FirstPage" )
    Print "FindDialog enabled? ---> " & acroApp.ToolButtonIsEnabled( "FindDialog" )
End If
AcroApp.CloseAllDocs()
AcroApp.Exit()
Set AcroApp = Nothing
```



AcroExch.App.ToolButtonRemove() Method

Description

Removes the specified button from the toolbar.

Syntax

```
returnValue = object.ToolButtonRemove( buttonName )
```

Arguments

Parameter	Description
buttonName	The name of the button to remove. A list a buttons can be seen in Appendix15, Table2 - Toolbar button names

Return Value

true if the button was removed, **false** otherwise.

Related Methods

- App.GetActiveTool()
- App.SetActiveTool()
- App.ToolButtonIsEnabled()

AcroExch.App.Unlock() Method

Description

Unlocks the Acrobat viewer 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.

Syntax

```
object.Unlock()
```

Note

- It is strongly recommended that Lock and Unlock be used if you call **OpenInWindow**.

Return Value

Always returns **true**.

Related Methods

- `App.Lock()`

AcroExch.AVDoc Object

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.AVDoc.BringToFront() Method

Description

Brings the window to the front.

Syntax

```
object.BringToFront()
```

Return Value

Always returns **true**.

AcroExch.AVDoc.ClearSelection() Method

Description

Clears the current selection.

Syntax

```
object.ClearSelection()
```

Return Value

true if the selection was cleared, **false** otherwise.

AcroExch.AVDoc.Close() Method

Description

Closes a document. You can close all open AVDocs by calling **App.CloseAllDocs()**.

Syntax

```
object.Close( bNoClose )
```

Arguments

Parameter	Description
bNoClose	If true , the document is closed without saving it. If false and the document has been modified, the user is asked whether or not the file should be saved.

Return Value

false if an error occurred while closing the file, **true** otherwise.

AcroExch.AVDoc.FindText() Method

Description

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

Syntax

```
object.FindText( text, caseSensitive, wholeWordsOnly, reset )
```

Arguments

Parameter	Description
text	The text that is to be found.
caseSensitive	If true , the search is case-sensitive. If false , it is caseinsensitive.
wholeWordsOnly	If true , the search matches only whole words. If false , it matches partial words.
reset	If true , the search begins on the first page of the document. If false , it begins on the current page.

Return Value

true if the text was found, **false** if it was not.

Example

```
Option Explicit

Dim AcroApp, AcroAVDoc
Dim gPDFPath, bReset, nCount

gPDFPath = "C:\QTTutorial.pdf"
' ** Initialize Acrobat by creating App object
Set AcroApp = CreateObject( "AcroExch.App" )
' ** show Acrobat
AcroApp.Show()
' ** Set AVDoc object
Set AcroAVDoc = CreateObject( "AcroExch.AVDoc" )
' ** open the PDF
If AcroAVDoc.Open( gPDFPath, "" ) Then
    AcroAVDoc.BringToFront()
    bReset = True : nCount = 0
    Do While AcroAVDoc.FindText( "Checkpoint", True, True, bReset )
        bReset = False : nCount = nCount + 1
        Wait 0, 200
    Loop
End If
AcroApp.CloseAllDocs()
AcroApp.Exit()
Print "The word 'Checkpoint' was found " & nCount & " times."
Set AcroApp = Nothing
```



AcroExch.AVDoc.GetAVPageView() Method

Description

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

Syntax

```
Set pageView = object.GetAVPageView()
```

Return Value

The **AcroExch.AVPageView**.

AcroExch.AVDoc.GetFrame () Method

Description

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

Syntax

```
Set frame = object.GetFrame()
```

Return Value

An **AcroExch.Rect** containing the frame.

Example

```
Option Explicit


Dim AcroApp, AcroAVDoc, AcroFrame
Dim gPDFPath

gPDFPath = "C:\QTTutorial.pdf"
' ** Initialize Acrobat by creating App object
Set AcroApp = CreateObject( "AcroExch.App" )
' ** show Acrobat
AcroApp.Show()
' ** Set AVDoc object
Set AcroAVDoc = CreateObject( "AcroExch.AVDoc" )
' ** open the PDF
If AcroAVDoc.Open( gPDFPath, "" ) Then
    AcroAVDoc.BringToFront()
    Set AcroFrame = AcroAVDoc.GetFrame()
    Print "Frame Left --> " & AcroFrame.left
    Print "Frame Right --> " & AcroFrame.right
    Print "Frame Top --> " & AcroFrame.top
    Print "Frame Bottom --> " & AcroFrame.bottom
End If
```

```

AcroApp.CloseAllDocs()
AcroApp.Exit()
Set AcroFrame = Nothing
Set AcroApp = Nothing : Set AcroAVDoc = Nothing

```



AcroExch.AVDoc.GetPDDoc () Method

Description

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

Syntax

```
Set avDoc = object.GetPDDoc()
```

Return Value

The **AcroExch.PDDoc**.

AcroExch.AVDoc.GetTitle () Method

Description

Gets the window's title.

Syntax

```
title = object.GetTitle()
```

Return Value

The window's title.

AcroExch.AVDoc.GetViewMode () Method

Description

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

Syntax

```
viewMode = object.GetViewMode()
```

Return Value

The current document view mode. Will be one of the values in Appendix Table 1 – **View Mode Enumeration**

Example

```
Option Explicit
```

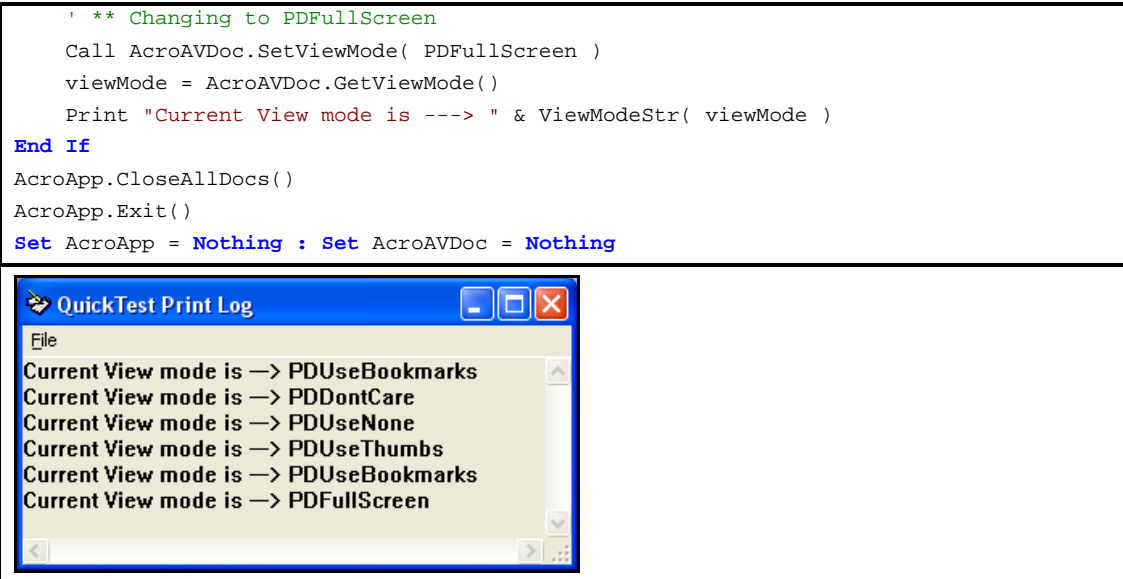
```

Const PDDontCare = 0
Const PDUseNone = 1
Const PDUseThumbs = 2
Const PDUseBookmarks = 3
Const PDFFullScreen = 4

Dim AcroApp, AcroAVDoc
Dim gPDFPath, viewMode
** This function translates contants to strings.
Private Function ViewModeStr( ByVal viewMode )
    Select Case viewMode
        Case PDDontCare : ViewModeStr = "PDDontCare"
        Case PDUseNone : ViewModeStr = "PDUseNone"
        Case PDUseThumbs : ViewModeStr = "PDUseThumbs"
        Case PDUseBookmarks : ViewModeStr = "PDUseBookmarks"
        Case PDFFullScreen : ViewModeStr = "PDFFullScreen"
    End Select
End Function

gPDFPath = "C:\QTTutorial.pdf"
' ** Initialize Acrobat by creating App object
Set AcroApp = CreateObject( "AcroExch.App" )
' ** show Acrobat
AcroApp.Show()
' ** Set AVDoc object
Set AcroAVDoc = CreateObject( "AcroExch.AVDoc" )
' ** open the PDF
If AcroAVDoc.Open( gPDFPath, "" ) Then
    If AcroAVDoc.IsValid = False Then ExitTest()
    AcroAVDoc.BringToFront()
    viewMode = AcroAVDoc.GetViewMode
    Print "Current View mode is ---> " & ViewModeStr( viewMode )
    ' ** Changing to PDDontCare
    Call AcroAVDoc.SetViewMode( PDDontCare )
    viewMode = AcroAVDoc.GetViewMode()
    Print "Current View mode is ---> " & ViewModeStr( viewMode )
    Wait 0, 500
    ' ** Changing to PDUseNone
    Call AcroAVDoc.SetViewMode( PDUseNone )
    viewMode = AcroAVDoc.GetViewMode()
    Print "Current View mode is ---> " & ViewModeStr( viewMode )
    Wait 0, 500
    ' ** Changing to PDUseThumbs
    Call AcroAVDoc.SetViewMode( PDUseThumbs )
    viewMode = AcroAVDoc.GetViewMode()
    Print "Current View mode is ---> " & ViewModeStr( viewMode )
    Wait 0, 500
    ' ** Changing to PDUseBookmarks
    Call AcroAVDoc.SetViewMode( PDUseBookmarks )
    viewMode = AcroAVDoc.GetViewMode()
    Print "Current View mode is ---> " & ViewModeStr( viewMode )
    Wait 0, 500

```



AcroExch.AVDoc.IsValid () Method

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.

Syntax

```
returnValue = object.IsValid()
```

Return Value

true if the document can still be used, **false** otherwise

AcroExch.AVDoc.Maximize() Method

Description

Maximizes the window if bMaxSize is true.

Syntax

```
object.Maximize( bMaximize )
```

Arguments

Parameter	Description
bMaximize	Indicates whether window should be maximized.

Return Value

Always returns **true**.

AcroExch.AVDoc.Open() Method

Description

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

Syntax

```
returnValue = object.Open( fullPath, tempTitle )
```

Arguments

Parameter	Description
<i>fullPath</i>	The full pathname of the file to open.
<i>tempTitle</i>	An optional title for the window in which the file is opened. If tempTitle is Empty or the empty string (""), it is ignored. Otherwise, tempTitle is used as the window title.

Note

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

Return Value

Always returns **true**.

Example

```
Option Explicit

Dim AcroApp, AcroAVDoc
Dim gPDFPath

gPDFPath = "C:\QTTutorial.pdf"
' ** Initialize Acrobat by creating App object
Set AcroApp = CreateObject( "AcroExch.App" )
' ** show Acrobat
AcroApp.Show()
' ** Set AVDoc object
Set AcroAVDoc = CreateObject( "AcroExch.AVDoc" )
' ** open the PDF
If AcroAVDoc.Open( gPDFPath, "NewTitle" ) Then
    If AcroAVDoc.IsValid = False Then ExitTest()
    AcroAVDoc.BringToFront()
End If
AcroApp.CloseAllDocs()
AcroApp.Exit()
Set AcroApp = Nothing : Set AcroAVDoc = Nothing
```

AcroExch.AVDoc.OpenInWindowEx() Method

Description

Syntax

Arguments

Parameter	Description

Return Value

AcroExch.AVDoc.PrintPages() and PrintPagesSilent() Methods

Description

Prints a specified range of pages, displaying a print dialog box. `PrintPages` always uses the default printer setting in WIN.INI.

PrintPagesSilent Prints a specified range of pages without displaying any dialog box.

Syntax

```
object.Open( firstPage, lastPage, psLevel, binaryOK, shrinkToFit )
```

Arguments

Parameter	Description
<i>firstPage</i>	The first page to print. The first page in a PDDoc is page 0.
<i>lastPage</i>	The last page to print. The first page in a PDDoc is page 0.
<i>psLevel</i>	If 1, PostScript Level 1 operators are used. If 2, PostScript Level 2 operators are also used.
<i>binaryOK</i>	If true , binary data may be included in the PostScript program. If false , all data is encoded as 7-bit ASCII.
<i>shrinkToFit</i>	If true, the page is shrunk (if necessary) to fit within the imageable area of the printed page. If false, it is not.

Return Value

false if there were any exceptions while printing, **true** otherwise.

AcroExch.AVDoc.SetFrame() Method

Description

Sets the window's size and location.

Syntax

```
Object.SetFrame( acroRect )
```

Arguments

Parameter	Description
<i>acroRect</i>	AcroExch.Rect specifying the window's frame.

Return Value

Always returns true.

AcroExch.AVDoc.SetTextSelection() Method

Description

Sets the document's selection to the specified text selection.

Syntax

```
Object.SetTextSelection( textSelect )
```

Arguments

Parameter	Description
<i>textSelect</i>	The text selection to use.

Note

- 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.

Return Value

Always returns **true**.

AcroExch.AVDoc.SetTitle() Method

Description

Sets the window's title.

Syntax

```
Object.SetTitle( title )
```

Arguments

Parameter	Description
<i>title</i>	The title to set. This method cannot be used on document windows, but only on windows created by plug-ins.

Return Value

Always returns **true**.

AcroExch.AVDoc.SetViewMode() Method

Description

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

Syntax

```
Object.SetViewMode( viewMode )
```


Arguments

Parameter	Description
<i>viewMode</i>	The view mode to set. Must be one of the values listed in Appendix Table 1 – View Mode Enumeration

Return Value

Always returns **true**.

Example

See Example [AcroExch.AVDoc.GetViewMode \(\) Method](#)

AcroExch.AVDoc.ShowTextSelect () Method

Description

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

Syntax

```
object.ShowTextSelect()
```

Return Value

Always returns **true**.

AcroExch.AVPageView Object

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 **PDPPage** objects for the document being displayed.

```
Option Explicit

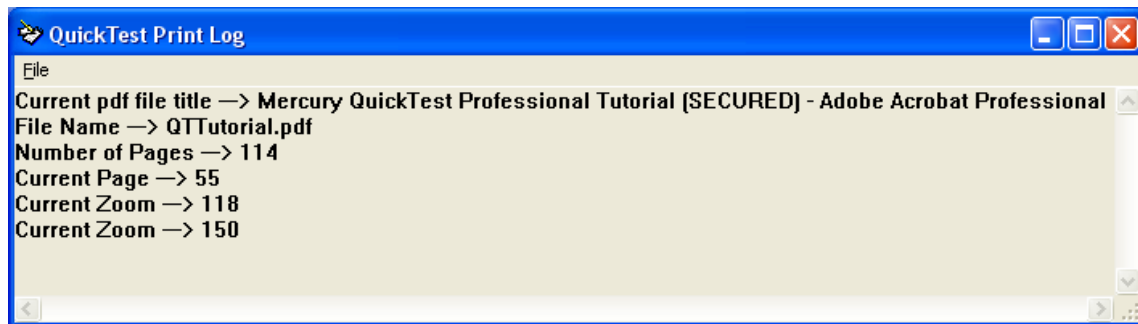
Const AVZoomNoVary = 0
Const AVZoomFitVisibleWidth = 4
Dim AcroApp, AcroAVDoc, AcroAvPageView, AcroPDDoc
Dim gPDFPath

gPDFPath = "C:\QTTutorial.pdf"
' ** Initialize Acrobat by creating App object
Set AcroApp = CreateObject( "AcroExch.App" )
' ** show Acrobat
AcroApp.Show()
' ** Set AVDoc object
Set AcroAVDoc = CreateObject( "AcroExch.AVDoc" )
' ** open the PDF
If AcroAVDoc.Open( gPDFPath, "" ) Then
    If AcroAVDoc.IsValid = False Then ExitTest()
    AcroAVDoc.BringToFront()
    Call AcroAVDoc.Maximize( True )
    Print "Current pdf file title ---> " & AcroAVDoc.GetTitle()
    Set AcroPDDoc = AcroAVDoc.GetPDDoc()
```

```

Print "File Name ---> " & AcroPDDoc.GetFileName()
Print "Number of Pages ---> " & AcroPDDoc.GetNumPages()
Set AcroAvPageView = AcroAVDoc.GetAVPageView()
Call AcroAvPageView.Goto( 55 )
Print "Current Page ---> " & AcroAvPageView.GetPageNum()
Print "Current Zoom ---> " & AcroAvPageView.GetZoom()
Call AcroAvPageView.ZoomTo( AVZoomNoVary, 150 )
Print "Current Zoom ---> " & AcroAvPageView.GetZoom()
AcroAvPageView.ReadPageDown()
Wait 0, 500
Call AcroAvPageView.ZoomTo( AVZoomFitVisibleWidth, 50 )
End If
AcroApp.CloseAllDocs()
AcroApp.Exit()
Set AcroApp = Nothing : Set AcroAVDoc = Nothing

```



AcroExch.AVPageView.DevicePointToPage () Method

Description

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

Syntax

```
object.DevicePointToPage( acroPoint )
```

Arguments

Parameter	Description
<i>acroPoint</i>	The AcroExch.Point whose coordinates are converted.

Return Value

AcroExch.Point containing the converted coordinates.

Related Methods

- AVPageView.PointToDevice()

AcroExch.AVPageView.DoGoBack () Method

Description

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

Syntax

```
object.DoGoBack()
```

Return Value

Always returns **true**.

Related Methods

- AVPageView.DoGoForward()

AcroExch.AVPageView.DoGoForward() Method

Description

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

Syntax

```
object.DoGoForward()
```

Return Value

Always returns **true**.

Related Methods

- AVPageView.DoGoBack()

AcroExch.AVPageView.GetAVDoc() Method

Description

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

Syntax

```
Set avDoc = object.GetAVDoc()
```

Return Value

The **AcroExch.AVDoc**.

Related Methods

- AVDoc.GetAVPageView
- AVDoc.GetPDDoc
- AVPageView.GetAVDoc

AcroExch.AVPageView.GetDoc() Method

Description

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

Syntax

```
Set pdDoc = object.GetDoc()
```

Return Value

The **AcroExch.PDDoc**.

Related Methods

- AVDoc.**GetAVPageView**
- AVDoc.**GetPDDoc**
- AVPageView.**GetAVDoc**

AcroExch.AVPageView.GetPage() Method

Description

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

Syntax

```
Set pdPage = object.GetPage()
```

Return Value

The **AcroExch.PDPage**.

Related Methods

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

AcroExch.AVPageView.GetPageNum() Method

Description

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

Syntax

```
pageNum = object.GetPageNum()
```

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**

AcroExch.AVPageView.GetZoom() Method

Description

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

Syntax

```
zoom = object.GetZoom()
```

Return Value

The current zoom factor.

Related Methods

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

AcroExch.AVPageView.GetZoomType() Method

Description

Gets the current zoom type.

Syntax

```
zoomType = object.GetZoomType()
```

Return Value

One of the zoom strategies types on Appendix 16 - **Zoom Strategy**

Related Methods

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

AcroExch.AVPageView.Goto() Method

Description

Goes to the specified page.

Syntax

```
Object.Goto( nPage )
```

Arguments

Parameter	Description
<i>nPage</i>	Page number of the destination page. First page in a PDDoc is page 0.

Return Value

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

Related Methods

- AVPageView.DoGoBack
- AVPageView.DoGoForward
- AVPageView.ReadPageDown
- AVPageView.ReadPageUp
- AVPageView.ScrollTo
- AVPageView.ZoomTo

AcroExch.AVPageView.ReadPageDown() and ReadPageUp() Methods

Description

Scrolls **forward/backward** through the document by “one screenfull”

Syntax

```
object.ReadPageDown()
object.ReadPageUp()
```

Return Value

Always returns true.

Related Methods

- AVPageView.DoGoBack
- AVPageView.DoGoForward
- AVPageView.Goto
- AVPageView.ScrollTo
- AVPageView.ZoomTo

AcroExch.AVPageView.ScrollTo() Method

Description

Scrolls to the specified location on the current page.

Syntax

```
returnValue = Object.ScrollTo( nX, nY )
```

Arguments

Parameter	Description
<i>nX</i>	x-coordinate of the destination.
<i>nY</i>	y-coordinate of the destination.

Return Value

true if the Acrobat viewer successfully scrolled to the specified location, **false** otherwise.

Related Methods

- AVPageView.DoGoBack

- AVPageView.**DoGoForward**
- AVPageView.**Goto**
- AVPageView.**ReadPageDown**
- AVPageView.**ReadPageUp**
- AVPageView.**ZoomTo**

AcroExch.AVPageView.ZoomTo() Method

Description

Zooms to a specified magnification.

Syntax

```
Object.ZoomTo( nType, nScale )
```

Arguments

Parameter	Description
<i>nType</i>	Zoom type. On of the Zoom Strategy
<i>nScale</i>	The desired zoom factor, expressed as a percent, e.g., 100 is a magnification of 1.0

Return Value

true if the magnification was set successfully, **false** otherwise.

Related Methods

- AVPageView.**GetZoomType**
- AVPageView.**Goto**
- AVPageView.**ScrollTo**

AcroExch.HiliteList Object

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.HiliteList.Add() Method

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 contains character or word highlights) to create a text selection from the highlight list.

Syntax

```
Object.Add( nOffset, nLength )
```

Arguments

Parameter	Description
<i>nOffset</i>	Offset of the first word or character to highlight. The first word/character on a page has an offset of zero.
<i>nLength</i>	The number of consecutive words or characters to highlight.

Return Value

Always returns **true**.

Related Methods

- `PDPage.CreatePageHilite`
- `PDPage.CreateWordHilite`

AcroExch.PDAnnot Object

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 Object

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.PDBookmark.Destroy () Method

Description

Destroys a bookmark. Note that it is not possible to create a bookmark with OLE.

Syntax

```
returnValue = Object.Destroy()
```

Return Value

false if the Acrobat viewer does not support editing (making it impossible to delete the bookmark), **true** otherwise.

Related Methods

- `PDBookmark.IsValid`

AcroExch.PDBookmark.GetByTitle() Method

Description

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

Syntax

```
returnValue = Object.GetByTitle( pdDoc, bookmarkTitle )
```

Arguments

Parameter	Description
<i>pdDoc</i>	The document (AcroExch.PDDoc object) containing the bookmark.
<i>bookmarkTitle</i>	The title of the bookmark to get. The capitalization of the title must match that in the bookmark.

Return Value

true if the specified bookmark exists (the method determines this using the **PDBookmark.IsValid()** method), **false** otherwise.

Related Methods

- **PDBookmark.GetTitle**
- **PDBookmark.SetTitle**

Example

```
Option Explicit
Private Const PDUseBookmarks = 3
Dim acroApp, avDoc, pdDoc, pdBookmark
Dim gPDFPath

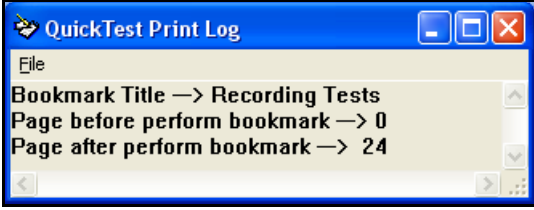
gPDFPath = "C:\QTTutorial.pdf"
' ** Initialize Acrobat by creating App object
Set acroApp = CreateObject( "AcroExch.App" )

' ** Set AVDoc object
Set avDoc = CreateObject( "AcroExch.AVDoc" )
' ** open the PDF
If avDoc.Open( gPDFPath, "Accessing PDF's" ) Then
    If avDoc.IsValid = False Then ExitTest()
    With acroApp
        .Show()
        .Maximize( True )
    End With
    avDoc.BringToFront()
    ' ** Setting view mode to bookmarks...
    Call avDoc.SetViewMode( PDUseBookmarks )
    Set pdBookmark = CreateObject( "AcroExch.PDBookmark" )
    If pdBookmark.GetByTitle( avDoc.GetPDDoc(), "Recording Tests" ) Then
        Print "Bookmark Title ----> " & pdBookmark.GetTitle()
        Print "Page before perform bookmark ----> " & avDoc.GetAVPageView.GetPageNum()
```

```

    Call pdBookmark.Perform( avDoc )
    Print "Page after perform bookmark ---> " & avDoc.GetAVPageView.GetPageNum()
End If
End If
AcroApp.CloseAllDocs()
AcroApp.Exit()
Set pdBookmark = Nothing
Set pdPage = Nothing : Set AcroApp = Nothing : Set avDoc = Nothing

```



AcroExch.PDBookmark.GetTitle() Method

Description

Gets a bookmark's title (up to 256 characters).

Syntax

```
returnValue = Object.GetTitle()
```

Return Value

The title.

Related Methods

- PDBookmark.GetByTitle
- PDBookmark.SetTitle

AcroExch.PDBookmark.IsValid() Method

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.

Syntax

```
returnValue = Object.IsValid()
```

Return Value

true if the bookmark is valid, **false** otherwise.

Related Methods

- PDBookmark.Destroy

AcroExch.PDBookmark.Perform() Method

Description

Performs a bookmark's action.

Syntax

```
returnValue = Object.Perform( avDoc )
```

Arguments

Parameter	Description
<i>avDoc</i>	AcroExch.AVDoc in which the bookmark is located.

Return Value

true if the action was executed successfully, **false** otherwise.

Related Methods

- **PDBookmark.IsValid**

AcroExch.PDBookmark.SetTitle() Method

Description

Sets a bookmark's title.

Syntax

```
returnValue = Object.SetTitle( newTitle )
```

Arguments

Parameter	Description
<i>newTitle</i>	The title to set.

Return Value

false if the Acrobat viewer does not support editing, **true** otherwise.

Related Methods

- **PDBookmark.GetByTitle**
- **PDBookmark.GetTitle**

AcroExch.PDDoc Object

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.PDDoc.AcquirePage () Method

Description

Acquires the specified page.

Syntax

```
Set pdPage = Object.AcquirePage( nPage )
```

Arguments

Parameter	Description
<i>nPage</i>	The number of the page to acquire. The first page in a PDDoc is page 0.

Return Value

true if the Acrobat viewer successfully scrolled to the specified location, **false** otherwise.

Related Methods

- AVPageView.**GetPage**
- AVPageView.**GetPageNum**
- PDDoc.**GetNumPages**
- PDPage.**GetDoc**
- PDPage.**GetNumber**
- PDPage.**GetRotate**
- PDPage.**GetSize**
- PDTextSelect.**GetPage**

AcroExch.PDDoc.ClearFlags () Method

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.

Syntax

```
Object.ClearFlags( nFlags )
```

Arguments

Parameter	Description
<i>nFlags</i>	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 **true**.

Related Methods

- PDDoc.**GetFlags**
- PDDoc.**SetFlags**

AcroExch.PDDoc.Close () Method

Description

Closes a file.

Syntax

```
Object.Close()
```

Note

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

Return Value

true if the document was closed successfully, **false** otherwise.

Related Methods

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

AcroExch.PDDoc.Create () Method

Description

Creates a new **AcroExch.PDDoc**.

Syntax

```
Set pdDoc = Object.Create()
```

Return Value

true if the document is created successfully, **false** if it is not or if the Acrobat viewer does not support editing.

AcroExch.PDDoc.CreateTextSelect () Method

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.

Syntax

```
Set pdDoc = Object.CreateTextSelect( nPage, acroRect )
```

Arguments

Parameter	Description
<i>nPage</i>	The page on which the selection is created. The first page in a PDDoc

	object is page 0.
<i>acroRect</i>	The AcroExch.Rect enclosing the region to select.

Return Value

An **AcroExch.PDTextSelect** containing the text selection. Returns **Nothing** 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**

Example

See Example in **AcroExch.PDTextSelect Object**

AcroExch.PDDoc.CreateThumbs () Method

Description

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

Syntax

```
returnValue = Object.CreateThumbs( nFirstPage, nLastPage )
```

Arguments

Parameter	Description
<i>nFirstPage</i>	First page for which thumbnail images are created.
<i>nLastPage</i>	Last page for which thumbnail images are created.

Return Value

true if thumbnail images were created successfully, **false** if they were not or if the Acrobat viewer does not support editing.

Related Methods

- AVDoc.**DeleteThumbs**

AcroExch.PDDoc.CropPages () Method

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).

Syntax

```
returnValue = _
    Object.CropPages( nStartPage, nEndPage, nEvenOrOddPagesOnly, acroRect )
```

Arguments

Parameter	Description
<i>nStartPage</i>	First page that is cropped. The first page in a PDDoc object is page 0.
<i>nEndPage</i>	Last page that is cropped.
<i>nEvenOrOddPagesOnly</i>	Value indicating which pages in the range are cropped. Must be one of the following: <ul style="list-style-type: none"> ■ 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
<i>acroRect</i>	An AcroExch.Rect specifying the cropping rectangle, which is specified in user space.

Return Value

true if the pages were cropped successfully, **false** otherwise.

Related Methods

- PDDoc.CropPages

AcroExch.PDDoc.DeletePages () Method

Description

Deletes pages from a file.

Syntax

```
returnValue = Object.DeletePages( nStartPage, nEndPage )
```

Arguments

Parameter	Description
<i>nStartPage</i>	The first page to be deleted. The first page in a PDDoc object is page 0.
<i>nEndPage</i>	The last page to be deleted.

Return Value

true if the pages were successfully deleted. Returns **false** if they were not or if the Acrobat viewer does not support editing.

Related Methods

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

AcroExch.PDDoc.DeleteThumbs () Method

Description

Deletes thumbnail images from the specified pages in a document.

Syntax

```
returnValue = Object.DeleteThumbs( nStartPage, nEndPage )
```

Arguments

Parameter	Description
<i>nStartPage</i>	First page whose thumbnail image is deleted. The first page in a PDDoc object is page 0.
<i>nEndPage</i>	Last page whose thumbnail image is deleted.

Return Value

true if the thumbnails were deleted, **false** if they were not deleted or if the Acrobat viewer does not support editing.

Related Methods

- **PDDoc.CreateThumbs**

AcroExch.PDDoc.GetFileName () Method

Description

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

Syntax

```
returnValue = Object.GetFileName( )
```

Return Value

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

Related Methods

- **PDDoc.Save**

AcroExch.PDDoc.GetFlags () Method

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.

Syntax

```
returnValue = Object.GetFlags( )
```

Return Value

A combination of document's flags. See Appendix **Document Flags Enumeration**

Related Methods

- PDDoc.ClearFlags
- PDDoc.SetFlags

AcroExch.PDDoc.GetInfo () Method

Description

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

Syntax

```
returnValue = Object.GetInfo( infoKey )
```

Arguments

Parameter	Description
<i>infoKey</i>	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

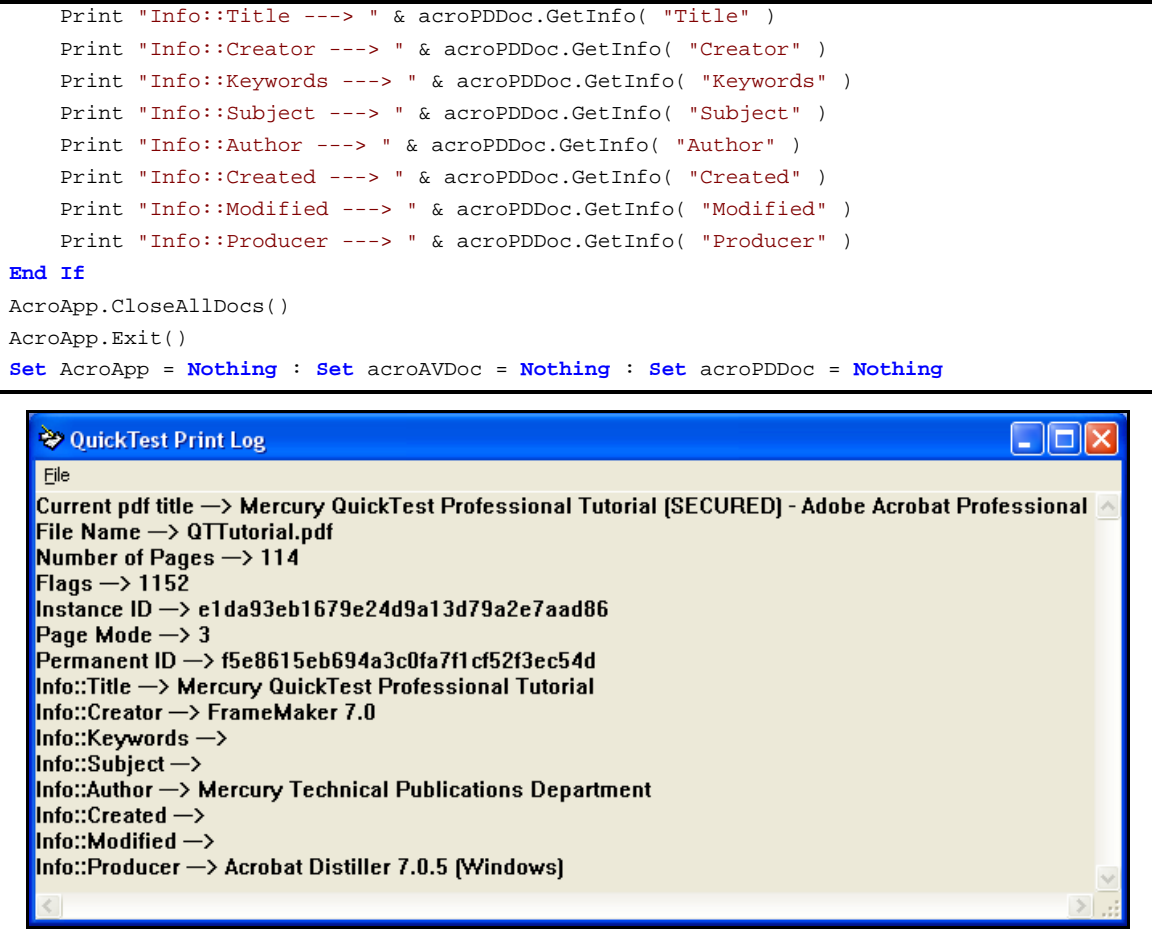
- PDDoc.SetInfo

Example

```
Option Explicit

Dim acroApp, acroAVDoc, acroPDDoc
Dim gPDFPath

gPDFPath = "C:\QTTutorial.pdf"
' ** Initialize Acrobat by creating App object
Set acroApp = CreateObject( "AcroExch.App" )
' ** show acrobat
acroApp.Show()
' ** Set AVDoc object
Set acroAVDoc = CreateObject( "AcroExch.AVDoc" )
' ** open the PDF
If acroAVDoc.Open( gPDFPath, Empty ) Then
    If acroAVDoc.IsValid = False Then ExitTest()
    acroAVDoc.BringToFront()
    Call acroAVDoc.Maximize( True )
    Print "Current pdf title ---> " & acroAVDoc.GetTitle()
    Set acroPDDoc = acroAVDoc.GetPDDoc()
    Print "File Name ---> " & acroPDDoc.GetFileName()
    Print "Number of Pages ---> " & acroPDDoc.GetNumPages()
    Print "Flags ---> " & acroPDDoc.GetFlags()
    Print "Instance ID ---> " & acroPDDoc.GetInstanceID()
    Print "Page Mode ---> " & acroPDDoc.GetPageMode()
    Print "Permanent ID ---> " & acroPDDoc.GetPermanentID()
```



AcroExch.PDDoc.GetInstanceID () Method

Description

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

Syntax

```
returnValue = Object.GetInstanceID()
```

Return Value

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

Related Methods

- PDDoc.GetPermanentID

AcroExch.PDDoc.GetJSObject () Method

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. At

<http://www.adobe.com/devnet/acrobat/pdfs/VBJavaScript.pdf>

Syntax

```
Set jso = Object.GetJSObject()
```

Return Value

The interface to the JavaScript object if the call succeeded, **Nothing** otherwise.

Example

Acrobat 7.0 includes a plug-in that can scan a document for spelling errors. This plug-in also provides JavaScript methods that can be accessed using a **JSObject**.

Option Explicit

```
Dim pdDoc, jso
Dim path, word, result
Dim i, j, foundErr

path = "C:\QTTutorial.pdf"
foundErr = False
Set pdDoc = CreateObject( "AcroExch.PDDoc" )
If pdDoc.Open( path ) Then
    Set jso = pdDoc.GetJSObject
    If Not jso Is Nothing Then
        count = jso.getPageNumWords( 42 )
        For i = 0 To count - 1
            word = jso.getPageNthWord( 42, i, true )
            If VarType( word ) = vbString Then
                result = jso.spell.checkWord( word )
                If IsArray(result) Then
                    foundErr = True
                    Print word & " is misspelled."
                    Print "Suggestions:"
                    For j = LBound( result ) To UBound( result )
                        Print result( j )
                    Next
                    Print String( 90, "-" )
                End If
            End If
        Next
        Set jso = Nothing
        pdDoc.Close
        If Not foundErr Then
            Print "No spelling errors found in " & path
        End If
    End If
Else
    Print "Failed to open " & path
End If
Set pdDoc = Nothing
```

AcroExch.PDDoc.GetNumPages () Method

Description

Gets the number of pages in a file.

Syntax

```
returnValue = Object.GetNumPages()
```

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**

AcroExch.PDDoc.GetPageMode () Method

Description

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

Syntax

```
returnValue = Object.GetPageMode()
```

Return Value

The current page mode. Will be one of the values in Appendix 16 - **View Mode Enumeration**

Related Methods

- AVPageView.**SetPageMode**

AcroExch.PDDoc.GetPermanentID () Method

Description

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

Syntax

```
returnValue = Object.GetPermanentID()
```

Return Value

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

Related Methods

- AVPageView.**GetInstanceID**

AcroExch.PDDoc.InsertPages () Method

Description

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

AcroExch.PDDoc.MovePage () Method

Description

Moves a page to another location within the same document.

AcroExch.PDDoc.Open () Method

Description

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

Syntax

```
returnValue = Object.Open( fullPath )
```

Arguments

Parameter	Description
<i>fullPath</i>	Full pathname of the file to be opened.

Return Value

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

Related Methods

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

AcroExch.PDDoc.OpenAVDoc () Method

Description

Opens a window and displays the document in it.

Syntax

```
Set avDoc = Object.OpenAVDoc( fullPath )
```

Arguments

Parameter	Description
<i>title</i>	The title to be used for the window. A default title is used if title is Empty , or an empty string.

Return Value

The **AcroExch.AVDoc** that was opened, or **Nothing** if the open fails.

Related Methods

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

AcroExch.PDDoc.ReplacePages () Method

Description

Replaces the indicated pages in the current document with those specified from the source document.

AcroExch.PDDoc.Save () Method

Description

Opens a window and displays the document in it.

Syntax

```
returnValue = Object.Save( nType, fullPath )
```

Arguments

Parameter	Description
<i>nType</i>	<p>Specifies the way in which the file should be saved.</p> <ul style="list-style-type: none"> ■ 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.
<i>fullPath</i>	The new pathname to the file, if any.

Return Value

true if the document was successfully saved. Returns **false** if it was not or if the Acrobat application does not support editing.

Related Methods

- `App.GetFileName`

AcroExch.PDDoc.SetFlags () Method

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.

AcroExch.PDDoc.SetInfo () Method

Description

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

AcroExch.PDDoc.SetPageMode () Method

Description

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

Syntax

```
Object.SetPageMode( pageMode )
```

Arguments

Parameter	Description
<i>pageMode</i>	The page mode to be set. Must be one of the values of View Mode Enumeration

Return Value

Always returns **true**

Related Methods

- `PDDoc.GetPageMode`
- `PDDoc.SetPageMode`

AcroExch.PDPage Object

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.PDPage.AddAnnot () Method

Description

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

Syntax

```
returnValue = Object.AddAnnot( nfterIndex, pdAnnot )
```

Arguments

Parameter	Description
<i>nAfterIndex</i>	Location in the page's annotation array to add the annotation. The first annotation on a page has an index of zero.
<i>pdAnnot</i>	AcroExch.PDAnnot to add.

Return Value

false if the Acrobat viewer does not support editing, **true** otherwise.

Related Methods

- **PDPage.AddNewAnnot**
- **PDPage.RemoveAnnot**

AcroExch.PDPage.AddNewAnnot () Method

Description

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

Syntax

```
Set pdAnnot = Object.AddNewAnnot( nfterIndex, subTypeem pdAnnot )
```

Arguments

Parameter	Description
<i>nAfterIndex</i>	Location in the page's annotation array to add the annotation. The first annotation on a page has an index of zero.
<i>subType</i>	Subtype of the annotation to create. Must be Text.
<i>pdAnnot</i>	AcroExch.PDAnnot to add.

Return Value

An **AcroExch.PDAnnot** object, or **Nothing** if the annotation could not be added.

Related Methods

- **PDPage.AddAnnot()**
- **PDPage.RemoveAnnot()**

AcroExch.PDPage.CreatePageHilite () Method

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()**.

Syntax


```
Set acroTextSelect = Object.CreatePageHilite( acroHiliteList )
```

Arguments

Parameter	Description
<i>acroHiliteList</i>	Highlight list for which a text selection is created. Use HiliteList.Add() to create a highlight list.

Return Value

The **AcroExch.PDTextSelect** containing the text selection, or **Nothing** 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**

Example

```
Option Explicit

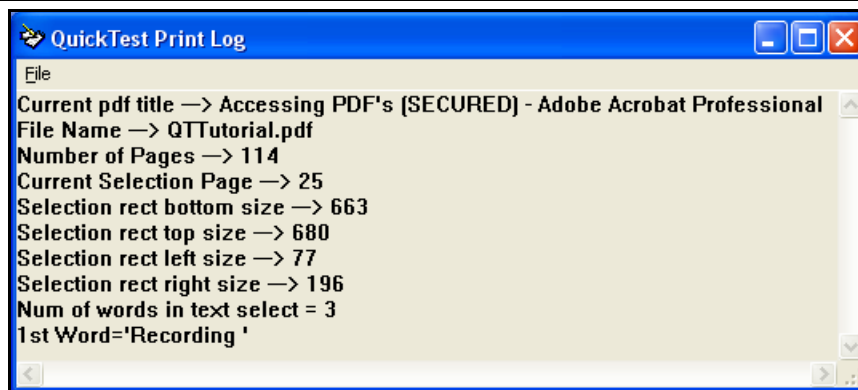
Dim acroApp, acroAVDoc, acroPDDoc, acroRect, acroPDPage, acroPageView
Dim acroHiList, acroPDTextSel
Dim gPDFPath, nElem

gPDFPath = "C:\QTTutorial.pdf"
' ** Initialize Acrobat by creating App object
Set acroApp = CreateObject( "AcroExch.App" )
' ** show acrobat
acroApp.Show()
' ** Set AVDoc object
Set acroAVDoc = CreateObject( "AcroExch.AVDoc" )
' ** open the PDF
If acroAVDoc.Open( gPDFPath, "Accessing PDF's" ) Then
    If acroAVDoc.IsValid = False Then ExitTest()
    acroAVDoc.BringToFront()
    Call acroAVDoc.Maximize( True )
    Print "Current pdf title ---> " & acroAVDoc.GetTitle()
    Set acroPDDoc = acroAVDoc.GetPDDoc()
    Print "File Name ---> " & acroPDDoc.GetFileName()
    Print "Number of Pages ---> " & acroPDDoc.GetNumPages()
    ' ** Create a list object
    Set acroHiList = CreateObject( "AcroExch.HiliteList" )
    ' ** Create a hilite that includes all possible text on the page
```

```

Call acroHiList.Add( 30, 11 )
' ** Create either a page or word hilite based on the list
Set acroPageView = acroAVDoc.GetAVPageView()
Call acroPageView.Goto( 25 )
Set acroPDPage = acroPageView.GetPage()
Set acroPDTextSel = acroPDPage.CreatePageHilite( acroHiList )
If acroPDTextSel Is Nothing Then
    Print "Failed to CreatePageHilite."
    ExitTest()
End If
' ** Set that as the current text selection and show it
Call acroAVDoc.SetTextSelection( acroPDTextSel )
Call acroAVDoc.ShowTextSelect()
' ** Get the number of words in the text selection and the first word in selection
If acroPDTextSel.GetNumText > 0 Then
    Print "Current Selection Page ---> " & acroPDTextSel.GetPage()
    Print "Selection rect bottom size ---> " & acroPDTextSel.GetBoundingRect.bottom
    Print "Selection rect top size ---> " & acroPDTextSel.GetBoundingRect.top
    Print "Selection rect left size ---> " & acroPDTextSel.GetBoundingRect.left
    Print "Selection rect right size ---> " & acroPDTextSel.GetBoundingRect.right
    Print "Num of words in text select = " & acroPDTextSel.GetNumText
    Print "1st Word='" & acroPDTextSel.GetText( 0 ) & "'"
Else
    Print "There are no words in the text selection"
End If
Call acroPDTextSel.Destroy()
End If
AcroApp.CloseAllDocs()
AcroApp.Exit()
Set acroPDTextSel = Nothing : Set acroRect = Nothing
Set acroHiList = Nothing : Set acroPageView = Nothing : Set acroPDPage = Nothing
Set AcroApp = Nothing : Set AcroAVDoc = Nothing

```



AcroExch.PDPage.CreateWordHilite () Method

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()**.

Syntax

```
Set acroTextSelect = Object.CreateWordHilite( acroHiliteList )
```

Arguments

Parameter	Description
<i>acroHiliteList</i>	Highlight list for which a text selection is created. Use HiliteList.Add() to create a highlight list.

Return Value

The **AcroExch.PDTextSelect**, or **Nothing** 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**

Example

```
Option Explicit

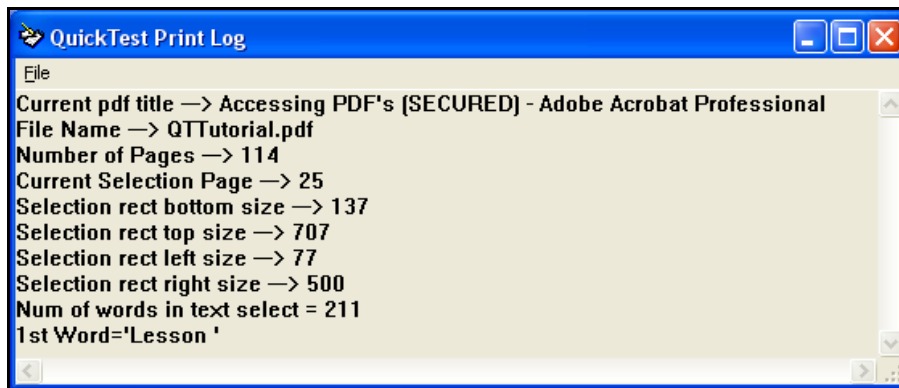
Dim acroApp, acroAVDoc, acroPDDoc, acroRect, acroPDPage, acroPageView
Dim acroHiList, acroPDTextSel
Dim gPDFPath, nElem

gPDFPath = "C:\QTTutorial.pdf"
' ** Initialize Acrobat by creating App object
Set acroApp = CreateObject( "AcroExch.App" )
' ** show acrobat
acroApp.Show()
' ** Set AVDoc object
Set acroAVDoc = CreateObject( "AcroExch.AVDoc" )
' ** open the PDF
If acroAVDoc.Open( gPDFPath, "Accessing PDF's" ) Then
    If acroAVDoc.IsValid = False Then ExitTest()
    acroAVDoc.BringToFront()
    Call acroAVDoc.Maximize( True )
    Print "Current pdf title ---> " & acroAVDoc.GetTitle()
    Set acroPDDoc = acroAVDoc.GetPDDoc()
    Print "File Name ---> " & acroPDDoc.GetFileName()
    Print "Number of Pages ---> " & acroPDDoc.GetNumPages()
    ' ** Create a list object
    Set acroHiList = CreateObject( "AcroExch.HiliteList" )
```

```

' ** Create a hilite that includes all possible text on the page
CallacroHiList.Add( 0, 32767 )
' ** Create either a page or word hilite based on the list
SetacroPageView =acroAVDoc.GetAVPageView()
CallacroPageView.Goto( 25 )
SetacroPDPPage =acroPageView.GetPage()
SetacroPDTextSel =acroPDPPage.CreateWordHilite(acroHiList )
IfacroPDTextSel Is Nothing Then
    Print "Failed to CreateWordHilite."
    ExitTest()
End If
' ** Set that as the current text selection and show it
CallacroAVDoc.SetTextSelection(acroPDTextSel )
CallacroAVDoc.ShowTextSelect()
' ** Get the number of words in the text selection and the first word in selection
IfacroPDTextSel.GetNumText > 0 Then
    Print "Current Selection Page ---> " &acroPDTextSel.GetPage()
    Print "Selection rect bottom size ---> " &acroPDTextSel.GetBoundingRect.bottom
    Print "Selection rect top size ---> " &acroPDTextSel.GetBoundingRect.top
    Print "Selection rect left size ---> " &acroPDTextSel.GetBoundingRect.left
    Print "Selection rect right size ---> " &acroPDTextSel.GetBoundingRect.right
    Print "Num of words in text select = " &acroPDTextSel.GetNumText
    Print "1st Word=' " &acroPDTextSel.GetText( 0 ) & "' "
Else
    Print "There are no words in the text selection"
End If
CallacroPDTextSel.Destroy()
End If
AcroApp.CloseAllDocs()
AcroApp.Exit()
SetacroPDTextSel = Nothing : SetacroRect = Nothing
SetacroHiList = Nothing : SetacroPageView = Nothing : SetacroPDPPage = Nothing
SetAcroApp = Nothing : SetAcroAVDoc = Nothing

```



AcroExch.PDPPage.GetAnnotIndex() Method

Description

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

Syntax

```
returnValue = Object.GetAnnotIndex( pdAnnot )
```

Arguments

Parameter	Description
<i>pdAnnot</i>	The AcroExch.PDAnnot whose index is obtained.

Return Value

The annotation's index.

Related Methods

- **PDPage.GetAnnot()**
- **PDPage.GetNumAnnots()**

AcroExch.PDPage.GetDoc() Method

Description

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

Syntax

```
Set pdDoc = object.GetDoc()
```

Return Value

The page's **AcroExch.PDDoc**.

Related Methods

- **AVPageView.GetPage**
- **AVPageView.GetPageNum**
- **PDDoc.AcquirePage**
- **PDDoc.GetNumPages**
- **PDPage.GetNumber**
- **PDPage.GetRotate()**
- **PDPage.GetSize()**
- **PDTextSelect.GetPage()**

AcroExch.PDPage.GetNumAnnots() Method

Description

Gets the number of annotations on the page.

Syntax

```
returnValue = object.GetNumAnnots()
```

Return Value

The number of annotations on the page.

Related Methods

- **PDPage.GetAnnot()**

- `PDPage.GetAnnotIndex()`

AcroExch.PDPage.GetNumber() Method

Description

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

Syntax

```
returnValue = object.GetNumber()
```

Return Value

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

Related Methods

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

AcroExch.PDPage.GetRotate() Method

Description

Gets the rotation value for the current page.

Syntax

```
Set point = object.GetSize()
```

Return Value

a number that represents one of the values from **Page Rotation Enumeration** at Appendix 16

Related Methods

- `AVPageView.GetPage`
- `AVPageView.GetPageNum`
- `PDDoc.AcquirePage`
- `PDPage.GetNumber`
- `PDPage.GetSize`
- `PDPage.SetRotate`
- `PDTextSelect.GetPage`

AcroExch.PDPage.GetSize() Method

Description

Gets a page's width and height.

Syntax

```
Set point = object.GetSize()
```

Return Value

An **AcroExch.Point** containing the width and height, measured in points.

Related Methods

- AVPageView.**GetPage()**
- AVPageView.**GetPageNum()**
- PDDoc.**AcquirePage()**
- PDPage.**GetNumber()**
- PDPage.**GetRotate()**
- PDTextSelect.**GetPage()**

AcroExch.PDPage.RemoveAnnot() Method

Description

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

Syntax

```
returnValue = object.RemoveAnnot( nIndex )
```

Return Value

false if the Acrobat viewer does not support editing, **true** otherwise.

Related Methods

- PDPage.**AddAnnot**
- PDPage.**AddNewAnnot**
- PDPage.**GetAnnotIndex**

AcroExch.PDPage.SetRotate () Method

Description

Sets the rotation for the current page.

Syntax

```
returnValue = Object.SetRotate( nRotate )
```

Arguments


Parameter	Description
<i>acroHiliteList</i>	Rotation value. See 0Page Rotation Enumeration at Appendix 16

Return Value

false if the Acrobat viewer does not support editing, **true** otherwise.

Related Methods

- AVDoc.**GetRotate()**

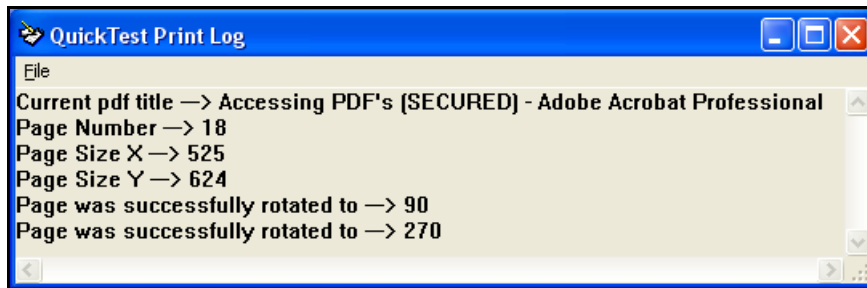
 **Example**

```

Option Explicit
Private Const pdRotate0 = 0
Private Const pdRotate90 = 90
Private Const pdRotate180 = 180
Private Const pdRotate270 = 270
DimacroApp, avDoc, pdDoc, pdPage
DimacroPoint
DimgPDFPath

gPDFPath = "C:\QTTutorial.pdf"
' ** Initialize Acrobat by creating App object
SetacroApp = CreateObject( "AcroExch.App" )
' ** show acrobat
acroApp.Show()
' ** Set AVDoc object
SetavDoc = CreateObject( "AcroExch.AVDoc" )
' ** open the PDF
If avDoc.Open( gPDFPath, "Accessing PDF's" ) Then
    If avDoc.IsValid = False Then ExitTest()
    avDoc.BringToFront()
    Call avDoc.Maximize( True )
    Print "Current pdf title ---> " & avDoc.GetTitle()
    Call avDoc.GetAVPageView.Goto( 18 )
    Set pdDoc = avDoc.GetPDDoc()
    Set pdPage = avDoc.GetPDDoc.AcquirePage( 18 )
    Print "Page Number ---> " & pdPage.GetNumber()
    SetacroPoint = pdPage.GetSize()
    Print "Page Size X ---> " &acroPoint.x
    Print "Page Size Y ---> " &acroPoint.y
    SetacroPoint = Nothing
    ' ** Rotating 90 deg ...
    If pdPage.SetRotate( pdRotate90 ) Then
        Print "Page was successfully rotated to ---> " &pdPage.GetRotate()
        Wait 1
    End If
    ' ** Rotating 270 deg ...
    If pdPage.SetRotate( pdRotate270 ) Then
        Print "Page was successfully rotated to ---> " &pdPage.GetRotate()
        Wait 1
    End If
    ' ** Restoring ...
    Call pdPage.SetRotate( pdRotate0 )
End If
AcroApp.CloseAllDocs()
AcroApp.Exit()
SetacroApp = Nothing : SetavDoc = Nothing : SetpdDoc = Nothing
SetpdPage = Nothing

```

AcroExch.PDTextSelect Object

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.

AcroExch.PDTextSelect.Destroy () Method

Description

Destroys a text selection.

Syntax

```
Object.Destroy()
```

Return Value

Always returns **true**.

Related Methods

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

AcroExch.PDTextSelect.GetBoundingRect () Method

Description

Gets a text selection's bounding rectangle.

Syntax

```
Set rect = Object.GetBoundingRect()
```

Return Value

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**

AcroExch.PDTextSelect.GetNumText () Method

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.

Syntax

```
returnValue = Object.GetNumText()
```

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()**

Example

```
Option Explicit

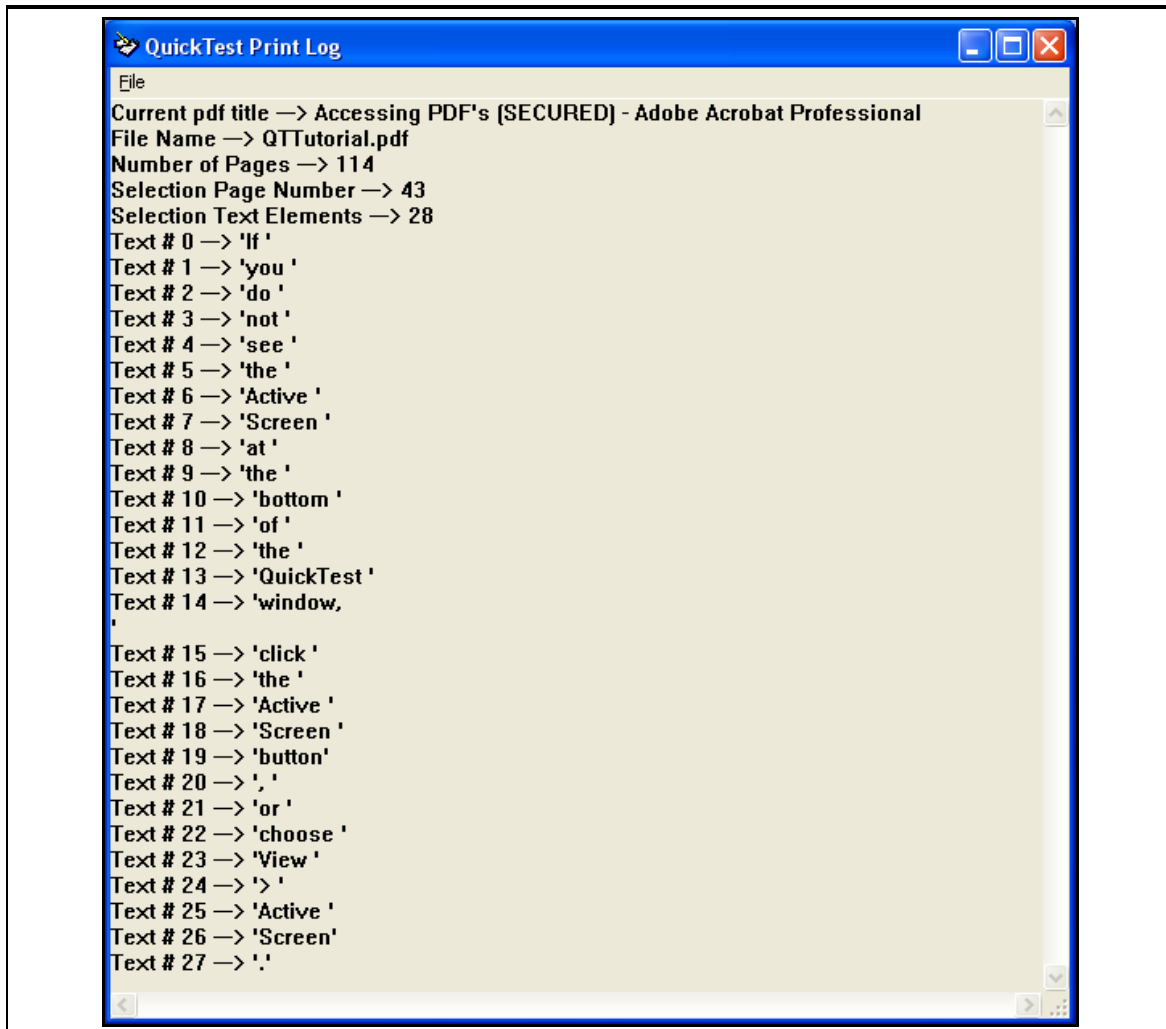
Dim acroApp, acroAVDoc, acroPDDoc, acroRect, PDTextSelect
Dim gPDFPath, nElem

gPDFPath = "C:\QTTutorial.pdf"
' ** Initialize Acrobat by creating App object
```

```

SetacroApp = CreateObject( "AcroExch.App" )
' ** show acrobat
acroApp.Show()
' ** Set AVDoc object
SetacroAVDoc = CreateObject( "AcroExch.AVDoc" )
' ** open the PDF
IfacroAVDoc.Open( gPDFPath, "Accessing PDF's" ) Then
    IfacroAVDoc.IsValid = False Then ExitTest()
    acroAVDoc.BringToFront()
    CallacroAVDoc.Maximize( True )
    Print "Current pdf title ----> " & acroAVDoc.GetTitle()
    SetacroPDDoc = acroAVDoc.GetPDDoc()
    Print "File Name ----> " & acroPDDoc.GetFileName()
    Print "Number of Pages ----> " & acroPDDoc.GetNumPages()
    SetacroRect = CreateObject( "AcroExch.Rect" )
    acroRect.Bottom = 380 : acroRect.Top = 400
    acroRect.Left = 100 : acroRect.Right = 500
    ' ** Selecting page 42 ( index is 43 )
    SetPDTextSelect = acroPDDoc.CreateTextSelect( 43, acroRect )
    IfPDTextSelect Is Nothing Then
        Print "Unable to Create TextSelect object."
        ExitTest()
    End If
    CallacroAVDoc.SetTextSelection( PDTextSelect )
    CallacroAVDoc.ShowTextSelect()
    Print "Selection Page Number ----> " & PDTextSelect.GetPage()
    Print "Selection Text Elements ----> " & PDTextSelect.GetNumText()
    ' ** Looping through text elements
    For nElem = 0 To PDTextSelect.GetNumText() - 1
        Print "Text # " & nElem & " ----> '" & PDTextSelect.GetText( nElem ) & "'"
    Next
    ' ** Destroying Text Selection
    CallPDTextSelect.Destroy()
End If
AcroApp.CloseAllDocs()
AcroApp.Exit()
SetPDTextSelect = Nothing : SetacroRect = Nothing
SetAcroApp = Nothing : SetAcroAVDoc = Nothing

```



AcroExch.PDTextSelect.GetPage () Method

Description

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

Syntax

```
returnValue = Object.GetPage()
```

Return Value

The text selection's page number. The first page in a **PDDoc** 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**()

AcroExch.PDTextSelect.GetText () Method

Description

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

Syntax

```
returnValue = Object.GetText( textIndex )
```

Arguments

Parameter	Description
<i>nTextIndex</i>	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**()
- PDPage.**CreatePageHilite**()
- PDDoc.**CreateTextSelect**()
- PDPage.**CreateWordHilite**()
- PDTextSelect.**Destroy**()
- PDTextSelect.**GetBoundingRect**()
- PDTextSelect.**GetNumText**()
- PDTextSelect.**GetPage**()
- AVDoc.**SetTextSelection**()
- AVDoc.**ShowTextSelect**()

AcroForm OLE Automation

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 have to 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 from within the Visual Basic environment. This is also necessary in order to allow early binding. Dim the program variables as 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).

AFormAut.App Object

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 document.

AFormAut.Field Object

A field in the document that is currently active in Acrobat.

AFormAut.Fields Object

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.

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.

Developer FAQ

User and Developer Resources

What Do I Need to Download from the Web to Get the Acrobat SDK?

The Acrobat **SDK** is made up of many different technical documents and samples for different operating systems. You can download the installer from the Web site to obtain the entire Acrobat **SDK**, or you can download each piece separately using the "exploded" version. You will need to determine what you would like to do with the SDK, and then check

if there is a sample that gets you started in the right direction.

See the Acrobat **SDK Main Page** at <http://partners.adobe.com/asn/acrobat/index.jsp> to access the Acrobat **SDK**. The Web site uses lock icons next to individual portions of the SDK to indicate that ASN membership or a subscription is required for access.

Where Can I Get Help With Acrobat Product Issues?

- For information and support for the Acrobat and LiveCycle products, you can visit the following sites:
- Acrobat Product Information: <http://www.adobe.com/products/acrobat/>.
- Acrobat Product Support: <http://www.adobe.com/support/products/acrobat.html>.
- Adobe LiveCycle Product Information: <http://www.adobe.com/products/server/>.
- For information on using or installing Adobe Acrobat products, start at **the Acrobat Product Support Page** at <http://www.adobe.com/support/products/acrobat.html>, which contains free downloads of the latest Acrobat product updates, answers to top product support issues, a searchable support database, user forums, and product support telephone numbers.
- For International customers, see the Adobe International Technical Support and Service Page.

What Are the API Differences Between Acrobat and Adobe Reader?

Acrobat provides a “full-featured” development environment that includes the entire Acrobat core API. There are some small differences between the public APIs available in Acrobat Professional and Acrobat Standard. These are documented in the Acrobat SDK Plug-in Guide and the Acrobat and **PDF Library Reference**.

The APIs that may be used for Adobe Reader are limited technically and legally. Technical limitations are documented in the Acrobat and PDF Library API Overview and the Acrobat Interapplication Communication Overview.

Both Acrobat and Adobe Reader accept plug-ins. The primary difference between the two is that Adobe Reader can neither make changes to a file nor save a file. API methods that change a file in such a way that a save would be required are not available in Adobe Reader.

What API Methods Are Available to Modify PDF Documents?

Modifying the page contents of a **PDF** file is primarily accomplished using the Acrobat core **API**.

Using the Acrobat core **API** greatly simplifies modifying and creating **PDF** page contents. To demonstrate this, there are several “snippets” available from the SnippetRunner sample plug-in that show you how to add data to the contents of a page, while ensuring that the **PDF** file is still readable after modification. To attempt to do this without using the core **API** would be significantly more difficult and could result in an unreadable **PDF** file.

Acrobat’s automation interfaces are limited mainly to what a user can do through the user interface and cannot modify the contents of a page.

For more information on the Acrobat core **API**, see the Acrobat and **PDF Library API Overview** and Acrobat and **PDF Library API Reference**.

How Can I Extract Text From PDF Documents Using the Acrobat SDK?

You can extract text with the Acrobat SDK in two ways:

- Use the Acrobat core API
- Use Acrobat's automation API

How Do I Use Command Lines with Acrobat and Adobe Reader on Windows?

These are unsupported command lines, but have worked for some developers. There is no documentation for these commands other than what is listed below. You can display and print a PDF file with Acrobat and Adobe Reader from the command line.

AcroRd32.exe pathname — Executes Adobe Reader and displays the file, whose full path must be provided. Other options for the command line are:

Option	Description
/n	Launches a separate instance of Acrobat or Adobe Reader, even if one is currently open.
/s	Opens Acrobat or Adobe Reader, suppressing the splash screen.
/o	Opens Acrobat or Adobe Reader, suppressing the open file dialog.
/h	Opens Acrobat or Adobe Reader in a minimized window.
/p pathname	Executes Adobe Reader and displays the Print dialog box
/t path "printername" "drivename" "portname"	Initiates Adobe Reader and prints a file, whose path must be fully specified, while suppressing the Print dialog box. Where <ul style="list-style-type: none"> ■ printername — The name of your printer. ■ drivename — Your printer driver's name, as it appears in your printer's properties. ■ portname — The printer's port. portname cannot contain any "/" characters; if it does, output is routed to the default port for that printer.

PDF Tasks

Find Text in all PDF's File Open

Description

This code is assuming you only have at least one AVDoc open. Also, if no occurrences of the queried text are found, a dialog box kicks up in the browser window.

Example

```
Option Explicit

Dim bFound, i
Dim AcroApp, AcroAVDoc
Set AcroApp = CreateObject( "AcroExch.App" )

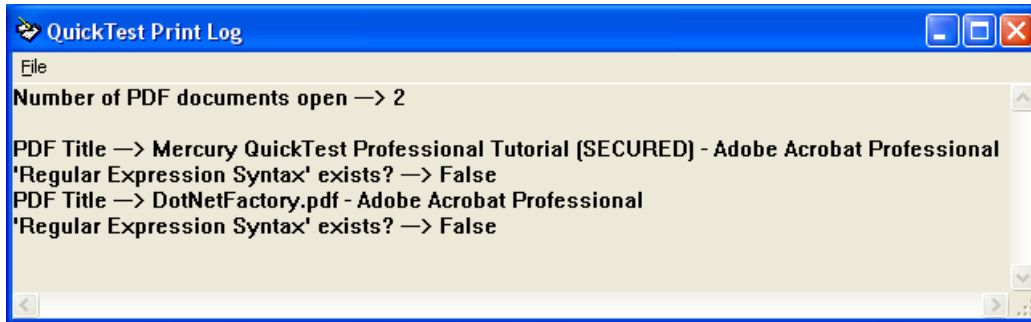
Dim bCaseSensitive, bWholeWordsOnly, bReset
```



```

bCaseSensitive = True
bWholeWordsOnly = False
bReset = False
For i = 0 To AcroApp.GetNumAVDocs - 1
    Set AcroAVDoc = AcroApp.GetAVDoc( i )
    If AcroAVDoc.IsValid Then
        bFound = AcroAVDoc.FindText( "Regular Expression Syntax", _
                                     bCaseSensitive, bWholeWordsOnly, bReset )
        Print "PDF Title ----> " & AcroAVDoc.GetTitle()
        Print "'Regular Expression Syntax' exists? ----> " & bFound
    End If
Next
Set AcroAVDoc = Nothing : Set AcroApp = Nothing

```



Invoking AVCommands Programmatically

Description

On Windows, you can get the width of a text string with `GetTextExtentPoint32()`. There are no Acrobat API functions to get the width of a text string; you have to go to platform-dependent code.

Example

```

Option Explicit

Dim bFound, i
Dim AcroApp, AcroAVDoc
Set AcroApp = CreateObject( "AcroExch.App" )

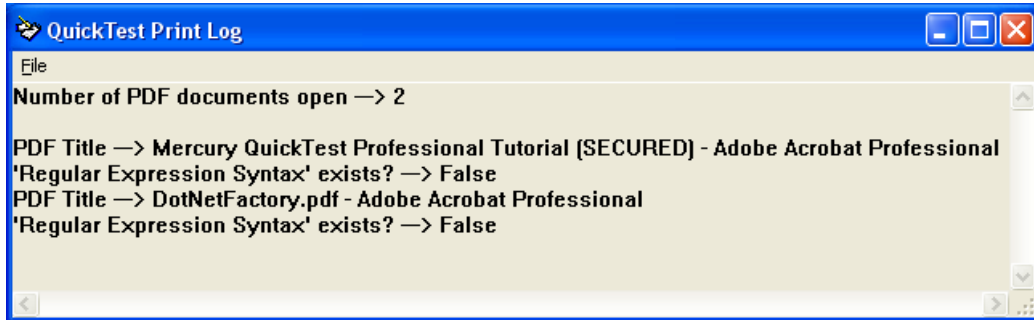
Dim bCaseSensitive, bWholeWordsOnly, bReset

bCaseSensitive = True
bWholeWordsOnly = False
bReset = False
For i = 0 To AcroApp.GetNumAVDocs - 1
    Set AcroAVDoc = AcroApp.GetAVDoc( i )
    If AcroAVDoc.IsValid Then
        bFound = AcroAVDoc.FindText( "Regular Expression Syntax", _
                                     bCaseSensitive, bWholeWordsOnly, bReset )
        Print "PDF Title ----> " & AcroAVDoc.GetTitle()
        Print "'Regular Expression Syntax' exists? ----> " & bFound
    End If

```

Next

```
Set AcroAVDoc = Nothing : Set AcroApp = Nothing
```



Searching words in PDF

Description

The following code search a specific word on a specific pdf file. If a pdf/pdf's files already open, it will search on the first pdf.

Example

Option Explicit

```
Dim app, pdDoc, avDoc, oDialog
Dim fname, gPDFPath, numOpenPDFs, bExisting
Dim InputText, dateTimeStart, dateTimeEnd, timeSpan

Private Function FindWordJSO( ByVal InputText )
    Dim bStop
    Dim jso, nCount, i, j
    Dim word, result, foundErr, nPages, nWords
    Dim rc, str_Renamed

    ' ** get JavaScript Object
    ' ** note jso is related to PDDoc of a PDF,
    Set jso = pdDoc.GetJSObject
    nCount = 0
    bStop = False
    ' ** search for the text
    If Not jso Is Nothing Then
        Print "Searching ... "
        ' ** total number of pages
        nPages = jso.numPages
        ' ** Go through pages
        For i = 0 To nPages - 1
            ' ** check each word in a page
            nWords = jso.getPageNumWords( i )
            For j = 0 To nWords - 1
                ' ** get a word
                word = jso.getPageNthWord( i, j )
                If VarType( word ) = vbString Then
                    ' ** compare the word with what the user wants
                    result = StrComp( word, InputText, vbTextCompare )
                End If
            Next j
        Next i
    End If
End Function
```

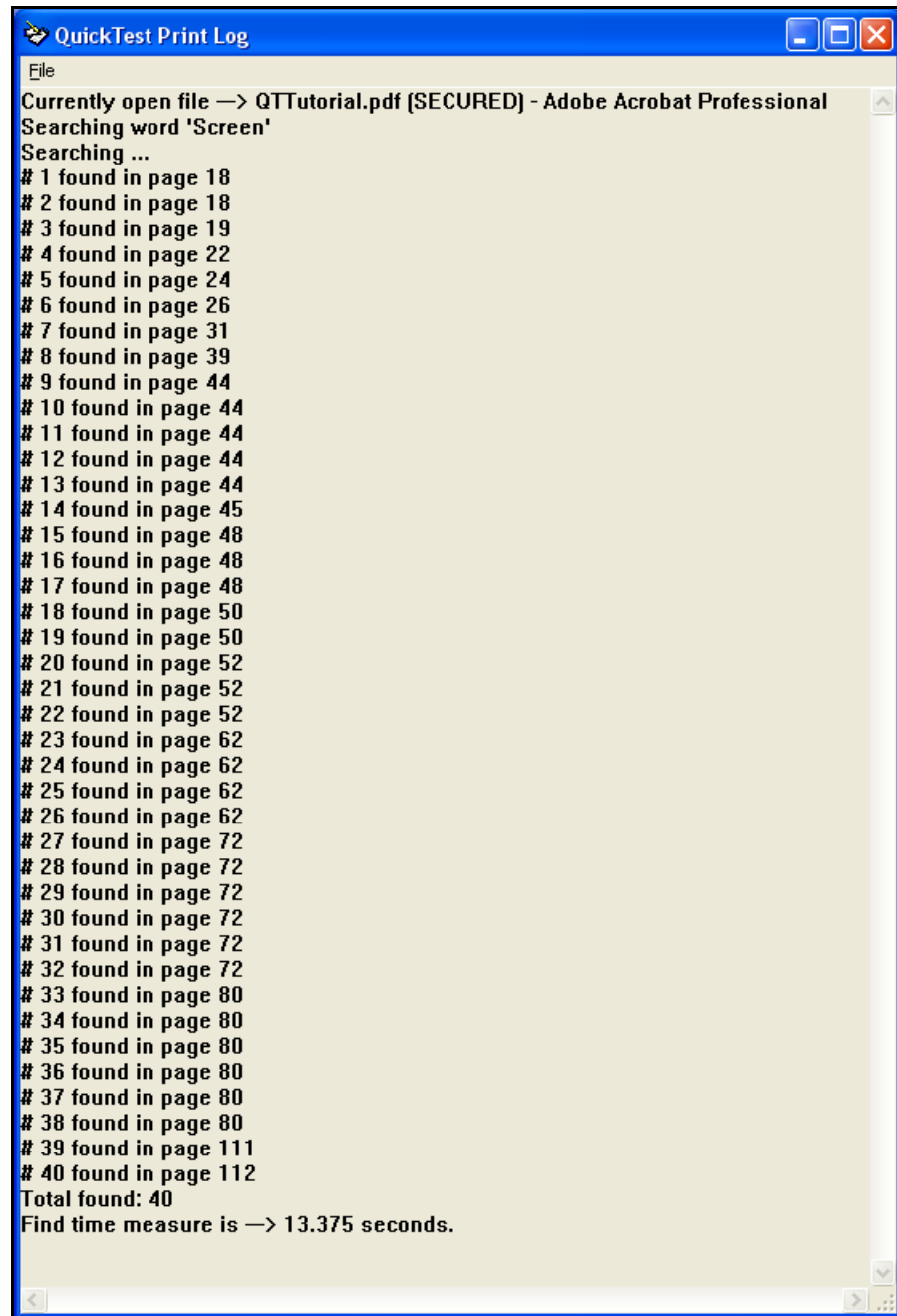
```

        ' ** if same
        If result = 0 Then
            nCount = nCount + 1
            rc = jso.selectPageNthWord( i, j )
            Print "# " & nCount & " found in page " & ( i + 1 )
            Print "Searching ... "
        End If
    End If
End If
Next
Next
End If
FindWordJSO = nCount
Set jso = Nothing
End Function

' ** Initialize Acrobat by creating App object
' ** If Acrobat is running, it will get the existing acrobat object.
Set app = CreateObject( "AcroExch.App" )
' ** show Acrobat
app.Show()
' ** is a PDF already open?
numOpenPDFs = app.GetNumAVDocs()
' ** if there is a PDF opened, get doc object
If numOpenPDFs > 0 Then
    bExisting = True
    Set avDoc = app.GetActiveDoc()
    Set pdDoc = avDoc.GetPDDoc()
    Print "Currently open file ----> " & avDoc.GetTitle()
Else
    bExisting = False
End If
' ** if no pdf open, then open a new one...
If Not bExisting Then
    Set oDialog = CreateObject( "UserAccounts.CommonDialog" )
    oDialog.Filter = "Pdf Files|.pdf"
    oDialog.FilterIndex = 1
    oDialog.InitialDir = Environment( "ProductDir" )
    If oDialog.ShowOpen = 0 Then
        ExitTest( "Error" )
    Else
        gPDFPath = oDialog.FileName
    End If
    Set pdDoc = CreateObject( "AcroExch.PDDoc" )
    ' ** open the PDF in acrobat
    If pdDoc.Open( gPDFPath ) Then
        fname = pdDoc.GetFileName
        pdDoc.OpenAVDoc( fname )
    Else
        MsgBox( "Failed to open " & gPDFPath )
        ExitTest( "Error" )
    End If
End If
InputText = InputBox( "Please input a word", "Accessing PDF", "" )

```

```
If Len( InputText ) = 0 Then
    MsgBox( "Please input a word" )
    ExitTest( 0 )
Else
    Print "Searching word '" & InputText & "'"
    Set dateTimeStart = DotNetFactory.CreateInstance( "System.DateTime" ).Now
    Call FindWordJSO( InputText )
    Set dateTimeEnd = DotNetFactory.CreateInstance( "System.DateTime" ).Now
    Set timeSpan = dateTimeEnd.Subtract( dateTimeStart )
    If nCount > 0 Then
        Print "Total found: " & nCount
        Print "Find time measure is ---> " & timeSpan.TotalSeconds & " seconds."
    Else
        Print "Not found in the document"
    End If
End If
' ** End Acrobat only if it was not existing before
' ** or it is existing but no PDF file opened.
If Not app Is Nothing Then
    If bExisting = False Then
        app.CloseAllDocs()
        app.Exit()
    End If
End If
' ** Cleaning
Set app = Nothing : Set pdDoc = Nothing : Set avDoc = Nothing
```



Appendix 15

View Mode Enumeration

Constant	Value	Description
PDDontCare	0	Leave the view mode as it is.
PDUseNone	1	Display the document, but neither bookmarks nor thumbnail images.
PDUseThumbs	2	Display the document and thumbnail images.

PDUseBookmarks	3	Display the document and bookmarks.
PDFFullScreen	4	Display the document in full screen mode.

Table 1 – View Mode Enumeration**Toolbar Items Names**

Constant	Description
UseNone	Displays only the document, but neither bookmarks nor thumbnail images.
UseBookmarks	Displays the document and bookmarks. UseThumbs Displays the document and thumbnail images.
endPageModeGroup	Separator (not visible in the toolbar).
Hand	Allows the user to scroll within the current page.
ZoomIn	Increases the zoom factor.
ZoomOut	Decreases the zoom factor. (not available after Acrobat 2.1)
Select	Allows the user to select text.
Note	Allows the user to create, select, or edit notes
Link	Allows the user to create a link or manipulate an existing link.
endToolsGroup	Separator (not visible in the toolbar).
FirstPage	Goes to the document's first page.
PreviousPage	Goes to the previous page in the document.
NextPage	Goes to the next page in the document.
LastPage	Goes to the document's last page.
endPageNavGroup	Separator (not visible in the toolbar).
GoBack	Goes to the previous view in the view history
GoForward	Goes to the next view in the view history.
endPageStackGroup	Separator (not visible in the toolbar).
Zoom100	Sets the zoom factor to 100%
FitPage	Sets the zoom factor to fit the entire page into the window.
FitVisible	Sets the zoom factor to fit the portion of the page on which drawing appears into the window.

Table 2 – Toolbar item names**Zoom Strategy Enumeration**

Zoom Type	Value	Description
AVZoomNoVary	0	No variable zoom (i.e., zoom is a fixed value such as 100%). Use this for XYZ zoom.
AVZoomFitPage	1	Fit page to window.
AVZoomFitWidth	2	Fit page width to window.
AVZoomFitHeight	3	Fit page height to window.
AVZoomFitVisibleWidth	4	Fit visible width to window.
AVZoomPreferred	5	

Table 3 – Zoom Strategy Enumeration**Page Rotation Enumeration** **Description**

Constant values that specify page rotation, in degrees. Used for routines that set and get the value of a page's Rotate key.

Zoom Type	Value	Description
pdRotate0	0	No variable zoom (i.e., zoom is a fixed value such as 100%). Use this for XYZ zoom.
pdRotate90	90	Fit page to window.
pdRotate180	180	Fit page width to window.
pdRotate270	270	

Table 4 – Page Rotation Enumeration**Document Flags Enumeration**

Zoom Type	Value	Description
PDDocNeedsSave	1	Document has been modified and needs to be saved.
PDDocRequiresFullSave	2	Document cannot be saved incrementally; it must be written using PDSaveFull .
PDDocIsModified	4	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	8	Document is based on a temporary file that must be deleted when the document is closed or saved.
PDDocWasRepaired	16	Document was repaired when it was opened.
PDDocNewMajorVersion	32	Document's major version is newer than current.
PDDocNewMinorVersion	64	Document's minor version is newer than current.
PDDocOldVersion	128	Document's version is older than current.
PDDocSuppressErrors	256	Don't display errors.

Table 5 – Document Flags Enumeration