

Automating Acrobat

P A R T 1

Using pdfmark

This is a practical guide to using pdfmark for creation of enhanced PDF files. It is especially suitable for VBA programmers, as most examples use Microsoft Word, but has much wider applicability. The methods shown, with a little adaptation, can be used in a range of application software and programming/macro languages.

Roy Walter

Automating Acrobat Part 1

© 2000 Brook House Ltd, Scotland. Text © 2000 Roy Walter.

All rights reserved. No part of this book, including design, may be reproduced or transmitted in any form, electronic, photocopying, recording or otherwise, without the written consent of the publisher. You may make a single printed copy of the electronic version of this book **for personal use only**.

The author and publisher have made every effort to ensure the accuracy of the information herein. The information contained in this book, however, is sold without warranty, either express or implied. Neither the author, Brook House Ltd, nor its resellers and distributors will be held liable for any damages caused or alleged to be caused either directly or indirectly by this book.

ISBN 1 873547 31 5

Trademark Acknowledgement

The publisher has endeavoured to provide trademark information about all companies and products mentioned in this book through the appropriate use of capitals. The publisher cannot, however, guarantee the accuracy of this information.

Published by Brook House Publishing Ltd
Garlieston House, Garlieston, Wigtownshire DG8 8HF Scotland
<http://www.brookhouse.co.uk>

Contents

Contents.....	3
Chapter 1 Introduction	5
<i>Who should read this book.....</i>	<i>5</i>
<i>The enhanced PDF file.....</i>	<i>6</i>
<i>pdfmark syntax.....</i>	<i>6</i>
<i>About the examples in this book.....</i>	<i>6</i>
Conventions used in this book	6
Chapter 2 PostScript and PDF files	8
<i>Selecting printers for PDF creation.....</i>	<i>9</i>
<i>Distiller automation.....</i>	<i>9</i>
Chapter 3 pdfmark basics.....	10
<i>Introduction to pdfmark.....</i>	<i>10</i>
<i>Getting your co-ordinates.....</i>	<i>11</i>
Explanatory notes	13
A cautionary note	14
<i>About creating pdfmark</i>	<i>14</i>
<i>Summary.....</i>	<i>16</i>
Chapter 4 Destinations and Actions.....	17
<i>View Destinations</i>	<i>17</i>
Destination fit types.....	18
<i>Actions.....</i>	<i>19</i>
GoTo action	19
GoToR action.....	20
Launch action	20
Article action.....	21
<i>Named destinations.....</i>	<i>21</i>
Destination creation tactics.....	25

Chapter 5 Acrobat Bookmarks	28
<i>Sources of bookmarks</i>	28
<i>Bookmark basics</i>	28
<i>Advanced bookmarks</i>	32
Multilevel bookmarks.....	32
<i>Bookmark ‘gotchas’</i>	37
Chapter 6 Link Annotations	39
<i>Controlling link appearance in pdfmark</i>	41
<i>Link creation tactics</i>	41
Links from hyperlink and other Word ‘goto’ fields.....	41
Chapter 7 Note Annotations	44
<i>Changing note icons in pdfmark</i>	45
<i>Note creation tactics</i>	47
Creating Notes from comments	47
Creating notes from text boxes	49
Chapter 8 Articles	51
<i>Article creation tactics</i>	52
Chapter 9 Info and Catalog Dictionaries	54
<i>Info Dictionary</i>	54
Info dictionary creation tactics	55
<i>The Catalog Dictionary</i>	55
Appendix 1 User-defined Functions Tutorial	58
<i>What is a function</i>	58
Appendix 2 Distiller Automation	62
<i>Distiller automation properties</i>	63
<i>Distiller automation events</i>	63
Appendix 3 Dealing with Rotated Pages	65

Appendix 2

Distiller Automation

The provision of an OLE interface to Distiller in Acrobat 4.0 is a great benefit, making it much easier to harness the power of Distiller for your own applications and workflows. The interface exposes one OLE object — PDFDistiller. As well as ease of use, the provision of this interface means that you can get status information from Distiller, and relay it to your users.

In VB/VBA you can use a class module to access the methods, properties, and events associated with the PDFDistiller object. So add a class module to your project, and add the following to the Declarations section of the module. (Note: make sure that your project includes a reference to the Acrobat Distiller library.)

```
Public WithEvents oDist As PDFDistiller
```

By using the WithEvents qualifier, you will find that the PDFDistiller events are added to the class module. A class module itself has two events, Initialize and Terminate, so add the following code to the Initialize event, to create the PDFDistiller object when the class is instantiated elsewhere in your code.

```
Private Sub Class_Initialize  
  
    Set oDist = New PDFDistiller  
  
End Sub
```

To use the class in your code you need to create a new instance using the New keyword. So in a standard module you would create a variable for the class instance, and then instantiate it as below.

```
Private Sub DistillFile(svl nputPS, svOutputPDF, svJobOptions)  
  
    ' Assuming you have called your class module cAcroDist  
    Dim myPDFDist As cAcroDist  
  
    Set myPDFDist = New cAcroDist  
    Call myPDFDist.oDist.FileToPDF(svl nputPS, _  
        svOutputPDF, svJobOptions)  
  
End Sub
```

The above example introduces the `FileToPDF` method for distilling a PostScript file. Obviously, you need to create a PostScript file somewhere else in your program for Distiller to work on. The arguments for the method should be self-explanatory, but the following points should be noted:

- If you omit the PDF filename (`svOutputPDF`), Distiller generates a PDF filename from `svInputPS`, replacing the extension with “.pdf”.
- If you don’t provide a Job Options filename, Distiller uses its current Job Options file.
- If you do provide a Job Options argument as a filename only, Distiller assumes the file to be in its Settings Directory.

Distiller automation properties

The following properties are available.

```
bShowWindow = [True/False]
```

- show/hide Distiller status window at startup

```
bSpoolJobs = [True/False]
```

- to determine whether or not jobs are spooled by Distiller

For example.

```
myPDFDist.oDist.bShowWindow = True
Call myPDFDist.oDist.FileToPDF _
    (svInputPS, svOutputPDF, svJobOptions)
```

Distiller automation events

There are six Distiller events that you can use to chart the progress and/or status of a job. Again, they are more or less self documenting, but for the sake of completeness they are listed below.

```
OnJobStart(ByVal strInputPostScript As String, _
ByVal strOutputPDF As String)
```

- fired when a job begins processing

```
OnJobDone(ByVal strInputPostScript As String, _
ByVal strOutputPDF As String)
```

- fired when a job completes successfully

```
OnJobFail (ByVal strInputPostScript As String, _  
ByVal strOutputPDF As String)
```

- fired when a job ends unsuccessfully

```
OnLogMessage(ByVal strMessage As String)
```

- fired at various times with the text messages that normally appear in Distiller's message log window. The text contains the usual carriage return-line feed pair at the end of each line. A single call to OnLogMessage may contain multiple lines or partial lines of text. In the current version of Distiller, the text that's passed in this message may contain line feed characters without carriage return characters. The application should not make any assumptions about how this text is formatted and should be prepared to receive either line feed characters (LF) alone or carriage return-line feed (CR-LF) pairs.

```
OnPercentDone(ByVal nPercentDone As Long)
```

- fired periodically during a job to indicate overall progress

```
OnPageNumber (ByVal nPageNumber As Long)
```

- fired periodically during a job to indicate the current page number

For example, you might want to display a progress dialog (from the class module) as follows.

```
Private Sub oDist_OnPercentDone(ByVal nPercentDone As Long)  
  
    With frmProgress  
  
        If nPercentDone = 100 Then  
            Beep  
        Else  
            .Label1.Visible = True  
            .Label1.Caption = "Distilling ... " & nPercentDone & "%"  
            .Repaint  
        End If  
  
    End With  
  
End Sub
```