Documaker™

Using the PDF Print Driver

version 11.2

One Lincoln Centre
5400 LBJ Freeway, Suite 300
Dallas, Texas 75240
Internet: www.docucorp.com

Phone: (U. S.) 214 891 6500
(EMEA) +44 (0) 1372 366 200
FAX: (U. S.) 214 987 8187
(EMEA) +44 (0) 1372 366 201

Client Support: (U. S.) 800 818 7778
(EMEA) +44 (0) 1372 366 222
Support@docucorp.com

PUBLICATION COPYRIGHT NOTICE

TRADEMARKS

SOFTWARE COPYRIGHT NOTICE AND COPY LIMITATIONS

PUBLICATION COPY LIMITATIONS

DISCLAIMER

The contents of this publication and the computer software it represents are subject to change without notice. Publication of this manual is not a commitment by Docucorp International to provide the features described. Docucorp International does not assume responsibility or liability for errors that may appear herein. Docucorp International reserves the right to revise this publication and to make changes in it from time to time without obligation of Docucorp International, to notify any person or organization of such revision or changes.

The screens and other illustrations in this publication are meant to be representative, not exact duplicates, of those that appear on your monitor or printer.

CONTENTS

Setting Up the PDF Print Driver

Customizing PDF Output

Working With Fonts

Adding Security to PDF Files

CHAPTER 1

Setting Up the PDF Print Driver

The PDF Print Driver creates Adobe Portable Document Format (PDF) files from output from Docucorp software such as the Policy Production System (PPS), Documaker Server, and Documaker Workstation.

PDF is a document language developed by Adobe Systems, Inc., that allows formatting and graphics information to be stored along with text. Using PDF files, you can make sure form sets viewed and printed match the originals.

A document stored in PDF format can be transmitted to and viewed on many types of systems. There are PDF viewer applications available for many platforms, both as stand-alone programs and as add-ons for existing applications (such as word processors and Internet web browsers). You can download Acrobat Reader from Adobe Systems' web site (www.adobe.com).

Print output directed to the PDF Print Driver is written to disk and stored in one or more files. You can then view these files using Acrobat Reader. This document discusses...

Installing the PDF Print Driver

First make sure you have the correct system requirements to run your Docucorp software. For instance, if you are using Policy Production System (PPS), see the Documaker Workstation Supervisor Guide for information on what you need to run the system. The PDF Print Driver runs on a variety of Windows 32-bit operating systems, such as Windows 2000 and Windows XP.

NOTE: The PDF Driver is also available on OS/390 platforms. Contact your sales representative for licensing and installation information.

Once you make sure your computer has the correct hardware and software, adding the ability to output PDF files from PPS requires these steps:

1. Insert the PDF Print Driver CD into your CD-ROM drive and use a text editor to view the readme file on the CD for detailed installation instructions.

   The installation routine installs the PDF DLL file into your Docucorp product's executable directory, such as \ppswin\dll.

2. Make sure you have the following options in your FSISYS.INI file:

NOTE: *Before* making any changes to these files, back up your INI files.

```
< Control >
    LoadPrintOnly  = Yes
< PrtType:PDF >
    Device         = E:\TEST.PDF
    Bookmark       = Yes,Page
    DownloadFonts  = No,Enabled
    Module         = PDFW32
    PageNumbers    = Yes
    PrintFunc      = PDFPrint
    SendOverlays   = No,Enabled
    SendColor      = Yes,Enabled
    PrintViewOnly  = No
    SplitText      = No
    SplitPercent   = 50
    Class          = PDF
    DisplayMode    = UseOutlines
    PaperSize      = 0
    Linearize      = Yes
    FontCompression= 2
```

Option	Description
Control	
LoadPrintOnly	Enter Yes to tell the system to load print only forms.
PrtType:PDF	

Option	Description

Device	Enter the path and file name for the PDF output. Here is an example:
	`Device = F:\PDF\~KEYID ~DATE ; [%I-%M %p];.PDF`
	In this example, the system will write a file to the \PDF directory on the F: drive. The file name would include the policy number (KeyID), the time the file was created, and it will have an extension of *PDF*. For example, the file written to disk would have a name similar to:
	`F:\PDF\A100[3-47PM].PDF`

Bookmark	This option contains three values. Separate the values with commas (,). The first value enables bookmarks. Enter Yes to tell the system to create bookmarks. The default is *No*. The second value indicates the lowest level for which the system will create bookmarks. You can choose from *Formset*, *Group*, *Form*, or *Page*. For example, if you enter *Form*, the system creates bookmarks for each form set, for each group in all form sets, and for each form in all of the groups. The default is *Page*. Use the third value to turn off generic bookmarks for all pages. For example, enter No if you do not want every page to have a bookmark like Page 01, Page 02, Page 03, and so on. If you only want TLE bookmarks to appear, enter No for the third value.
DownloadFonts	Set to *No, Enabled*. Set to *Yes* if you want to embed fonts. See Embedding Fonts on page 33 for more information.
Module	The name of the program module which contains the system's PDF print driver. See also the Class option.
PageNumbers	Set to *No* if you do not want page numbers. Defaults to *Yes*.
PrintFunc	The name of the program function that is the main entry point into the system's PDF print driver.
SendOverlays	Set to *No, Enabled*
SendColor	Set to *Yes, Enabled*
PrintViewOnly	Use this option to output view only forms when you create PDF files. Set to *Yes* to print these images. Defaults to *No*. Images marked as Entry only will not be printed, as these usually are the worksheet type images used for data collection.
SplitText	Use the SplitText and SplitPercent options if you see text that is not positioned accurately when the PDF file is viewed. If the SplitText option is set to *Yes*, every text string will be split and adjusted position according to the value of the SplitPercent option. See Handling Fonts with Multiple Width Tables on page 36 for more information.
SplitPercent	This value can be -1 or any integer from zero (0) to 100. -1 means every text string will be split on a space. The integer is used to calculate the threshold for split and adjustment.

Option	Description
Class	Specifies the printer classification, such as AFP, PCL, XER, PST, or GDI. If you omit this option, the system defaults to the first three letters from the Module option. Some internal functions expect a certain type of printer. For instance, all 2-up functions require an AFP printer. The internal functions check the Class option to make sure the correct printer is available before continuing.
DisplayMode	This option lets you control how the PDF file is initially displayed. To have the PDF file open... - with bookmarks (document outline) visible, enter *UseOutlines* - with thumbnail images visible, enter *UseThumbs* -.in full-screen mode, with no menu bar or other window controls visible, enter *FullScreen* - in default mode, with neither bookmarks or thumbnails visible, enter *UseNone*
PaperSize	This option selects the paper size. The most commonly chose options are: 0=Letter (default) 1=Legal 2=A4 3=Executive 98=Custom For a list of all options, see Paper Sizes on page 13. When deciding the size the system first sets the page size to the size of the first image on the page. If the page size is *Custom*, the page size will be set to the form size. If the page size is now *Letter* (the default), the PDF Print Driver checks the PaperSize option. If the PaperSize is specified, the system uses it to determine the size. If the PaperSize option is set to *Custom* and page size is less than *Letter*, the page size will set to *Letter*. Otherwise, the system uses the custom width and height.
Linearize	Enter Yes to create linearized PDF files. See Creating Linearized PDF Files on page 25 for more information.
FontCompression	Use this option to compress embedded fonts. Enter zero (0), 1, 2, or 3 to indicate the level of compression. Zero indicates no compression and three indicates the highest level of compression. The default is two (2). Keep in mind that this option only compresses the ASCII portion of PostScript fonts, so the compression ratio is between 5-15%. For True Type fonts, the compression ratio is between 40-50%.

3 If you have a Printers control group, simply add this option to that control group:

```
PrtType = PDF
```

If you do not have a Printers control group, add this control group and option. For example, your Printers control group might look like this:

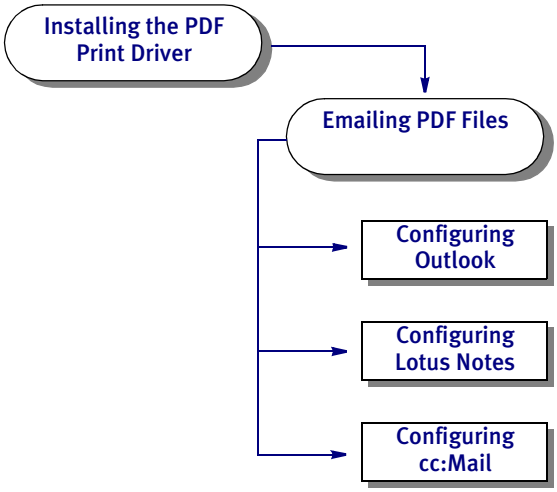
```
< Printers >
    PrtType = PDF
```

Your system can now use the PDF Print Driver. The following topic describes some additional features you can set up to work with the PDF Print Driver.

NOTE: Because of some features included in the PDF Print Driver, you must use Acrobat 5.x and above to open the PDF files it produces.

Additional Feature Setup

This illustration shows the various steps involved in setting up the PDF Print Driver, setting up your system to email PDF files, and configuring various email systems. It also shows how to set up the PDF Print Driver and then configure your system to send recipient output to multiple printers.



This table shows your options:

For information on	See	You must perform this task
Installing and configuring the PDF Print Driver	Installing the PDF Print Driver on page 2	Before you can email PDF files or print recipient output to multiple devices
Emailing PDF files	Emailing PDF Files on page 7	After setting up the PDF Print Driver
Configuring Microsoft Outlook	Configuring Outlook on page 8	After setting up the PDF Print Driver and setting up your system to email PDF files
Configuring Lotus Notes	Configuring Lotus Notes on page 8	After setting up the PDF Print Driver and setting up your system to email PDF files.
Configuring cc:Mail	Configuring cc:Mail on page 8	After setting up the PDF Print Driver and setting up your system to email PDF files

EMAILING PDF FILES

Once you set up the PDF Print Driver, make the following changes in your INI files to set up your system to email files.

NOTE: You can find additional information on email support in the Documaker Workstation Supervisor Guide.

Your INI files should have these settings:

```
< Printers >
    PrtType     = EPT
< PrtType:EPT >
    FileName    = F:\PDF\~KEYID ~DATE ; [%I-%M %p];.PDF
    InitFunc    = EPTInit
    KeepFile    = Yes
    Message     = PDF File attached...
    Module      = EPTW32
    PrintFunc   = EPTPrint
    PrtType     = PDF
    RecipFunc   = CSTSetMailRecip
    RecipMod    = CSTW32
    Subject     = PDF File from PPS
    TermFunc    = EPTTerm
```

The FileName option is where you specify the path and file name for the PDF file. Be sure to include the spaces and characters as noted.

In this example, the system writes a file to the \PDF directory on the F: drive. The file name includes the policy number (KeyID), the time the file was created, and an extension of *PDF*.

For example, the file written to disk might look like the one shown here:

```
F:\PDF\A100[3-47PM].PDF
```

When you finish with these changes, you can then perform one of the following tasks to configure your email system.

- Configuring Outlook on page 8
- Configuring Lotus Notes on page 8
- Configuring cc:Mail on page 8

Configuring Outlook

Once you set up the PDF Print Driver, configure your system to email PDF files, and follow the steps below, you can use the system to create a PDF file of a form set and then email that file via Outlook to a recipient.

Add these settings to your FSIUSER.INI file using a text editor:

```
< Mail >
    MailType    = MSM
< MailType:MSM >
    MailFunc    = MSMMail
    Module      = MSMW32
    Name        = MSMail
    Password    = (intentionally left blank)
    UserID      = (user ID to access Mail)
```

If you are unsure of the user ID, check your control panel for mail setting or with your Administrator.

Configuring Lotus Notes

Once you set up the PDF Print Driver, configure your system to email PDF files, and follow the steps below, you can use the system to create a PDF file of a form set and then email that file via Lotus Notes to a recipient.

Add these settings to your FSIUSER.INI file using a text editor:

```
< Mail >
    MailType    = VIM
< MailType:VIM >
    MailFunc    = VMMail
    Module      = VIMW32
    Name        = Notes (Name of the Notes Server)
    Password    =
    UserID      =
```

Configuring cc:Mail

Once you set up the PDF Print Driver, configure your system to email PDF files, and follow the steps below, you can use the system to create a PDF file of a form set and then email that file via cc:Mail to a recipient.

Add these settings to your FSIUSER.INI file using a text editor:

```
< Mail >
    MailType    = CCM
< MailType:CCM >
    MailFunc    = VMMail
    Module      = VIMW32
    Name        = CCMail (Name of the mail server)
    Password    =
    UserID      =
    DataPath    =  V:\ccdata (Default path for the email system)
```

Using the PDF Print Driver with GenPrint

The GenPrint program creates the print stream for each recipient batch and sends it to a printer output file. A *batch active* flag tells PRTLIB's installed output function to keep the current file open and to append each output group to the active stream, only closing the file at the end of the batch.

In most GenPrint processing situations, this is how you want it to work. For PDF, however, you need to break each recipient record into a separate file. The following topics describe how to send each form set to a separate file.

Changing the GenPrint Program

You can use the MultiFilePrint() callback function when running the Documaker Server (GenPrint) in multi-step mode. This callback function is included with the GenPrint program and lets you create a separate file for each transaction.

NOTE: To print multiple files when you are using Documaker Server in single-step mode, use the PrintFormset rule. For more information on this rule, see the Rules Reference.

To use this callback function, you must change the FSISYS.INI file as shown below:

```
< Print >
   CallbackFunc    = MultiFilePrint
   MultiFileLog    = { full path file name of a log file (optional) }

< RunMode >
   DownloadFAP     = Yes
   LoadFAPBitmap   = Yes
   CheckNextRecip  = No
```

Here is an example setup:

```
< Printer >
   PrtType         = PDF
< PrtType:PDF >
   Device          = e:\test.pdf
   DownLoadFonts   = No,Disabled
   LanguageLevel   = Level2
   Module          = PDFW32
   PageNumbers     = Yes
   PrintFunc       = PDFPrint
   SendOverlays    = No,Disabled
   SendColor       = Yes,Enabled
< Printer1 >
   PORT            = ..\RPEX1\DATA\BAT10001.PDF
< Printer2 >
   PORT            = ..\RPEX1\DATA\BAT20001.PDF
... (and so on)
```

Setting the CheckNextRecip option

When you use the PDF Print Driver, set the CheckNextRecip INI option to No (the default is Yes.) The GenPrint program uses this option to look ahead to subsequent recipient records and queue up recipient records that match the same form set.

This improves system performance when many recipient batch records are placed in the same print batch and sorted together. However, it is essential that each recipient record be viewed as a separate print transaction for this to work. Without this option disabled, each file will contain multiple recipients for the same form set, which is probably not what you want.

Using overlays

You cannot use overlays with the PDF Print Driver. There is no way to generate PDF overlays or use them at print time. Because of this, the GenPrint program ignores the SendOverlays option when it prints to the PDF Print Driver. FAP files and bitmaps must be loaded, which is indicated by setting these INI options:

```
DownloadFAP    = Yes
LoadFAPBitmap  = Yes
```

Using the MultiFilePrint Callback function

If you specify the MultiFileLog option in the Print control group, the specified file is created at the start of the GenPrint program when the callback is installed. The file is closed at the end of the GenPrint program when the callback is uninstalled.

At the end of each transaction, a new output file name is constructed and the GenPrint program's normal behavior of only outputting to one file is overridden. MultiFilePrint makes the following assumption about the output file name:

```
XXXX####
XXXX= four characters that are preserved.
#### = four characters set to a zero-filled sequence number.
```

MultiFilePrint assumes that the original print batch name ends in 0001. The second file opened will be 0002, and so on, up to 9999. MultiFilePrint assumes that no single recipient batch contains more than 9999 recipient batch records. If this is the case, a custom version of MultiFilePrint is required.

Avoid this approach, however, since this is a large number of output files to attempt to track and manage. MultiFilePrint does work with multiple print batches, and each batch can contain up to 9999 recipient records.

If you turned on logging, as each file is completed the system creates a log record in the log file you specified.

Using the log file

The log record has the following format:

```
;FIELD1;FIELD2;FIELD3;FIELD4;FIELD5;FIELD6;FIELD7;FIELD8;FIELD9;

FIELD1= Logical recipient batch file name
FIELD2 =Physical (full file name) recipient batch file
FIELD3= Group name 1 (e.g., Company)
FIELD4= Group name 2 (e.g., L.O.B.)
FIELD5= Group name 3 (usually empty)
FIELD6= TransactionId (e.g., Policy no)
FIELD7= Transaction type
FIELD8= Recipient type (as specified in POLFILE or FORM.DAT)
FIELD9= Print output file (full file name)
```

The log file is provided for use by a custom application and implementation to handle the management and distribution of the many individual output files.

Generating Separate Files

You can generate separate files for each transaction when you choose PDF (or RTF) from WIP or batch print.

The name of the files will have a rolling number appended to the end of the name that starts the process and is filled in on the Print window. This is automatically handled and you do not have to set INI options to get the WIP or batch print to work as long as your PrtType name is PrtType:PDF.

There are several INI options you can use to override the naming process and also name other print drivers that require this unique handling.

```
< BatchPrint >
    NoBatchSupport = PDF
    PreLoadRequired= PDF
```

These are the default settings and cannot be overridden. However, you can specify other PrtType print driver definitions you want to fall into these same categories.

Option	Description
NoBatchSupport	Indicates that the named PrtType items, separated by semicolon, do not really support batch transactions and require special handling.
PreLoadRequired	Lets you specify all the PrtType items, separated by semicolon, that should be forced to load the form set prior to the starting print. Most print drivers don't require this special requirement, but some, such as PDF do.

Also, you can name PrtType specific items under the BatchPrint control group to override the normal Device naming option. Here is an example:

```
< BatchPrint >
    PDF = ~HEXTIME .PDF
    RTF = ~HEXTIME -~KeyID .RTF
```

Any batch print sent to PrtType:PDF (picking PDF on the Print window) will override the name and store the current hexidecimal date and time, such as BCF09CA4.PDF, which is an eight-character name, as the name of each transaction's output.

Also, you can combine INI built-in calls as shown in the RTF example. Here any WIP or batch print sent to RTF will name the files using the HEXTIME and the KeyID from the WIP transaction. This will result in names similar to this: BCF099A4-123456.RTF

Note that you must leave a space after the built-in INI function name for it to work properly. That space will not appear in the resulting output name.

LIMITATIONS

The PDF Print Driver does not currently support the full set of Adobe Acrobat PDF capabilities. The following are a some of the product's limitations.

Type 1 fonts

If the PostScript Font Name/Setup Data setting in the FXR does not match a PDF base font, the PDF Print Driver maps the following PostScript font names into PDF base font names:

- Courier-Italic maps to Courier-Oblique
- Courier-BoldItalic maps to Courier-BoldOblique
- Univers-Medium maps to Helvetica
- Univers-Bold maps to Helvetica-Bold
- Univers-MediumItalic maps to Helvetica-Oblique
- Univers-BoldItalic maps to Helvetica-BoldOblique

Finally, if the PostScript font name fails to map to a PDF base font name using these rules, then fixed pitch fonts map to Courier and proportional fonts map to Helvetica. If a font has bold, italic or bold and italic attributes, the Courier or Helvetica PDF base font with corresponding attributes will be used.

Code pages

Currently, only the ANSI code page (also known as code page 1004) is supported for PDF files. Normally, this will only be an issue if you are trying to support international characters. If you are a DAP user and have used Monotype fonts for printing, this should not be an issue.

Page sizes

The PDF Print Driver currently supports these standard page sizes in PDF files:

- Letter (8.5in. x 11in.)
- Legal (8.5in. x 14ins)
- A-4 (8.26in. x 11.69in.)
- Executive (7.25in. x 10.5in.)

Portrait and Landscape page orientations are available for these standard page size. Custom page sizes will be converted into Letter size with corresponding orientation.

PDF objects

Although Acrobat Reader supports variable fields, radio buttons, push buttons, list boxes, and hypertext links, the PDF Print Driver does not support the creation of these objects within a PDF file.

Paper Sizes

Here is a complete list of all the paper sizes you can choose from in the PaperSize INI option:

For	Enter
US letter	zero (o). This is the default.
US legal	1
ISO A4	2
US executive	3
US ledger	4
US tabloid	5
US statement	6
US folio	7
US fanfold	8
ISO A0	20
ISO A1	21
ISO A2	22
ISO A3	23
ISO A5	25
ISO A6	26
ISO A7	27
ISO A8	28
ISO A9	29
ISO A10	30
ISO 2A	32
ISO 4A	34
ISO B0	40
ISO B1	41
ISO B2	42
ISO B3	43
ISO B4	44
ISO B5	45
ISO B6	46

For	Enter
ISO B7	47
ISO B8	48
ISO B9	49
ISO B10	50
ISO 2B	52
ISO 4B	54
ISO C0	60
ISO C1	61
ISO C2	62
ISO C3	63
ISO C4	64
ISO C5	65
ISO C6	66
ISO C7	67
ISO C8	68
ISO C9	69
ISO C10	70
ISO DL	71
JIS B0	80
JIS B1	81
JIS B2	82
JIS B3	83
JIS B4	84
JIS B5	85
JIS B6	86
JIS B7	87
JIS B8	88

For	Enter
JIS B	89
JIS B10	90
custom	98

CHAPTER 2

Customizing PDF Output

There are a number of options you can use to customize the PDF files you produce with the PDF Print Driver. This chapter discusses your options:

Setting PDF Compression Options

You can choose from these PDF compression methods:

Choose	For
0 (zero)	no compression
1	best speed
2	default compression
3	best compression

To override the default, add the Compression option in the PrtType:PDF control group in the DAP.INI file.

```
< PrtType:PDF >
    Compression   = 3
```

You can test the various compression options to see what works best for your implementation by comparing...

- The time difference between the request to view the transaction in Acrobat Reader and when it is displayed.
- The size of the PDF file after it is retrieved.

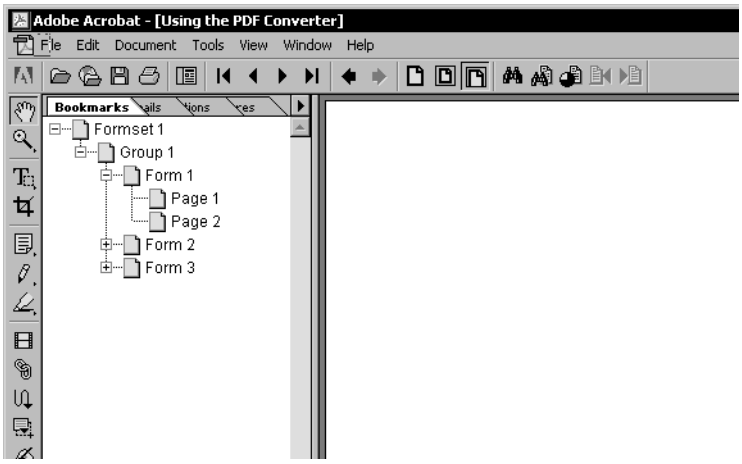
You can also control how much compression is used for fonts embedded into your PDF files. To do this, see the FontCompression option, discussed on page 4.

Setting Up Bookmarks

The PDF Print Driver sets up bookmarks at these levels by default:

- Form set - The text for this bookmark is the recipient name if applicable, otherwise it is the name of the form set.
- Group - The text for this bookmark is the name of the group.
- Form - The text for this bookmark is the description of the form. If there is no form description, the PDF Print Driver uses the name of the form.
- Page - The text for this bookmark is the name of the page.

If the text for any of the bookmarks is blank, the PDF Print Driver inserts text to describe the bookmark level, such as Formset, Group, Form, or Page, and the index for that level. Here is an example:



Creating custom bookmarks

The PDF Print Driver uses now lets you use custom rules to create custom bookmarks in the PDF file. To do this, you must create a custom rule.

With a custom rule, you can use the *extra info* in the FAP objects to store custom bookmark titles. Currently, the system uses *extra 1* and *extra 2*, leaving *extra 3* for bookmark titles.

If you choose do develop a custom rule to use *extra 3* for this purpose, keep these things in mind:

- The setting for the Bookmark option remains the same.
- The maximum length for a custom bookmark title is 128 characters.
- This feature is not a callback, so all bookmark settings using *extra 3* must be finished before you send them to the PDF Print Driver.
- The PDF Print Driver expects to receive a character string for extra 3 and the first eight characters must be *BOOKMARK*. The actual bookmark text begins with the ninth character and is NULL terminated.
- If you do not set *extra 3* (NULL handle) or the first eight characters are not *BOOKMARK*, the PDF Print Driver ignores *extra 3* and uses its original logic to create bookmarks.

Refer to the discussion of the FAPGetExtraInfo and FAPPutExtraInfo functions in the API documentation, available from the DOSS site, for information on getting and setting *extra 3*.

Here's how the system determines the text for a bookmark in a form set:

If you are filtering by recipient, the system...

1 Checks extra3 of the recipient (128 character limit).

2 Checks extra3 of the form set (128 character limit).

3 Check the recipient name (15 character limit).

If you are not filtering by recipient, the system...

1 Checks extra3 of the form set (128 character limit).

2 Checks the form set name, which cannot exceed eight characters.

Interfacing with Imaging Systems

The PDF Print Driver can add free form text or data at the beginning of a batch or each form set within the batch. This can help you interface with imaging systems such as RightFax.

Use the TEXTScript option to specify the DAL script you want to run. This DAL script creates a free form data or text buffer and adds it to the print stream.

Here is an example of the DAL script:

```
* Populate the PDF stream comment with these values from RCBDFD
faxnum = trim(GVM('FaxNumber'))
faxname = trim(GVM('FaxName'))

AddComment('<TOFAXNUM:' & faxnum & '>',1)
AddComment('<TONAME:' & faxname & '>',1)

Return
```

Notice the use of the second parameter to the AddComment DAL function. The `1` indicates the string should be an ASCII string. If you omit this parameter, the system converts the string into an EBCDIC string. You can also use the TEXTCommentOn option to tell the system to add free form text or data to the beginning of every form set or print batch. Here is an example:

```
< PrtType:PDF >
    TEXTScript     = imaging.DAL
    TEXTCommentOn  = formset
```

Producing Optimal PDF Output

To produce optimal PDF output, make the following changes in your font cross-reference (FXR) file.

NOTE: The tools used to customize a font cross-reference file are included with the full PPS system and in Documaker Server. For PPS runtime users, the people who create the libraries of form sets you use will have these tools and are responsible for setting up the FXR correctly.

- Remove font IDs built from the FORMSX font.

  This is not needed in Metacode conversions and there is no equivalent built-in Acrobat font. Furthermore, the original entry does not contain character width information.

- Fix point size settings for font IDs.

  Metacode fonts do not contain point size information. Therefore, a point size is approximated when a Metacode font is inserted into the FXR file. Sometimes, this approximation is incorrect. Many Metacode fonts include the point size as a part of its file name. For example, font ID 5 was built from the Metacode font AR07BP but is listed as having a point size of 8. However, AR07BP is really a 7-point font and the point size for this font ID should be changed to 7.00.

- Remove font IDs built from landscape fonts.

  For landscape fonts, where the equivalent portrait fonts are included in the FXR, the font IDs for the landscape fonts should be deleted and the landscape font name should be in the Rotated Fonts field of the portrait font. For example, font ID 4 was built from the landscape font AR07BL (Arial Bold 7-point). Font ID 5 was built from the portrait font AR07BP (Arial Bold 7-point). Therefore, font ID 4 was deleted and *;;AR07BL* was added to the Rotated Font Files field in the Metacode section of the Printers tab of the FXR file.

- Remove non-text fonts from the FXR file.

  You may decide to leave a non-text font in the FXR if you have an equivalent TrueType or PostScript font to embed and you have enabled font downloading. If not, remove the font IDs for non-text fonts.

  If the non-text font contains a signature or graphic, such as a company logo, convert the font to a logo (LOG) file. Fonts of this style have contiguous characters and are always referenced one way. For example, the font JOHNDO might contain only the letters *A*, *B*, and *C*. When the letters *ABC* are printed together using the JOHNDO font, the signature *John Doe* is printed. When a font is always used with a single contiguous set of characters, convert the font to a logo.

  For non-text fonts that contain characters which will be printed using a variety of combinations, you cannot use a logo. In this case, make the Xerox font available in the directory specified by the FontLib option in the MasterResource control group.

Examples of these types of non-text fonts include OCR, MICR, and barcode fonts. For example, a ZIP code (barcode) font produces a different picture when different ZIP codes are used. The ZIP code for 30309 looks different than the ZIP code for 49501. Using this approach, a bitmap image is created from the Xerox font using the specified characters (30309, 49501, and so on.) in the print stream. Only use this approach for fonts that are used infrequently because it results in larger PDF files which affects the time it takes to create and download a PDF file.

- Make sure color bitmaps are not saved as Comp TIFF or Comp Pack. These compression methods are not supported by PDF.

  The Comp TIFF format is designed to compress monochrome bitmaps, not color ones. The Comp Pack format is only useful when the color bitmap is 16 or 256 colors. There is no reason to use Comp Pack on a full-color (24-bit) bitmap.

  In most cases, you can simply leave full color bitmaps as JPEG or bitmap files and not convert them at all. The only reason to convert these files to the LOG format is to move them to a platform that does not support those file types, like MVS.

NOTE: The PDF driver supports monochrome, monocolor, 8-bit color (256 color) and 24-bit color logos.

- Run the FXRVALID utility to check the FXR file.

  The FXRVALID utility reports missing font files, incorrect built-in Acrobat font names, and so on. Correct any errors reported by the FXRVALID utility.

- Avoid specialty fonts when mapping to built-in Acrobat fonts.

  The built-in Acrobat fonts include Courier, Helvetica, and Times plus a couple of fonts containing non-text characters (Symbol and ZapfDingbats). Fonts such as Arial and Univers are pretty similar to Helvetica in terms of appearance and size and the built-in Acrobat font can often be used without any noticeable effect. Some font vendors also supply versions of the fonts where the characters are condensed (narrow) or expanded (wide). Although these fonts may have a similar character appearance, their size has been altered in a way that makes mapping to a built-in Acrobat fonts problematic.

- Use SplitText option when using built-in Acrobat fonts.

  When using built-in Acrobat fonts, many printer fonts are mapped to a single built-in Acrobat font. Many times, the printer fonts are not scaled perfectly in terms of their character widths. For example, the letter *A* in a 5-point printer font may have a width of 8 dots. Meanwhile, the letter *A* for the same font at 10-point (twice the size) may have a width of 14 dots (instead of 8*2 or 16 dots). This becomes a problem when mapping these non-scalable printer fonts to the built-in Acrobat fonts. The built-in Acrobat font is scalable and 10-point characters are twice the size as 5-point characters. You can use the SplitText and SplitPercent options to hide the differences that result from mapping non-scalable printer fonts into built-in (scalable) Acrobat fonts.

  If you set the SplitText option to Yes, every text string will be split and adjusted position according to the value of the SplitPercent option. The default is No.

The value of the SplitPercent option can be -1 or any integer between zero (0) and 100.

When you set this option to -1, every text string will be split and adjusted one position on a space. When you set this option to a positive value, such as 45, the differences between the character widths of the font used for a text string and character widths of the base font are accumulated.

Once the accumulated differences greater than 45/100*(width of the space character), the text string is split and a new accumulation is started. Therefore, smaller values for the SplitPercent option will produce better visual results at the cost of slightly slower performance and somewhat larger PDF files.

According to tests performed on a 55-page and a 100-page document, the size of the one with the SplitPercent option equal to -1 is about 12 percent greater than the one without any text split. The performance difference is hardly noticeable.

Creating Linearized PDF Files

Use the Linearize INI option to specify whether you want linearized or non-linearized PDF files.

A linearized PDF file is created so a browser can display the first page of the PDF file before it finishes loading the entire file. This lets those who are using the PDF file start working with it sooner. If you are creating large PDF files which may be accessed via the internet, you probably want to linearize them.

NOTE: Sometimes this is called “optimizing a PDF file.”

```
Linearize = Yes
```

Set this option to Yes to create a linearized PDF file. Setting this option to No tells the PDF Driver to produce a non-linearized PDF file.

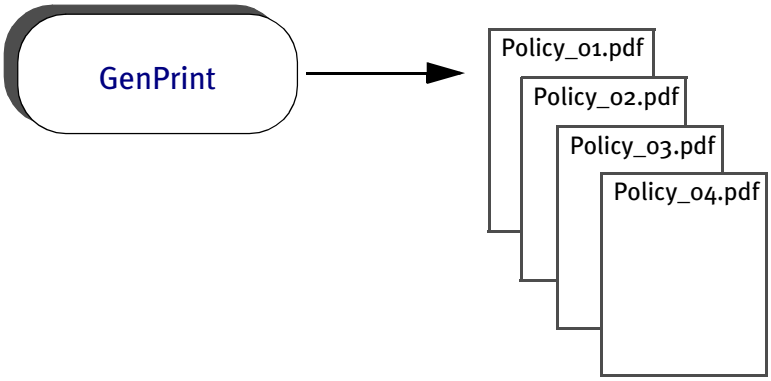
EXAMPLES

Here are some examples of how you can use the PDF Print Driver.

- Using the KeyID field to create one PDF file per transaction on page 26
- Naming recipient print streams in using single-step mode on page 28

Using the KeyID field to create one PDF file per transaction

This scenario shows how to set up your system so it will produce a single PDF file for each transaction and the PDF file will be named using the KeyID field and the PDF extension, such as: *policynumber.pdf*.



This scenario assumes:

- You are running Documaker Server in multi-step mode, with separate the GenTrn, GenData, and GenPrint programs executing separately
- The KeyID field (policy number) is unique throughout the entire batch run
- You have one recipient per transaction

To accomplish this, set up your system as described here:

1 Create a DAL library that contains the function to use to create the unique file name. Then copy the DAL library into DEFLIB. Assume the library illustrated below is named *SETPDFNM.DAL*.

Here is a sample DAL library that contains two DAL scripts, one to illustrate a simple way to accomplish this, and a more complex example that can handle more possible name collisions.

```
BEGINSUB SIMPLE_NAME
* This example presumes:
*   One occurance per run of a given policy number
*   Only one recipient per transaction
*
    IF HAVEGVM("PolicyNum")
        RETURN("data\" & GVM("PolicyNum") & ".pdf")
    END
    COUNTER += 1
    RETURN("data\Emptyfile" & COUNTER & ".pdf")
ENDSUB

BEGINSUB COMPLEX_NAME
* This example is more complex and assumes
*   many possible factors should be considered
```

```
*   to prevent a name collision
*
COUNTER += 1
IF HAVEGVM("Company")
    FILENAME = GVM("Company") & \
               GVM("Lob") & \
               GVM("PolicyNum") & \
               GVM("TransactionType") & \
               GVM("RCBRcpCode") & \
               GVM("RunDate") & \
               COUNTER
ELSE
    FILENAME = "emptyfile" & COUNTER
END
RETURN("data\" & FILENAME & ".pdf")
ENDSUB
```

2 Enable the DAL library to be loaded. In your FSISYS.INI file, add the following option:

```
< DALLibraries >
    LIB = SETPDFNM
```

3 Enable the multi-file print callback function and log file. In your FSISYS.INI file, include options similar to these:

```
< Print >
    CallbackFunc = CUSMultiFilePrint
    MultiFileLog = data\pdflog.txt
```

4 Make sure PDF print is enabled. In your FSISYS.INI file, include options similar to these:

```
< PrtType:PDF >
    Module = PDFW32
    PrintFunc = PDFPrint
< Printer >
    PrtType = PDF
```

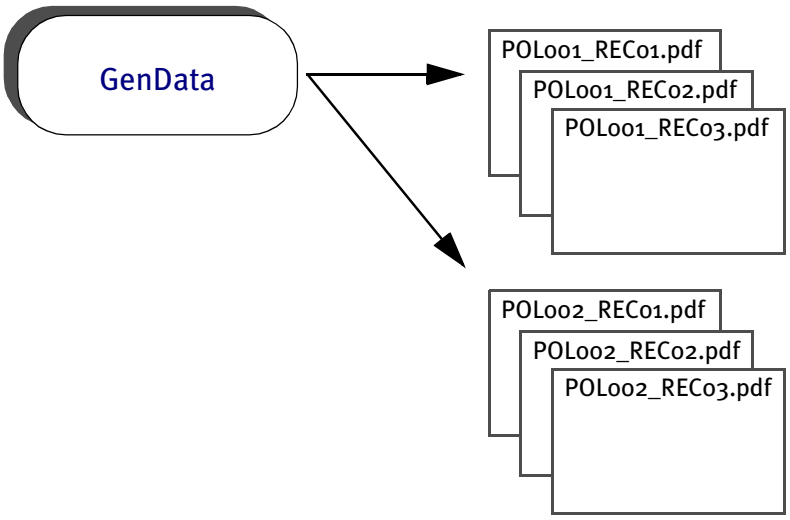
5 Make the Port option call the DAL function to get the name. In your FSISYS.INI file, change all of the Port options to something like this:

```
< Printer1 >
   Port = ~DALRUN SIMPLE_NAME
< Printer2 >
   Port = ~DALRUN SIMPLE_NAME
< Printer3 >
   Port = ~DALRUN SIMPLE_NAME
< Printer4 >
   Port = ~DALRUN SIMPLE_NAME
< Printer5 >
   Port = ~DALRUN SIMPLE_NAME
... and so on ...
```

When the GenPrint program runs, it will create a name using the KeyID (policy number) field.

Naming recipient print streams in using single-step mode

This scenario shows how to set up a master resource library (MRL) so it will produce a unique file name for each recipient of each transaction when running Documaker Server in single-step mode.



This scenario assumes:

- You have multiple recipients per transaction.
- You are using the XDB for its token lookup ability. (Using XDB is not required, it was just used in this example.)

To accomplish this, set up your MRL as described here:

1 Create a DAL library file in your MRL DefLib directory that contains the user-defined function to create unique file names. Assume the library file illustrated below is named: *unique_print_names.dal*.

This file contains two DAL user-defined functions; one illustrates using an XML file extract file, and a more complex example that can handle possible name collisions using a standard extract file. The DAL scripts are not extract specific.

```
* The xml_prt_names script assumes:
* Three unique global variables are defined in the RCBDFDFL.DAT file.
* The XDB has an entry called policy_number, whose xpath points to the
* policy number and each policy number is unique.
* Only three recipients exist.
* The PrintFormSet control group has the correct INI options defined.

BeginSub xml_prt_names
   prt_name = Lower(GetINIString(, "Printer", "PrtType"))
   pol_num1 = Trim(?("policy_number")) & ".insured." & prt_name
   pol_num2 = Trim(?("policy_number")) & ".agent."   & prt_name
   pol_num3 = Trim(?("policy_number")) & ".company." & prt_name
   SetGVM("PrtName001" ,pol_num1 ,,"C",25)
   SetGVM("PrtName002" ,pol_num2 ,,"C",25)
   SetGVM("PrtName003" ,pol_num3 ,,"C",25)
EndSub
```

```
* The std_prt_names script assumes:
* Three unique global variables are defined in the RCBDFDFL.DAT file.
* Only three recipients exist.
* The PrintFormSet control group has the correct INI options defined.

BeginSub std_prt_names
   cnt += 1
   prt_name = Lower(GetINIString(, "Printer", "PrtType"))
   pol_num1 = "insured." & #cnt & prt_name
   pol_num2 = "agent."   & #cnt & prt_name
   pol_num3 = "company." & #cnt & prt_name
   SetGVM("PrtName001" ,pol_num1 ,,"C",25)
   SetGVM("PrtName002" ,pol_num2 ,,"C",25)
   SetGVM("PrtName003" ,pol_num3 ,,"C",25)
EndSub
```

2 Make sure your FSISYS.INI or FSIUSER INI file has the following control groups and options defined. These examples assume you want to create PDF output.

Enable the DAL sub-routine library to be loaded:

```
< DALLibraries >
   Lib             = unique_print_names.dal
```

Enable the multiple file print function:

```
< PrintFormSet >
   MultiFilePrint  = Yes
   LogFile         = .\data\pdflog.dat
   RCBDFDField     = PrtName
```

Make sure PostScript print is enabled:

```
< Printer >
   PrtType         = PDF
< PrtType:PDF >
   Module          = PDFW32
   PrintFunc       = PDFPrint
```

Define the Port options as follows:

```
< Insured >
   Printer         = Insured
   Port            = .\print\.pdf
< Agent >
   Printer         = Agent
   Port            = .\print\.pdf
< Company >
   Printer         = Company
   Port            = .\print\.pdf
```

3 Make sure the RCBDFDFL.DFD file has the following global variables defined.

```
FieldName = PRTName001
FieldName = PRTName002
FieldName = PRTName003

< Field:PRTName001 >
   Int_Type   = Char_Array
   Int_Length = 26
   Ext_Type   = Char_Array_No_Null_Term
```

```
   Ext_Length = 25
   Key        = N
   Required   = N
< Field:PRTName002 >
   Int_Type   = Char_Array
   Int_Length = 26
   Ext_Type   = Char_Array_No_Null_Term
   Ext_Length = 25
   Key        = N
   Required   = N
< Field:PRTName003 >
   Int_Type   = Char_Array
   Int_Length = 26
   Ext_Type   = Char_Array_No_Null_Term
   Ext_Length = 25
   Key        = N
   Required   = N
```

4 Include the following rule in your AFGJOB.DAT file to call the DAL user-defined function before each transaction is executed.

```
/* Every form set in this base uses these rules. */
<Base Form Set Rules>
;NoGenTrnTransactionProc;;;
;PrintFormset;;;
;UseXMLExtract;;;
;ResetOvFlw;;;
;BuildFormList;;;
;PreTransDAL;;xml_prt_names();
```

Keep in mind you could also use this PreTransDAL rule:

```
;PreTransDAL;;std_prt_names();
```

If you want to use the *std_prt_names* DAL script.

When the GenData program runs, it creates print output files as follows. Assume the policy numbers are: *MVF 01-12-03* and *GRA 06-22-03*.

- When using the *xml_prt_names* DAL library function:

```
MVF 01-12-03.insured.pdf
MVF 01-12-03.company.pdf
GRA 06-22-03.company.pdf
...
```

- When using the *std_prt_names* DAL library function:

```
Agent.1.pdf
Insured.1.pdf
Company.1.pdf
Agent.2.pdf
...
```

CHAPTER 3

Working With Fonts

The fonts you use determine how the text on the page looks. With PDF files, you can choose to simply use the base fonts distributed with Acrobat Reader or you can embed the actual fonts used into the PDF file. The latter approach makes sure the document represented by the PDF file looks just like the original.

Embedding fonts also makes for larger PDF files, so if file size is a greater consideration than fidelity, you may want to choose not to embed fonts or to design the document with fonts similar to those distributed with Acrobat Reader.

This chapter discusses...

Using the Base Fonts

Adobe includes the following fonts with Acrobat Reader. You do not have to embed these fonts in PDF files.

Fixed Pitch Fonts	Proportional Fonts
Courier	Helvetica
Courier-Bold	Helvetica-Bold
Courier-Oblique	Helvetica-Oblique
Courier-BoldOblique	Helvetica-BoldOblique
	Times-Roman
	Times-Bold
	Times-Italic
	Times-BoldItalic
	Symbol
	ZapfDingbats

Embedding Fonts

The PDF Print Driver lets you embed fonts into the PDF print stream. This topic discusses when to embed fonts and when not to. It also describes how the system determines which base font to use or which custom font to embed.

Generally, you want the PDF Print Driver to reproduce your document so that it looks exactly as it did when you created it. Embedding fonts lets you accomplish this if you are using fonts that do not match the criteria discussed below.

When not to embed fonts

If you do not need to reproduce the exact look of the original document, you do not need to embed fonts. If you use the base Monotype (formerly Agfa) fonts and the FXR file Docucorp distributes, you do not need to embed fonts. The Monotype fonts are scalable and the Monotype Courier, Times, and Univers fonts are mapped to the names of the 14 base fonts Adobe distributes with Acrobat Reader.

NOTE: For a list of the base fonts, see Using the Base Fonts on page 32.

The Monotype Letter Gothic (fixed pitch, sans-serif) font is mapped to the base Adobe Courier (fixed pitch, serif) font. If you prefer the sans-serif look of Letter Gothic, you should embed that font.

Adobe also uses a standard scaling algorithm. If your implementation uses fonts that scale exactly as Adobe expects, you do not need to embed fonts. The PDF Print Driver determines the fonts to use and scales them for you. See Not Embedding Fonts on page 34 for more information.

In summary, you do not need to embed fonts if the fonts you are using ...

- Are already scalable
- Closely match the PDF base fonts

When to embed fonts

Embedding fonts lets you control the appearance of the document by letting you specify which fonts Acrobat Reader should use and what the font width will be. When you need to reproduce the exact look of the original document and you use custom fonts that do not scale exactly as Adobe expects, you should embed fonts.

To embed fonts, you need a set of PostScript Type 1 fonts or TrueType fonts, and you need to set the DownloadFonts INI option to *Yes*. You also need to run the FXRVALID utility to prepare your FXR file. For detailed instructions, see Using Embedded Fonts on page 35.

Not Embedding Fonts

If you are not going to embed fonts, you must set the following INI option to *No*, as shown here:

```
DownloadFonts = No
```

When you use a font that is not included in the 14 base fonts distributed with Acrobat Reader, the PDF Print Driver uses the information in the following fields, in this order, to determine what to do with the font:

- Setup Data field on the Other tab of the Font Properties window in the Font Manager. The system checks this field to see if its contents matches one of the 14 base Adobe fonts or an equivalent Monotype font name.
- Font Name field under PostScript on the Printer tab of the Font Properties window in the Font Manager. The system checks this field to see if its contents matches one of the 14 base Adobe fonts or an equivalent Monotype font name.
- TypeFace field on the Description tab of the Font Properties window in the Font Manager. The system checks this field to see if its contents matches one of the 14 base Adobe fonts or an equivalent Monotype font name.

Note: For a list of the base fonts, see Using the Base Fonts on page 32.

When the system matches a criteria, it then stops. If, after checking these fields, the system does not find information that matches one of the 14 base Adobe fonts or an equivalent Monotype font, it then maps...

- Proportional fonts to the Adobe Helvetica font (normal, bold, italic, or bolditalic)
- Fixed pitch or non-proportional fonts to the Adobe Courier font (normal, bold, italic, or bolditalic)

The system then checks these fields to determine additional font attributes:

- Spacing (fixed pitch or proportional)
- Style (italic or upright)
- Stroke Weight (bold or normal)

Using Embedded Fonts

If you are going to embed fonts, you must have either PostScript Type 1 or TrueType fonts. In addition, you must set the following INI option to *Yes*, as shown here:

```
DownloadFonts = Yes
```

You must also have the following information set up correctly for the PDF Print Driver to embed the font. If there is an error in any of the following fields, the PDF Print Driver will substitute one of the 14 base fonts using the criteria discussed in the topic, Not Embedding Fonts on page 34.

- Options field on the Other tab of the Font Properties window is set to one (1) if the field should be embedded or zero (0) if it should not be embedded.
- Font Index field on the Other tab of the Font Properties window specifies the width table to use for a group. Properly scaled font IDs have the same grouping value. This value is the font ID of one of the fonts in the group.
- Font File Name field on the Other tab and/or PostScript Font File Name field on the Printer tab. (Postscript => .PFB TrueType => .TTF). This field contains the file name of the PostScript or TrueType font you want to embed. This file should exist in the directory specified by the FontLib setting in your master resource library.

Note: The information stored in the A,R3 OTH record in the FXR appears in the fields on the Other tab of the Font Properties window in the Font Manager. You can edit the information there. The FXRVALID utility can also create this record. For more information about the FXRVALID utility, see the Docutoolbox Reference. You can access this manual from Docucorp's internet site (www.docucorp.com\doss).

Handling Fonts with Multiple Width Tables

For each font family, only one font with one width table (called base font) is created in PDF. So if a text string is using a font with a different width table from base font, the length of this string may be greater or smaller than anticipated and it may overlap with other text strings.

To solve this problem, you need to split text strings and adjust positions, so that overlapping between text strings can be avoided. To help you do this, the PDF Print Driver lets you use the following INI options:

```
< PrtType:PDF >
    SplitText      = Yes
    SplitPercent   =
```

If you set the SplitText option to *Yes*, every text string will be split and adjusted position according to the value of the SplitPercent option. The default is *No*.

The value of the SplitPercent option can be -1 or any integer between zero (0) and 100. When you set this option to *-1*, every text string will be split and adjusted one position on a space. When you set this option to a positive value, such as *45*, the differences between the character widths of the font used for a text string and character widths of the base font are accumulated.

Once the accumulated differences greater than 45/100*(width of the space character), the text string is split and a new accumulation is started. Therefore, smaller value for the SplitPercent option produce better results but also slow performance and result in larger files.

According to tests performed on a 55-page and a 100-page document, the size of the one with the SplitPercent option equal to -1 is about 12 percent greater than the one without any text split. The performance difference is hardly noticeable.

NOTE: You can also use the SplitText and SplitPercent INI options in the AFP print driver to eliminate differences in text positioning on 240 dpi and 300 dpi AFP printers.

Using Font Cross Reference Files

The quality of the created PDF files is in large part influenced by the setup information contained in the font cross-reference (FXR) file. Keep the following tips in mind when looking at your FXR file to optimize the quality of your PDF output.

PostScript font names should be present in your FXR, and all font IDs should contain one of the following PostScript font names in the Setup Data field for PostScript printing. The names of the PostScript fonts are case sensitive.

- Courier
- Courier-Bold
- Courier-Oblique
- Courier-BoldOblique
- Helvetica
- Helvetica-Bold
- Helvetica-Oblique
- Helvetica-BoldOblique
- Times-Roman
- Times-Bold
- Times-Italic
- Times-BoldItalic
- Symbol
- ZapfDingbats
- Courier-Italic
- Courier-BoldItalic
- Univers-Medium
- Univers-Bold
- Univers-MediumItalic
- Univers-BoldItalic

The point size value should be present and should be within 33% of the font height. Font heights are measured in 2400 dots per inch while point sizes are measured in 72 dots per inch, so some conversions to equivalent units will be necessary to determine their relative values.

NOTE: All of the fonts listed above, except for the Univers fonts, are included in Adobe's Acrobat Reader and do not have to be embedded in PDF files.

The spacing value (either fixed or proportional) should be present and accurate. Here is a list of PostScript fonts sorted by spacing value.

Fixed Pitch Fonts	Proportional Fonts
Courier	Helvetica
Courier-Bold	Helvetica-Bold
Courier-Oblique	Helvetica-Oblique
Courier-BoldOblique	Helvetica-BoldOblique
Courier-Italic	Times-Roman
Courier-BoldItalic	Times-Bold
	Times-Italic
	Times-BoldItalic
	Univers-Medium
	Univers-Bold
	Univers-MediumItalic
	Univers-BoldItalic
	Symbol
	ZapfDingbats

The font style value (upright or italic) should be present and accurate; it should match a PostScript font with an equivalent font style.

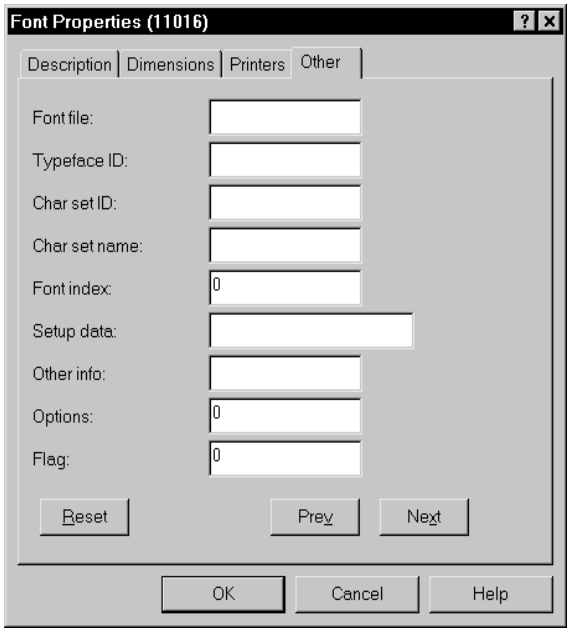
The font weight (bold or normal) should be present and accurate; it should match a PostScript font with an equivalent font weight.

How to embed fonts

Follow these instructions to embed fonts.

NOTE: You can embed TrueType or PostScript fonts. The PDF Print Driver uses the font file extensions to distinquish between the two types of fonts. TrueType fonts have a *TTF* extension. PostScript fonts have a *PFB* extension.

1. Use a text editor to open the INI file. Set the DownloadFonts option in the PrtType:PDF control group to Yes.
2. Next, use Font Manager to set up your font cross reference file (FXR). Start Font Manager and select the font cross reference file you want to edit. Then select the font you want to embed and click Edit.
3. On the Font Properties window, use the Other tab to set up the font for downloading.



- Enter 1 in the Options field to indicate that the font should be downloaded (a zero (0) tells the system not to download the font).
- Use the Font Index field to group fonts.
- Enter the name of the font file in the Font File field.

4 Set up the remaining fields (Char set ID, Setup Data, and so on) as you would for a PostScript font.

Keep in mind...

- If you set the DownloadFonts option to No, all fonts used in the image are mapped to one of the 14 Type 1 fonts distributed with the system and no fonts are downloaded.
- If you do not define the fields on the Other tab, the system will not download the font—even if you set option to Yes. Instead, the system will map the font to one of the 14 Type 1 fonts.
- For symbol fonts, such as DocuDings, make sure the Char set ID field on the Other tab is set to *WD*.

If you set the DownloadFont option to Yes,

- The system will download fonts used in the document as long as the Options fields on the Other tab are set to 1.
- The system will map the fonts used in the document to one of the 14 Type 1 fonts if the Options fields on the Other tab is set to zero (0).
- Downloadable fonts with the same value in their Index fields are considered in the same group. Only one font will be downloaded (embedded) for each group.

NOTE: Each font you embed increases the PDF file size by approximately 40kb. See FontCompression on page 4 for information on compressing fonts.

CHAPTER 4

Adding Security to PDF Files

You can make your PDF documents more secure by encrypting them and by adding security settings. Secure PDF documents may be required for electronic bills, confidential documents (like medical records), and other documents containing sensitive material (bank statements, loan applications, and so on).

Security settings let you assign passwords for opening and modifying the document and control access to printing, editing, and annotating the document.

A document can have both an open password and an owner password and these passwords can consist of up to 32 characters. You can view a document with either type of password, but you must enter the owner password to change the document or its security settings.

If you use the security settings to restrict access to certain features, all toolbar and menu items related to those features are dimmed in Adobe Reader or Acrobat.

To enforce these restrictions, the content of the document is encrypted using a 40- or 128-bit algorithm specified by Adobe.

This chapter provides information on the following:

Configuring the Security Features

The PDF security features are built into the PDFLIB library so there are no files to install.

Configuring the INI Files

In each PrtType:XXX control group for which you want security features, add this INI option:

```
< PrtType:PDF >
    Encrypt = Yes
```

This option tells the PDF library to load the PDFCRYPT library and encrypt PDF files. Encrypting the PDF file changes the file so it is no longer easy to read when transmitted over the network. It also means you cannot use a text editor to alter the file without destroying it.

Since, however, no passwords or permissions have been specified, this option provides only a minimal amount of security. If someone gets a copy of the file, there is no way to prevent that person from viewing the file in Acrobat Reader or altering it with the full Acrobat product.

You can, however, create more secure documents by including additional options in the PDF printer control group of your INI file.

In the same control group as the Encrypt option, you can add the SecurityGroup option to specify a control group which will contain the permissions, passwords, and encryption strength. Here is an example:

```
< PrtType:PDF >
    SecurityGroup = PDF_Encryption
    Encrypt = Yes
< PDF_Encryption >
    .
    .
```

NOTE: While the only way to specify passwords is through INI options, the PDFKEYGEN built-in function lets you create a custom built-in that supplies the actual passwords. This way, the only thing that the INI file contains is

```
OwnerKey = ~PDFKEYGEN ~CUSTOMPASSWORDRULE
```

Keep in mind the INI files are generally inside a firewall and the passwords are stored in encrypted form.

Setting Up a Security Control Group

The SecurityGroup option specifies the control group where permissions, passwords, and encryption strength are set. The permissions let you control if users can...

- Print the document. (You can also control the print quality.)
- Modify the document.
- Copy text or graphics to the clipboard.
- Add or update annotations.

- Fill in form fields.
- Access the document with accessibility tools, such as text to speech applications.
- Add navigational aids to the document.

This table shows the options you can set. All AllowXXX options default to Yes, meaning the permission is granted. See also Understanding permissions on page 45.

Option	Description
AllowPrinting	Lets the user print the document.
AllowModify	Lets the user modify the document.
AllowCopy	Lets the user copy data to the clipboard.
AllowAnnotate	Lets the user add or update annotations.
AllowFormFields	Lets the user fill in form fields. You must set the KeyLength to 128 to use this option.
AllowAccessibility	Lets accessibility tools, such as text to speech applications, access the document. You must set the KeyLength to 128 to use this option.
AllowAssembly	Lets the user add navigational elements to the document. You must set the KeyLength to 128 to use this option.
AllowHighQualityPrinting	If you set AllowPrinting to Yes, set this option to Yes to let the user generate a high quality hard copy of the document. If you set AllowPrinting to Yes and this option to No, users can only print draft quality hard copy. You must set the KeyLength to 128 to use this option.
OwnerKey	Specifies the password required to change the document or its security settings. The password takes the form of an encrypted 64-byte hexadecimal encoded string. See Choosing passwords on page 44 for more information.
UserKey	Specifies the password required to open the document. The password takes the form of an encrypted 64-byte hexadecimal encoded string. See Choosing passwords on page 44 for more information.
KeyLength	Specifies the encryption strength, in bits, either 40 or 128. The default is 128. The key length must match the length provided to the PDFKey tool. The choice of key length is a primary factor in determining how secure the PDF file will be. Computers are fast enough that 40-bit encryption is susceptible to attacks in which every possible encryption key is tried. The 128-bit encryption is much more secure and allows finer control of permissions, but PDF files encrypted with a key length of 128 bits can only be viewed using Adobe Acrobat and Acrobat Reader 5.0 or later. Because Acrobat Reader is available at no cost, this restriction is generally no more than a minor inconvenience.

Choosing passwords

An encrypted version of the open and owner passwords as well as permission information is kept in the security control group. These take the form of 64-byte hexadecimal encoded strings. Here is an example:

```
6d6f62d768bc143cefa30fc4fd3cc00eb1f638157f1d985dd5fe8ebfaa7c8317
```

There are two ways to generate these keys:

- The PDFKey tool generates these keys (see Using the PDFKey Tool on page 50 for more information). Because the permissions are encoded in the keys, the permissions in the Security control group *must* match those provided to the PDFKey tool.
- You can also use the PDFKEYGEN built-in function to generate the OwnerKey and UserKey information. See Using the PDFKEYGEN Function on page 51 for more information.

You can set passwords any way you like and they can consist of up to 32 characters. Choosing the password is up to you. You can use special characters in passwords but, when using the PDFKey utility, you have to enclose the entire password in quotation marks (“) just as you would if the password contains spaces. Here is an example:

```
pdfkw32 /U="~!@#$%^&*()_+"      /O="?<>{}[]-/\|"
```

This table describes the levels of security you get based on the passwords you set up:

Add this password		
Owner	**Open**	**To provide this level of security**
Yes	Yes	You must enter the owner or open password to view the document. Users are bound by the assigned permissions You cannot change the security settings without the full Acrobat product and the owner password.
Yes	No	No password is required to view the document. Users are bound by the assigned permissions. You cannot change the security settings without the full Acrobat product and the owner password.
No	Yes	You must enter the open password to view the document Anyone with the full Acrobat product and the open password can view and change the PDF file, including the file’s security settings.
No	No	No password is required to view the document. Users are bound by the assigned permissions. Anyone with the full Acrobat product can change the security settings — so it makes little sense to set passwords in this manner.

NOTE: If you forget a password, there is no way to retrieve it from the document. Be sure to store passwords in a secure location in case you forget them.

Understanding permissions

This table shows how the permissions are related.

If this is set to True	Attempts to set the following to False are ignored
AllowModify	AllowAnnotate, AllowFormFields, and AllowAssembly
AllowAnnotate	AllowFormFields and AllowAssembly
AllowFormFields	AllowAssembly

This table provides a general overview of how permissions are related.

If this is set to True	Attempts to set the following to False are ignored
AllowModify	AllowAnnotate, AllowFormFields, and AllowAssembly
AllowAnnotate	AllowFormFields and AllowAssembly
AllowCopy	AllowAccessibility
AllowFormFields	AllowAssembly

NOTE: PDF files with security settings are called *secure* files. You cannot insert a secure PDF file into another PDF file. You can, however, insert a nonsecure PDF file into a secure PDF file. In this case, the nonsecure PDF file inherits the secure PDF file's security settings.

These tables show all permitted combinations of permissions for Adobe PDF files. If you set permissions differently than those shown below, Adobe changes them based on the information shown in these tables.

KeyLength	Modify	Annotations	FormFields	Assembly
40	Allowed	Allowed	Allowed	Allowed
40	Not allowed	Allowed	Allowed	Allowed
40	Not allowed	Not allowed	Allowed	Allowed
40	Not allowed	Not allowed	Not allowed	Allowed
128	Not allowed	Not allowed	Not allowed	Not allowed
128	Allowed	Allowed	Allowed	Allowed
128	Not allowed	Allowed	Allowed	Allowed
128	Not allowed	Not allowed	Allowed	Allowed
128	Not allowed	Not allowed	Not allowed	Allowed
128	Not allowed	Not allowed	Not allowed	Not allowed

KeyLength	Copy	Read Access
40	Allowed	Allowed
40	Not allowed	Allowed
40	Not allowed	Not allowed
128	Allowed	Allowed
128	Not allowed	Allowed
128	Not allowed	Not allowed

KeyLength	Print
40	High resolution
40	None
128	High resolution
128	Low resolution
128	None

While testing the security feature of the PDF Print Driver, Docucorp has noted an issue with Adobe's products in which permissions set correctly in the PDF Print Driver are displayed incorrectly on the Adobe security settings window. It appears the permissions work as expected, regardless of whether those settings display correctly.

There are, however, a couple of caveats that do not follow these permitted combinations:

For a 40-bit encryption

Adobe lets you set Annotation to *Not allowed*, even though Modify was set to *Allowed*. Below are the permission settings for this caveat.

```
Encryption     = Yes
KeyLength      = 40
Print          = High resolution
Modify         = Allowed
Annotation     = Not allowed
Form Fields    = Allowed
Assembly       = Allowed
Copy           = Not allowed
Accessibility  = Not allowed
```

In addition...

- When you set AllowModify to *Allowed*, AllowAnnotations remains *Not allowed*, although it should be *Allowed*.
- When you set AllowAnnotations to *Allowed*, AllowAssembly remains *Not allowed*, although it should be *Allowed*.
- When you set AllowFormFields to *Allowed*, AllowFormFields and AllowAssembly both remain *Not allowed*, although they should be *Allowed*.

- When you set AllowAccessibility to *Allowed*, it remains *Not allowed*.
- When you set AllowAssembly to *Allowed*, it remains *Not allowed*.

For a 128-bit encryption

Adobe lets you set Modify to *Allowed*, even though Annotation and Form Fields were set to *Not allowed*. Below are the permission settings for this caveat.

```
Encryption     = Yes
KeyLength      = 128
Print          = Low resolution
Modify         = Allowed
Annotation     = Not allowed
Form Fields    = Not allowed
Assembly       = Allowed
Copy           = Not allowed
Accessibility  = Not allowed
```

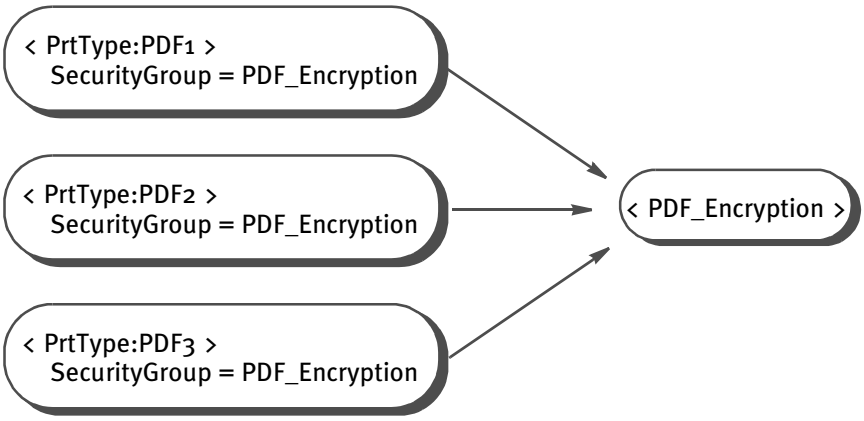
In addition...

- When you set AllowModify to *Allowed*, AllowAnnotations and AllowFormFields both remain *Not allowed*, although they be *Allowed*.
- When you set AllowCopy to *Allowed*, AllowAccessibility remains *Not allowed*.
- When you set AllowAnnotations to *Allowed*, AllowAssembly remains *Not allowed*, although it should be *Allowed*.
- When you set AllowFormFields to *Allowed*, AllowAssembly remains *Not allowed*, although it should be *Allowed*.

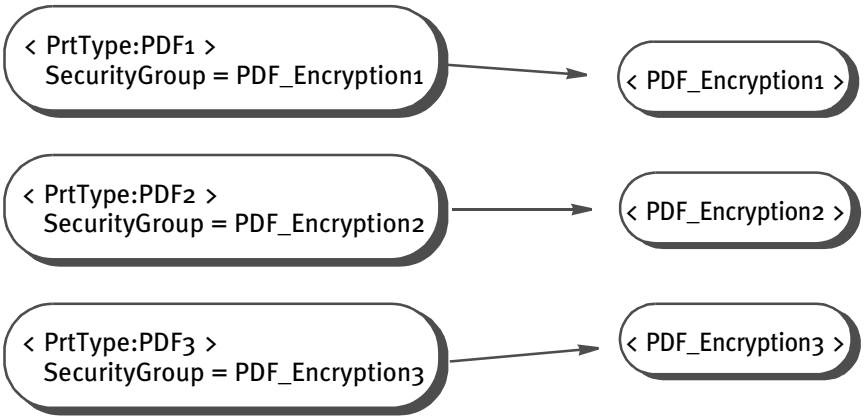
NOTE: If you encounter any other unusual combinations, you may want to contact Adobe.

Setting up multiple security groups

You can set up your PDF PrtType:XXX control groups to share a common set of security settings or have security settings tailored to each control group. This illustration shows how you would set your printer options to use the same security settings:



Or, you can have custom security settings for each printer control group:



Choose the approach that works best in your situation.

Setting up document-level security

You can implement unique passwords at the document level. All you have to do is provide a custom *~PDFPASSWORD* function (the name can vary).

For each recipient in each transaction in each batch, the GenPrint program calls the PDFPrint function in the PDFW32.DLL file. The PDFPrint function checks the Encrypt option in the PrtType:PDF control group to see if encryption is desired.

If the Encrypt option is set to Yes, the PDFPrint function calls the PDFCryptInit function to an encryption *context*. If no passwords are passed to the PDFCryptInit function, as is the case with the PDF Print Driver, it checks the OwnerKey option:

```
< PDF_Encryption >
    OwnerKey  =
```

Keep in mind that the SecurityGroup control group must be specified in this option:

```
< PrtType:PDF>
    SecurityGroup = PDF_Encryption
```

If the OwnerKey option is set to *~PDFKEYGEN ~PDFPASSWORD*, the INI built-in function calls the ~PDFPASSWORD function, then passes the result to the ~PDFKEYGEN function.

NOTE: The ~PDFPASSWORD function varies for each implementation and is usually part of CUSLIB.

Using the PDFKey Tool

Use the PDFKey tool to generate the encrypted passwords used in the security control group.

Program names

Windows 32-bit	PDFKW32.EXE
MVS	See the Documaker Server Installation Guide

Place the executable file (PDFKW32.EXE) in the directory which contains the other Docucorp binary files.

Syntax

pdfkw32 /U /O /K /P /F /M /C /N /R /A /? /H

Parameter	Description
/U	Enter the password required to open the document. You can enter up to 32 characters. Passwords are case sensitive.
/O	Enter the password required to modify the document or its security settings. You can enter up to 32 characters. Passwords are case sensitive.
/K	Enter either 40 or 128 to specify the length of the encryption key. The default is 128.
/P	Enter No to prevent users from printing the file, L to permit low quality printing, or H to permit high quality printing.
/F	Enter No to prevent users from entering form fields. The default is Yes.
/M	Enter No to prevent users from modifying the document. The default is Yes.
/C	Enter No to prevent users from copying text from the document to the clipboard. The default is Yes.
/N	Enter No to prevent users from annotating the document. The default is Yes.
/R	Enter No to prevent users from using reader accessibility tools to view the document. The default is Yes.
/A	Enter No to prevent users from adding navigation aids, such as bookmarks. The default is Yes.
/? or /H	Enter either of these parameters to display information about the syntax and parameters you can use with this utility.

All parameters are case insensitive and can be preceded with either a backslash (\) or a dash (-). Omit spaces between an argument and its value. Passwords can, however, contain spaces. Simply enclose the entire password parameter in quotation marks, as shown here:

```
pdfkw32 “/O=Password With Spaces”
```

For more information about passwords see Choosing passwords on page 44.

Using the PDFKEYGEN Function

You can use the PDFKEYGEN built-in function to generate the OwnerKey and UserKey information.When you include a string in this form...

```
OWNERPASS/USERPASS
```

this built-in function generates the OwnerKey and the UserKey values. The OwnerKey is returned via the function call. The UserKey is added to the INI context. To omit one or both of the passwords, modify the string as shown here:

```
"/USERPASS"
 "OWNERPASS/"
 "/"
```

To use the PDFKEYGEN built-in, first create a custom built-in that returns the password string in the format described above.

To call PDFKEYGEN (assuming the custom built-in function is registered as PDFPASSWORD), specify the OwnerKey INI option as shown here:

```
OwnerKey = ~PDFKEYGEN ~PDFPASSWORD
```

You can omit the UserKey option since it will be supplied via the above call.

NOTE: You can find information on writing and registering built-in functions in the API documentation at:

https://pd.docucorp.com/support/doc/rel102/api/coralib/INISyste.htm#INIRegisterFunction

Example Security Settings

Here are some examples of how you can set up your PDF security configuration. The examples include choosing permissions, passwords, and encryption strengths; generating keys with the PDFKey tool; and setting the appropriate INI options.

Example 1

This example shows how to set up:

- 128-bit encryption security
- An owner password of *Docucorp*
- No password required to open the document
- Permissions that prevent readers from modifying information or copying information to the clipboard

Here is how you would run the PDFKey tool:

```
PDFKW32 /O=Docucorp /K=128 /M=N /C=N
```

The output of the PDFKey tool is:

```
OwnerKey: e551db056412e608eeab3e5f96619ac9296b49e419467fdb8879f75f95c2a816
UserKey: 605e0bb040c9ce39d650e718b3127f9b10000000fc00000000000000000000000
```

Here are the additions you would make to your INI file, assuming that the printer type control group is called PrtType:PDF.

```
< PrtType:PDF >
  Encrypt = Y
  SecurityGroup = PDF_Encryption_Example_1
< PDF_Encryption_Example_1 >
  KeyLength = 128
  OwnerKey = e551db056412e608eeab3e5f96619ac9296b49e419467fdb8879f75f95c2a816
  UserKey = 605e0bb040c9ce39d650e718b3127f9b10000000fc00000000000000000000000
  AllowModify = No
  AllowCopy = No
```

You do not have to enter a password to view documents generated with these INI options. To modify a document, however, or to copy information from it to the clipboard, you must enter the owner password. Also, because 128-bit encryption option is used, you must have Acrobat 5.0 or later to view these documents.

Example 2

This example shows how to set up:

- 40-bit encryption security
- An owner password of *Docucorp*
- A open password of *EncryptionIsFun*
- No permission restrictions

Here is how you would run the PDFKey tool:

```
pdfkw32 /o=Docucorp /u=EncryptionIsFun /k=40
```

The output of the PDFKey tool is:

```
OwnerKey: 90ed4c70598767459de9523a9ce7e77ac51f4459257401fbae1936b6b1bbc7fd
UserKey: f6ac1d4b26000000000000000000000020000005800000000000000000000000
```

Here are the additions you would make to your INI file:

```
< PrtType:PDF >
  Encrypt = Yes
SecurityGroup = PDF_Encryption_Example_2
< PDF_Encryption_Example_2 >
  KeyLength = 40
  OwnerKey = 90ed4c70598767459de9523a9ce7e77ac51f4459257401fbae1936b6b1bbc7fd
  UserKey = f6ac1d4b2600000000000000000000002000000058000000000000000000000000
```

To view documents generated with these settings, you must enter the password *EncryptionIsFun* when prompted by Acrobat. Once the document opens, there are no restrictions.

To change the security settings, you must enter the owner password *Docucorp*. Because 40-bit encryption is used, you only need Acrobat 4.0 or later to view documents created with these settings.

Example 3

This example shows how to set up:

- 128-bit encryption security
- No owner password
- A open password of *AnythingGoes*
- No permission restrictions

Here is how you would run the PDFKey tool:

```
PDFKW32 /u=AnythingGoes
```

Since the key length defaults to 128, the keys are:

```
OwnerKey: 87958ea2053c6e89302ba926dfd78a2b3d0213d8e9569734d3045a8be297370e
UserKey: 75a73844c09078ad1a587957d4328e3410000000830000000000000000000000
```

Here are the additions you would make to your INI file:

```
< PrtType:PDF >
  Encrypt = Yes
SecurityGroup = PDF_Encryption_Example_3
< PDF_Encryption_Example_3 >
  KeyLength = 128
  OwnerKey = 87958ea2053c6e89302ba926dfd78a2b3d0213d8e9569734d3045a8be297370e
  UserKey = 75a73844c09078ad1a587957d4328e3410000000830000000000000000000000
```

To view documents created with these settings, you must enter the password *AnythingGoes* when prompted. You then have unrestricted use of the document. Since there is no owner password, you can even modify security options. Because you are using 128-bit encryption, you must have Acrobat 5.0 or later to open the document.

Tips

In case you run into problems, keep in mind...

- If you cannot open a password-protected PDF file after supplying the *correct* password, this probably indicates you have errors in your setup for producing secure PDF files.

  The command line parameters used with the PDFKey tool, which produces the OwnerKey and UserKey hex strings, must match the secure PDF INI settings used when the PDF file is produced. Check your secure PDF INI settings for problems such as misspellings and INI settings that do not match the parameters used when running PDFKey. For example, specifying a 40-bit key length to PDFKey (/K=40) and using a KeyLength=128 INI setting will produce a PDF file that will not open in Acrobat.

- If you set the Print option (/P) to No in the PDFKey tool, you cannot set the AllowHighQualityPrinting option to No in the INI file. If you do, the result is an error, such as the one described above.

- When generating the hex key using the PDFKey tool, the system prints the output in INI format, as shown here:

  ```
  D:\rel103>pdfkw32
  < PDF_Encryption >
  KeyLength = 128
  OwnerKey = 36451bd39d753b7c1d10922c28e6665aa4f3353fb0348b536893e3b1db5c579b
  UserKey = 7880927481fd184b32c0efb547ef5adb100000006c0000000000000000000000
  ```

  The output is formatted this way so you can copy and paste these settings into your INI file.

- When configuring PDF encryption for use with IDS, you must modify two INI files: DAP.INI and the INI file for the MRL. The DAP.INI file contains all of the SecurityGroup names and SecurityGroups for all of the MRLs for which encryption is needed. The MRL's INI file contains only the SecurityGroup name and settings used by that MRL.

  You can also set up multiple SecurityGroups within one MRL. Both the DAP.INI and MRL's INI file would contain the names of all SecurityGroups in the PrtType:PDF control group as well as the individual SecurityGroups.

Index

A

B

C

D

E

F

G

I

J

K

L

M

N

O

P

R

S

T

U