XML Professional Publisher:

PDF Support in XPP

for use with XPP 9.7
April 2024

Legal Notice

Contents

Figures

Tables

About This Manual

PDF Support in XPP describes the process for converting an XPP division to Adobe Portable Document Format (PDF) using PDF macros.

This process produces PDF files in a reliable, easily managed manner, letting you maintain a single source from which to produce both printed and electronic output.

Using the options you select on the PDF tab in the Print dialog box, XPP processes jobs and produces either PDF files or PDF-enhanced PostScript files for use with a PDF conversion tool. These files contain elements such as links from tables of contents and indexes, cross-document links, bookmarks, article beads (or article threads), and notes. This eliminates the need to manually add links or other elements with a PDF editor, such as Adobe Acrobat.

Where Do I Start?

This document is for all users who want to convert XPP jobs into PDF documents.

If you need to...	*Then read ...*
Get an overview of PDF and XPP support.	Chapter 1
Convert an XPP document to PDF.	Chapter 2
Insert PDF XyMacros in XPP jobs for conversion to PDF markup.	Chapter 3
Set up the PDF Support Spec and the *_distill_parms*, *gsdistill*, or *_gsdistill_parms* files.	Chapter 4
Set up associated XPP Specs and process jobs to convert to PDF, including: • the Job Ticket • the Document Assembly Ticket • the CI Spec • the Background Queue Spec • Device Options file	Chapter 5
Perform job processing to create a PDF document, including: • Checking cross-references across the job • Verifying PDF markup in a job with *showpdf*. • Activating job processing to create a PDF-enhanced PostScript file for PDF generation.	Chapter 6
Understand *showpdf* progress and error messages.	Appendix A
Understand which PDF conversion options Ghostscript supports.	Appendix B

Conventions Used in This Manual

This manual uses a set of symbolic, typographic, and terminology conventions to categorize specific information. Take a few moments to become familiar with these conventions before you use this manual.

Convention	*Description*
Bold	Bold type, used in procedures, indicates the object of the action. It may be a menu option, a push button, or a field, and so forth. For example, "select **Open**" means select the menu option called Open. Position **cursor** means to move the cursor to a specific location. Enter appropriate **information** means that you enter information that is appropriate for your site or for the specific job. Bold may be used elsewhere in the manual to denote emphasis.
Key	Capital first letter and the word "key" indicates a key on the keyboard. Capital first letter and the words "Softkey menu" indicate the menu pad to the right of the XyView. Unless otherwise indicated, this manual assumes that you press the Enter key at the end of a command line.
Key+Key	This sequence indicates a Shortcut Key combination. Hold down the first key while also pressing the second key, that is **ALT+F4** means to hold down the *Alt* key while also pressing the *F4* key.
`Message`	A monospaced typeface indicates an application's response to your action. For example, "the message `Enter Value` appears" means that the application displays the words "Enter Value."
Italics	In running text, an italic typeface indicates a new term; for example, "The replacement string of characters is the *output string*." In a system message, a field entry, or an argument to a command, an italic typeface indicates a variable. For example, *filename* is a variable in the message "Can't open *filename*." Italics are also used for the names of programs, such as *Perl*.
" "	Quotation marks indicate that you should enter exactly what the instructions tell you to enter. For example, type "**yes**" means to type the letter **y**, the letter **e**, and the letter **s**.
[ ]	Reverse-video square brackets represent tags in standard XPP. Tags are general-purpose commands defined in the Item Format Spec and embedded in a document. They generally format logical components of text, such as chapter openings, headers, or lists. For example, the tag for beginning a chapter might be **[chap]**.

Convention	*Description*
《 》	Reverse-video angle brackets represent XPP-supplied macros (XyMacros) and user-defined macros. XyMacros are commands embedded in text to set or change formatting or typographic style. For example, the XyMacro to end a page is **《ep》**. Reverse-video angle brackets also represent tags when you use XPP in either XML or SGML mode. Note that when in XML or SGML mode, XPP does not use the conventional reverse-video square brackets.
《?xxx》	Reverse-video angle brackets with a beginning question mark represent macros when using XPP in SGML mode.
《?xxx?》	Reverse-video angle brackets with a beginning and ending question mark represent macros when using XPP in XML mode.

When entering values for some arguments in macros and for some fields in specs, you must *qualify* the value by specifying a *unit* of measure. The available unit qualifiers are:

- q — Points
- p — Picas
- c — Ciceros
- d — Didots
- i — Inches
- m — Millimeters
- k — Kyus
- n — Microns (XPP units)
- z —Centimeters

For example, 6q means 6 points, 4p means 4 picas.

Notes:

- *You can also use pc, pt, in, mm, and cm in fields where the system allows the standard ISO unit abbreviations.*
- *For fields in specs, no more than two decimal places can be entered for Points, Inches, Didots, Centimeters, Millimeters, or Kyus units. If a more precise value is needed, use the equivalent value in Microns (XPP units).*

Chapter 1

Introduction

This chapter describes the XPP process that enables you to convert XPP documents to Adobe's Portable Document Format (PDF).

This chapter discusses:

- PDF (Portable Document Format)
- Features of PDF Support in XPP
  - For Acrobat Distiller
  - For Ghostscript

What is PDF?

Portable Document Format (PDF):

- Is the native file format of Adobe Acrobat products.
- Is a PostScript®-based document format language that retains the page layout, typography, color and graphics of the original document.
- Enables users to exchange and view electronic documents on any platform where a PDF viewer is available.

A PDF document contains one or more pages in a device- and resolution-independent format. PDF creates screen viewable pages that look like the printed document.

A PDF document may contain elements for interactive viewing, such as annotations or hypertext links. Hypertext links or hot spots are navigational aids that make it easier to read PDF documents online. A hypertext link is a selected area in a document that you click to jump to another area of the same document or to another document.

Producing a PDF Document

To produce a PDF document, a PDF converter, such as Ghostscript or Acrobat Distiller, converts PostScript disk files to PDF documents. After creating a PDF file, you can:

- Use a PDF viewer, like Acrobat Reader, to view and print the document contained in the file.
- Navigate through the document using thumbnail sketches, hypertext links, and bookmarks.

PDF Support in XPP describes how XPP supports the conversion of a document directly to a PDF document or to a PDF-enhanced PostScript disk file that a PDF converter can use to create a PDF document. Being able to create PDF files from XPP enables you to maintain a single source from which to produce both print and electronic output.

XPP PDF Support Features

XPP gives you multiple options to create PDF documents. There are three options to create PDFs directly from XPP: use Ghostscript, which is built in to XPP, or use another PDF converter, such as Acrobat Distiller which is installed and configured separately, or use the ”Direct to PDF” (purchasable) option, which bypasses the distilling process. Alternately, you can create a PDF-ready PostScript file and put it through a separate PDF converter on the same system or a different system. Each method has features and limitations, described in the following sections.

XPP, using PDF XyMacros, produces a PostScript file (or PDF file directly) with PDF markup elements, eliminating the need to manually add links or other elements with a PDF editor for many common PDF features. A PostScript file can then be converted to a PDF document as part of the XPP job, or processed separately by a PDF converter.

XPP provides:

- The ability to create a single output file from multiple XPP divisions in a job.
- The ability to create tables of contents and indexes automatically using the XPP CITI option.
- Cross-reference links from tables of contents and indexes to the appropriate pages across all divisions in the job.
- Control over final document display with PDF property sheets.
- The ability to create PDF text notes.
- PDF threading support so readers can browse story blocks that continue on non-consecutive pages or main text and pickups that occur in a multi-column format in the PDF document.

Related Information for Acrobat Distiller

- Refer to Adobe documents for information about Distiller and Reader.
- Refer to the Adobe home page on the World Wide Web (http://www.adobe.com) for additional information.

Ghostscript PDF Features

XPP lets you create PDF documents directly from XPP using Ghostscript. This lets you create PDFs without having to put PostScript files through a separately purchased PDF creation tool, such as Adobe Distiller.

Ghostscript supports the following PDF features:

- PDF through version 1.5 (Acrobat 6) of the PDF specification.
- A subset of Adobe Distiller 6 parameter operators.
  Refer to Appendix B for details.
- A subset of pdfmarks.
- Optimized PDF

These features make creating PDFs with Ghostscript an appropriate choice for general-purpose PDFs.

Optimized PDF

Optimized PDF allows Web sites to display PDF in a browser more quickly. When using Ghostscript to distill to PDF, the output driver takes **–pdfoptgs** argument, enabling the PostScript driver to create an optimized PDF file.

Note: *This option only applies when using Ghostscript to distill to PDF. If you wish to use Adobe Acrobat Distiller to create optimized PDF, you must configure this option outside of XPP in the Acrobat Distiller job options.*

Related Information

Refer to Appendix B for the list of options that are supported by Ghostscript when creating PDF documents directly from XPP.

Chapter 2

Creation of a PDF Document

This chapter describes how to create a PDF document, including:

- Software requirements
- Converting an XPP job to PDF
- Choosing the PDF converter program and options
- Understanding naming conventions of PDF-enhanced PostScript files and PDF documents
- Creating PDF tables of contents and indexes without using CITI

Software Requirements

If you want to create PDFs using a Distiller program, XPP can run on the same system as the Distiller or on a separate system. To execute the Distiller automatically from XPP, install Distiller on the same system running XPP or mount the Distiller directory remotely.

If you cannot access the Distiller application directly from XPP, you can copy the PDF-enhanced PostScript file created by XPP to the system running Distiller for conversion to a PDF document.

You can also create PDFs from XPP using Ghostscript, which distills Postscript files into PDFs, or using XPP's ″Direct to PDF″ (purchasable) option, which bypasses the distilling process.

Creating PDF Documents

This section lists steps to follow when creating a PDF document and refers you to corresponding chapters.

Tip! Refer to Chapter 13, ”Printing” in the *XPP User Guide* for information on PDF print options.

Table 2-1 *Steps to Create PDF Documents*

Step	*Task*	*Refer to*
1	Open divisions in an XPP job and insert PDF start and end XyMacros. If you use CITI to generate tables of contents and indexes with links to the appropriate pages, CITI inserts the PDF XyMacros automatically.	Chapter 3
2	Edit the Job Ticket and enter the name of the PDF Support Spec and the CI Spec.	Chapter 5
3	Edit the PDF Support Spec at the job level and specify attributes for the PDF file that will be created.	Chapter 4
4	Edit the Document Assembly Ticket. • Resolve cross-references across divisions in the job by setting the *Resolve Xrefs across Job?* field to **yes** and *Compose* to **yes**. • Enter the names of divisions you want CITI to generate (*pdftoc*, *pdfidx*, and/or *citi*), and the names of *main* divisions to include in the PDF document. Set the PDF field to **yes** and the CITI field to **yes**. • To generate a PDF-ready PostScript file without converting it to a PDF file, set the *Execute Distiller?* field to **no**. • To generate a PDF-ready file, set the *Execute Distiller?* field to **yes**.	Chapter 5
5	Edit the CI Spec and enter the names of main divisions containing marked entries to extract for tables of contents and indexes, and divisions (*pdftoc*, *pdfidx*, and/or *citi*) to generate. • Set the *PDF Level (1-9)* for elements that will have PDF markup generated. • Enter optional data in the *Optional Data* fields for PDF Levels 1–9.	Chapter 5

Table 2-1 *Steps to Create PDF Documents (Continued)*

Step	*Task*	*Refer to*
6	Pick a PDF converter option: To use Adobe Distiller—create or edit the *_distill_parms* ASCII file at the job level for additional switches to pass to the Acrobat Distiller during job processing. Alternatively, from PathFinder, select **File > Print**, select *PS to PDF file* on the *Print* tab, click the *PS to PDF* tab, and then select *Use Adobe Acrobat Distiller*. To use Ghostscript—create or edit the `_gsdistill_parms` file at the job level or the `gsdistill` file at the system level for switches to pass during job processing. Alternatively, from PathFinder, select **File > Print**, select *PS to PDF file* on the *Print* tab, click the *PS to PDF* tab, and then select *Use Ghostscript (gs)*. To use "Direct to PDF" (purchasable option), from PathFinder select **File > Print** and then select *PDF file* on the *Print* tab. Note: XPP's **Direct to PDF** option does not require a PDF conversion from PostScript.	Chapter 4
7	Select *Compose* from PathFinder to compose active divisions listed in the Document Assembly Ticket and resolve cross-references. (Right-click the job; XPP displays a pop-up menu listing Process. Select Process; XPP displays a menu of processes; Select Compose.)	Chapter 6
8	Select *CITI* from the Process pop-up menu to generate *pdftoc*, *pdfidx*, and/or *citi* divisions.	Chapter 6
9	Select *PDF Check* from the Process pop-up menu to run the *showpdf* program that verifies PDF markup in the job. The *showpdf* program informs you whether the format of PDF XyMacros is correct and if there are missing start or end XyMacros.	Chapter 6
10	From PathFinder, select **File > Print** (or right-click on the job you want to print and select *Print* from the pop-up menu). XPP displays the Print dialog box. Select the *PS to PDF file* radio button on the *Print* tab, and then select distill options on the *PS/PDF* tab, or select the *PDF file* radio button to bypass the distill process (requires purchase of the "Direct to PDF" option).	Chapter 6
11	After the PDF document is created, you can use a PDF viewer, such as Acrobat Reader, to view, print, and navigate through the document.	PDF viewer documentation

Choosing the PDF Converter Program and Options

Creating Native PDFs

XPP’s ″Print Direct to PDF″ is an output option that bypasses the two-step *produce-PostScript-then-produce-PDF* paradigm. There is no distiller requirement. XPP has retained the options to print an XPP job or division directly to a printer, write to a PostScript file, and to distill (Acrobat Distiller or Ghostscript) to a PDF. XPP’s Print Direct to PDF option is considerably faster than the two-step format/distill process; it also allows access to new, PDF-specific features not available in PostScript.

Note: The ″Print Direct to PDF″ option is a licensed, add-on feature.

Configuring Print Direct to PDF

Before you can use the ″Print Direct to PDF″ option from XPP, you must convert the encoding files for PostScript Type 1 fonts, and also ensure that they have Adobe FontMetrics (`.afm`) font descriptions.

Note : This is a separately licensed feature. See the “Post-Installation Configuration” chapter in the appropriate publication for instructions on Configuring Print Direct to PDF.

For more information on converting font encoding files, see the appropriate publication:

- *XPP Installation and Upgrade Guide for Windows*
- *XPP Installation and Upgrade Guide for Linux*

Running Print Direct to PDF

You access the ″Print Direct to PDF″ feature using the *PDF file* option on the Print tab (**File > Print**) in PathFinder, or by right-clicking on a job. You can also invoke the `divpdf` utility from the command line.

When using the ″Print Direct to PDF″ feature, note the following:

- PDF output honors the optimized for web option.
- From the command line, enter `xyhelp divpdf` for a list of options. See the *XML Professional Publisher: Command-Line Utilities* publication for more information.
- From PathFinder, fonts are not automatically embedded unless the Enable font download option is enabled. However, from the command line, divpdf will include fonts as a subset. Use the `-noembed` and `-nosubset` options to disable this behavior.
- For job processing, the default output type is ″pdf.″ You can override this with the `-da` option.

- The default SEP spec name is “pdf.” You can override this with the `-sep` option.
- To produce ″pdf″ directly through auto-processing, create a background queue entry and set the Device Name to “pdf” (for the command line, use `divpdf`). The Modify options will open a Device Option File (DOF) spec with options, which you can enable.

See Chapter 13, ″Printing″ in the *XML Professional Publisher: User Guide* for information on ″Print Direct to PDF.″

PDF Pass-Through Macro Support

The following PDF pass-through macros *are* supported in PS/PDF and ″Direct to PDF″ output:

- 2DEST
- 2PAGE
- BOOK
- DEST
- DOCINFO
- INDEX
- LAUNCH
- NOTE
- TOC
- WWW
- XREF

The following PDF pass-through macros *are not* supported in ″Direct to PDF″ output:

- GEN
- RECT
- ARTICLE
- 2ART
- PS

Using the PostScript to PDF Converter

When you install XPP, the PostScript formatter is configured to create PDF output using the built-in default command for Acrobat Distiller (*acrodist* on Windows, *distill* on UNIX). To use this, the program must exist in your executable search path. You may override that by providing a custom distiller command and options through various techniques.

How to Change the PDF Converter Defaults

There are several ways you can override the built-in distiller command or provide command line options:

- Define the environment variable XYV_DISTILL to provide a new default command to use Acrobat Distiller to create PDF files.
- Create the text file $XYV_EXECS/sys/config/gsdistill to select Ghostscript as the default distiller and to provide command options that apply to all PDF jobs.
- Create the text file _distill_parms in the Job directory to provide command options that apply to a specific job or division when Acrobat Distiller is used.
- Create the text file _gsdistill_parms in the Job directory to provide command options that apply to a specific job when Ghostscript is the distiller program.

You can also override the configured default program by selecting the distiller program to use for the current run of the formatter with a command line option (or a choice in the PathFinder Print dialog, PS/PDF tab)

To use Acrobat Distiller:

- Use the **–pdfuseacro** command line option.
- Select the *Use Adobe Acrobat Distiller* option in PathFinder.

To use Ghostscript:

- Use the **–pdfusegs** command line option
- Select the *Use Ghostscript (gs)* option in PathFinder.

Refer to “How XPP Determines the Distiller Program and Options to Use” on page 2-11 to see the order in which XPP processes these options.

Specifying Default Acrobat Distiller Command and Options with XYV_DISTILL

XYV_DISTILL is an environment variable you use to specify the exact command to launch Acrobat Distiller and options that apply to all PDF jobs. For instance, on a Windows system, you might define XYV_DISTILL to be:

```
C:\PROGRA~1\Adobe\ACROBA~1.0\DISTIL~1\acrodist  /N  /Q
```

This example shows the *acrodist* command at a specified path, with the new instance flag (/N) and the quit flag (/Q), which makes Distiller run independently of other instances of Distiller and ignore watched folders, and then terminate correctly when it completes creation of the PDF file.

If on Windows the path name contains any spaces you need to use the DOS short directory names, as shown in the example above. The other option is to use double quotes in order to use the long path name that contains spaces, for example

″C:\Program Files\Adobe\Acrobat 6.0\Distiller\acrodist.exe″ /N /Q

If you run Acrobat Distiller in the background, you must also disable the interactive features in the Preferences—Acrobat Distiller dialog box, such as “Ask for PDF File Destination.″ Otherwise, the file hangs in the queue.

You can use XYV_DISTILL on Windows, UNIX, and Linux.

The _distill_parms and _gsdistill_parms Files

You can specify PDF generation parameters in the ASCII *_distill_parms* or *_gsdistill_parms* files stored at the job level. During job processing or when printing a single division, XPP passes parameters in the *_distill_parms* file or *$XYV_EXECS/sys/config/gsdistill* and *_gsdistill_parms* files to the PDF converter.

Note: *Parameters set in these files override any settings for those parameters which may be set in the PDF Support Spec.*

To create and edit the *_distill_parms* or *_gsdistill_parms* file:

1. Go to the command line prompt.
2. Change to the job level directory.
3. Use an ASCII editor, such as *vi* to create a file named *_distill_parms* or *_gsdistill_parms*.
4. Insert standard Distiller or Ghostscript commands in the file, then store and exit.

Example Here is a sample *_distill_parms* file at the job level for Distiller V3:

```
-v -embedall -subsetfonts on
```

Note: *If you set the* **efd** *(Enable Font Download) switch in the PDF queue with Acrobat Distiller V3, you must add the* ***–embedall*** *command to the _distill_parms file in the job directory.*

Refer to Chapter 5 for information on setting device options.

Warning

Parameter switches are not fully compatible between all versions of Adobe Distiller or with Ghostscript.

If you are running Distiller from the operating system command line, be certain that the switches you use are compatible with the version of the Distiller that you are running. Thumbnails, for example, are supported in Distiller V2.1 but not in V3.0. Using the switches for thumbnails from V2.0 will cause V3.0 Distiller to fail.

If you want to create PDFs from XPP directly using Ghostscript, refer to Appendix B for a detailed list of which Distiller options are supported.

Configuring XPP to Use Ghostscript

To select Ghostscript as the default program to create PDFs, you may create the file `$XYV_EXECS/sys/config/gsdistill`. This directs XPP to use Ghostscript to convert to PDF rather than Adobe Distiller.

When XPP uses Ghostscript to create PDF, the formatter creates the following initial gs command:

```
$XYV_EXECS/gs/gs -q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite
-sOutputFile=input.pdf -c currentuserparams /VMThreshold get
3000000 .max setvmthreshold -f input.ps
```

where *input.ps* is a PostScript file created by the formatter and *input.pdf* is the resulting PDF file with the same base name, but a .pdf file extension.

Providing Arguments to Ghostscript

If the `gsdistill` file is not empty, its contents will be added to the Ghostscript command line. For instance, Ghostscript creates PDF version 1.3 (which corresponds to Acrobat version 4.0) by default. You could change this by adding the following switch to the file:

```
-dCompatibilityLevel=1.4.
```

Values in the gsdistill file apply to all jobs that use Ghostscript as distiller.

To provide job-specific options when using Ghostscript as distiller, create the text file _gsdistill_parms in the Job directory. Its contents will be added to the command line. Refer to Appendix B for options Ghostscript supports.

Providing Arguments to Acrobat Distiller

To provide job-specific options when using Acrobat Distiller, create the text file _distill_parms in the Job directory. Its contents will be added to the command line.

For example, the following set of command arguments might be put in a _distill_parms file for use with Acrobat distiller 3.0:

```
-v -embedall -subsetfonts on
```

***Note:** If you set the **−efd** (Enable Font Download) option in the XPP job to be distilled with Distiller 3.0, you must add the **−embedall** command to the _distill_parms file in the job directory.*

Parameters set in option files override any settings for those parameters that may be set in the PDF Support Spec.

Warning

Parameter switches are not fully compatible between all versions of Acrobat Distiller or with Ghostscript.
For example, Thumbnails are supported in Distiller 2.1, but not in Distiller 3.0; using the switch for thumbnails from Distiller 2.0 causes Distiller 3.0 to fail.

Refer to Appendix B for distilling options available in Ghostscript.

How XPP Determines the Distiller Program and Options to Use

The formatter program, *psfmtdrv*, uses the following logic to determine which distiller program to use:

1. If the **−pdfusegs** option is present, use Ghostscript;
2. else if XYV_DISTILL is defined, use it to run Acrobat Distiller;
3. else if the **−pdfuseacro** option is present, use Acrobat Distiller;
4. else if $XYV_EXECS/sys/config/gsdistill exists, use Ghostscript;
5. else use Acrobat Distiller.

The formatter then determines which command options to use in this way:

1. If using Acrobat Distiller and XYV_DISTILL is defined, use its value as the Acrobat Distiller command and options that apply to all PDF jobs; if _distill_parms exists at the job level, add its contents to the command line.
2. else if using Acrobat Distiller, use the built-in default command; if _distill_parms exists at the job level, add its contents to the command line.
3. else if using Ghostscript, use $XYV_EXECS/gs/gs as the command; if $XYV_EXECS/sys/config/gsdistill exists, add its contents to the command line; if _gsdistill_parms exists at the job level, add its contents to the command line.

Understanding Names for PDF-ready Disk Files

When you select *PS to file* on the Print tab in the Print dialog box, XPP creates a PDF-ready PostScript disk file.

XPP uses the following default name format:

jobname [_divname] [_pdfspecname] [_setpagenum].ps

By default:

- *jobname* is always listed.
- *_divname* is included if you are printing from the division level.
- *_pdfspec* is included if you are using **–pdfmark**.
- *_setpagenum* is included if you are using **–setsize.**

These default naming conventions apply unless you provide a path/file name on the Output File tab or use **–pn** to supply an explicit output file name.

If you select *PS to PDF file* on the Print tab and *Use PDF markup* from the PS/PDF tab on the Print dialog box, the PDF conversion tool converts the PostScript file to a PDF file. The default name of the resulting file is the same as the name of the PostScript file, but with the extension .pdf instead of .ps.

Creating PDF TOCs and Indexes Without CITI

CITI is an XPP option that can generate tables of contents and indexes automatically without having to insert additional PDF markup in text.

If you do not have the CITI option or do not want CITI to generate tables of contents and indexes, you can do the following:

1. Edit XPP divisions and insert PDF XyMacro pairs around text to mark for tables of contents and/or indexes.
2. Enter the name of the PDF Spec in the Job Ticket.
3. Edit the PDF Spec.
4. Enter the names of active divisions for PDF processing in the Document Assembly Ticket.
5. Select Print from the job’s pop-up menu and specify PDF-related options.
   - If the *Execute Distiller?* field in the Document Assembly Ticket is set to **no**, XPP creates a PDF-enhanced PostScript file consisting of active divisions.
   - If the *Execute Distiller?* field in the Document Assembly Ticket is set to **yes**, XPP creates a PostScript file consisting of active divisions and then runs the PDF converter on the file to create a PDF document.

Chapter 3

PDF XyMacros

This chapter describes PDF XyMacros supported by XPP Version 8.1 and later, including:

- The function of PDF XyMacros
- PDF page numbers
- The format of PDF XyMacros
- Guidelines for using PDF XyMacros
- Descriptions and examples of PDF XyMacros
- The relationship of PDF XyMacros to Adobe *pdfmarks*

Function of PDF XyMacros

PDF XyMacros define PDF features in the PostScript file, such as the start and end of text strings that are hot spots or hypertext links in a PDF document.

The PDF XyMacro set differs from XPP standard XyMacros because composition does not process the contents of PDF XyMacros. Instead of processing PDF XyMacros, XPP composition passes the PDF XyMacros to the XPP PostScript formatter program (*psfmtdrv*) that converts their contents into corresponding *pdfmarks* in the PostScript language file.

A *pdfmark* is an operator in the PostScript language file that represents annotation, link, bookmark, document, and page-cropping information for use by the PDF converter. For information on *pdfmarks*, refer to Adobe documentation.

Guidelines for PDF XyMacros

Guidelines for PDF XyMacros are:

- XyMacros must be entered in start and end pairs.
- PDF XyMacros may contain required key-values and optional-data fields. You must enter required values, such as names and note text. You can also enter user-defined optional data in the XyMacro, such as *RGB* color values.
- The following PDF XyMacro types – which do not create a hotspot link and do not capture text from the PDF start to the PDF end XyMacro – can be nested with all other PDF XyMacro types: ARTICLE, NOTE, PS, GEN, and DOCINFO.

  The following PDF XyMacro types cannot be nested with each other: TOC, BOOK, INDEX, XREF, DEST, 2PAGE, 2DEST, 2ART, LAUNCH, WWW, and RECT.
- A colon (:) is the first character in a PDF XyMacro, followed by either *pdfs* (pdf start) or *pdfe* (pdf end).
- You can enter PDF XyMacros, such as XREF or 2DEST, within tag and macro definitions in the expansion fields of XyMacros, or use them in the *Prestring* or *Poststring* fields of an Item Format Spec. These XyMacros pass through to the PostScript output formatter as computer generated text, but are honored as if they were present in the main text.
- If you use the CITI option to generate PDF tables of contents and/or indexes you do not have to insert PDF TOC and INDEX XyMacros in your document. CITI inserts the XyMacros automatically.

- You can insert PDF XyMacros manually within XPP divisions, regardless of whether you use CITI to generate PDF tables of contents and indexes.
- If you are creating an XyASCII file with PDF XyMacros that will be imported into an XPP job, be aware of XPP naming conventions when entering the PDF XyMacros, such as which characters are valid, or the maximum length of story and pickup names.

PDF Page Numbers

PDF has no knowledge of the number that appears on a printed page. The first page in a PDF document is numbered page 1. To number other pages, the pdfmark operator uses the sequence number of a page, relative to the start of the document.

When XPP processes a job, it combines all active divisions into one PostScript file and numbers the pages sequentially, starting with page 1.

The page number in a PDF XyMacro may differ from the number that appears on the printed page. For example, this document uses multi-part page numbers, and the first page in chapter 2 is numbered *2-1*. If page 2-1 were the 15th page from the start of the job, the PDF sequence page number would be *15*.

When the CITI option processes main divisions, it calculates page numbers relative to the start of the division. When CITI creates *pdftoc* and *pdfidx* divisions, it inserts the relative page number preceded by a *plus sign* (+) and the division name in the PDF start XyMacro. The plus sign (+) indicates a page number relative to the start of the division. When the PostScript formatter program (*psfmtdrv*) processes a job, it knows that:

- Page numbers with a preceding plus sign (for example, +*2*) are relative to a specific division, which *psfmtdrv* converts to sequence numbers for the whole job (that is, relative to the PostScript file).
- Page numbers without a preceding plus sign (for example, *2*) are sequence numbers relative to the whole job (that is, within the PostScript file).

For example, these XyMacros **⟪:pdfs;INDEX;+1,02chap⟫***text***⟪:pdfe;INDEX⟫** indicate that an index item appears on the first page of the division named *02chap*.

Sequence Page Numbers for the Job

To specify the page sequence number within the whole job, that is, the page position within the PostScript file, omit the plus sign (and division name).

For example, to mark an item that appears on the tenth page of the PDF document, enter the sequence page number without the plus sign:

⟪:pdfs;INDEX;10⟫*text***⟪:pdfe;INDEX⟫**

If the PDF XyMacro contains both the sequence number and the division name, XPP uses the sequence number and ignores the division name:

⟪:pdfs;INDEX;10,08chap⟫*text***⟪:pdfe;INDEX⟫**

Relative Page Numbers for the Division

To specify the page number relative to the start of the named division, insert a plus sign (+) before the page number. When XPP creates a PostScript file, it converts the page numbers relative to the start of the division into page sequence numbers for the job.

For example, to mark an entry that appears on the first page of the division named *02chap*, enter the relative page number (with a preceding plus sign) and the division name.

《:pdfs;INDEX;+1,02chap》*text***《:pdfe;INDEX》**

Use Page Sequence Numbers Carefully

Be careful when entering sequence page numbers for the whole job. If changes occur in the XPP divisions that comprise the PostScript file, entries could appear on different pages and your sequence page numbers will be incorrect.

Page Numbering Example

For example, if you are not using CITI to generate an external PDF Table of Contents, you can enter PDF Table of Contents XyMacros in XPP divisions. You mark text you want to include in the external Table of Contents by inserting TOC XyMacro pairs around each entry.

You mark PDF Table of Contents entries in four XPP divisions. Chapter 1 has 8 pages, Chapter 2 has 16 pages, Chapter 3 has 6 pages, and Chapter 4 has 18 pages.

If you add a new section to Chapter 1, Chapter 1 increases to 10 pages when you recompose the division named C. So the sequence numbers for all entries in following divisions (Chapters 2 through 4) will be incorrect in the PostScript file, and ultimately, in the PDF document.

Format of PDF XyMacros

The format of PDF start and end XyMacros is:

《:pdfs;*type***;***required-value(s)***;***opt-data***》**

《:pdfe;*type***》**

Table 3-1 *Format of PDF Macros*

Entry	*Description*
《 》	Open and close angle brackets enclose each macro.
:pdfs	A start PDF XyMacro appears before any document that may be included in the PDF element, such as a hot spot or link. The beginning colon (:) tells XPP composition not to process the macro contents.
type	A mnemonic or number representing the PDF XyMacro type. You can use the mnemonic and number interchangeably. Table 3-2 on page 3-7 lists pdfmarks by type.
;	A semi-colon separates arguments in a PDF XyMacro.
required-value(s)	Sometimes referred to as key-values for the pdfmark. A comma separates multiple values.
opt-data	Optional data that is passed to the pdfmark for use by Acrobat, such as RGB color values. Follow standard PDF conventions when defining optional data.
:pdfe	An end PDF XyMacro appears after any document that may be included in the PDF document, such as a hot spot or link. The end XyMacro must be the same type as the start XyMacro.

For example, you can enter this sample NOTE (type 7) XyMacro in an XPP division in one of two ways:

《:pdfs;NOTE;This is sample note text.》《:pdfe;NOTE》

or

《:pdfs:7;This is sample note text.》《:pdfe;7》

XPP converts the PDF XyMacro to a pdfmark that it passes to the PDF-ready PostScript file in this format:

```
[ /Rect [ x1 y1 x2 y2]
/Contents (This is sample note text.)
/ANN pdfmark
```

PDF XyMacro Types

Table 3-2 lists the number and mnemonic of each PDF XyMacro, the corresponding *pdfmark*, and required values.

Table 3-2 *PDF Macros by Type*

PDF Macro Type		**pdfmark operator**	**Required Values**
ID No.	**Mnemonic (Description)**		
1	TOC (Table of Contents)	/OUT	Relative page (number), subordinate count (number) division name*
2	BOOK (BookMark)	/OUT	Sequence page (number), subordinate count (number)
3	INDEX (Index)	/ANN	Relative page (number), division name*
4	XREF (Cross-reference)	/ANN	Reference name from ⟪**rx**⟫ (Reference Marked Spot) XyMacro, or the name of an XPP footnote or pickup (string)
5	ARTICLE	/ARTICLE	Article name (string) Not supported with *Print Direct to PDF*.
6	DEST (Named destination)	/DEST	Destination name (string)
7	NOTE (Note)	/ANN	Note text (string)
8	2PAGE (Go to Page)	/ANN	Sequence page (number)
9	2DEST (Go to Dest)	/ANN	Destination name to go to (string)
10	2ART (Go to Article)	/ANN	Article name to go to (string) Not supported with *Print Direct to PDF*.
11	LAUNCH (Launch)	/ANN	File name (string)
12	WWW (Link to WWW)	/ANN	URL (string)
13	PS (Pass-through PostScript)	/PS	PostScript language code (string) Not supported with *Print Direct to PDF*.
14	GEN (Generic)	“generic”	String Not supported with *Print Direct to PDF*.

Table 3-2 *PDF Macros by Type (Continued)*

PDF Macro Type		pdfmark operator	Required Values
15	DOCINFO	/DOCINFO	Key-value pairs
16	RECT (Generic with hotspot Rectangle)	“generic”	String Not supported with *Print Direct to PDF*.

* Refer to the descriptions of TOC and INDEX XyMacros for information about required values for these XyMacros.

Descriptions of PDF XyMacros

This section describes PDF XyMacro pairs supported by XPP that you can insert in XPP divisions. Each macro can be code, using either a Keyword, such as BOOK, TOC, etc., or can be a numeric ID, such as 1, 2, 3, etc.

TOC — Table of Contents

The Table of Contents XyMacros mark text included in a *pdftoc* table of contents division and also to be included in the external PDF table of contents. Table of Contents XyMacros generate the same *pdfmark* as BOOK XyMacros for creating a PDF outline entry. In addition, each table of contents entry becomes a hot spot, with a link to the appropriate page in the PDF document.

- The PDF table of contents (*pdftoc*) division can be prepended to the main PDF document for viewing in the PDF viewer window.
- The external PDF table of contents appears in the Bookmarks window, on the left side of the PDF viewer window, when the PageMode is set to *UseOutlines*.

Note: *There are exceptions when generating a table of contents division where you might want to generate a* pdfidx *type division instead of a* pdftoc *type division (such as when you do not want marked text to also become PDF Bookmarks).*

The numeric ID for TOC is 1.

Format The format of the PDF table of contents XyMacro pairs is:

«:pdfs;TOC;+*rel-page#***,***count***,***div-name***;***opt-data***»**TOC-text**«:pdfe;TOC»**

—or—

«:pdfs;1;+*rel-page#***,***count***,***div-name***;***opt-data***»**TOC-text**«:pdfe;1»**

Note: *You can also enter an absolute page sequence number, in which case, you do not enter the division name. For example,*

«:pdfs;TOC;abs-page#,count;opt-data»

Specifying Alternate PDF Bookmark Text

You can specify alternate title text to appear in the PDF Bookmarks window instead of the TOC-text that appears in the main document. Using /Title ( ) in the optional data field instructs XPP to use that string instead of generating a title.

For example:

«:pdfs;TOC;+*1,-0,chap2;/Title (The Title I Want)***»«:pdfe;TOC»**

Comments When the *Link to target position* field in the PDF spec is set to ’yes’ and TOC is being used, with content between the **《:pdfs》** and **《:pdfe》**, XPP will generate a pdfmark that includes a position attribute of the form /View [XYZ x y 0], where *x* is slightly left of the start of the target content, *y* is slightly above the target content, and *0* is a *zoom* factor that means no change to the current zoom factor and page display mode.

Note: *Whether or not the page actually scrolls to that point in the viewer window depends on limitations in the viewer behavior.*

Without Using CITI

If you do not use CITI to generate a PDF table of contents division, you must enter PDF table of contents XyMacros in XPP divisions. Mark text to include in the external PDF table of contents (as well as make into a hot spot) by inserting the TOC XyMacro pairs around each entry.

Enter the following arguments:

- The page number relative to the start of the division, preceded by a plus sign (+).

  Entering the page number without the preceding plus sign specifies the sequence page number within the job (instead of the relative page number within the division).
- The count (the number of children or subordinate levels immediately following the entry).

  You can affect the display of the table of contents in the PDF viewer by displaying the count with or without a preceding minus sign (**-**). No preceding minus sign causes the TOC to display in expanded format. Preceding the count with a minus sign causes the TOC to display ”collapsed”, that is, subordinate levels following the entry do not display automatically; the user opens them interactively in the PDF viewer.
- The name of the division (only if using the relative page number (+). If you use the +relative-page# variation of TOC and do not specify the division name, the default is a relative page number in the current division.
- Optional data, such as color values.

Using CITI with Standard XPP

If you use CITI to generate a table of contents *citi* division, you can use the same CITI tags and codes to generate a *pdftoc* division that includes hot spot links and that also generates an external PDF table of contents. You do not have to enter PDF table of contents XyMacros in text. When CITI processes the job, it inserts the PDF XyMacro pairs around the extracted TOC entries after the *rsv* Register Save XyMacro, and supplies the relative page number, subordinate count number, and division name. For example:

《rsv;...》《:pdfs;...》{INSERT}^...Extracted Text...%**《:pdfe;...》**

Using CITI with XML/SGML

If you are processing an XML or SGML document, then the *rsv* and PDF XyMacro pairs are output as XPP processing instructions. The format for one complete PDF level entry within a *citi* record is:

XML format:

《?xpp rsv;...?》《?xpp :pdfs;...?》{INSERT}^...Extracted Text ...%
《?xpp :pdfe; ...?》

SGML format:

《?xpp rsv;...》《?xpp :pdfs;...》{INSERT}^...Extracted Text...%**《?xpp :pdfe;...》**

Formatting Order of Importance

Note the importance of the order of the data within any one level:

1. The RSV XyMacro precedes the PDF start so it is not within the PDF start/end string.
2. The INSERT string (if any) follows the PDF start so any data it generates is included within the PDF start/end string.
3. The PDF start and INSERT string precede the ^ (hex lf) field delimiter so they are not part of the text field, and therefore, do not get sorted, deduped, etc.
4. The PDF end follows the generated pgraf mark so it is within the PDF start/end string.

The Process of Generating a PDF Table of Contents Division

To use CITI to generate a PDF table of contents division:

- Edit the Document Assembly Ticket for the job and enter **pdftoc** in the Division Type field of the table of contents division to generate. In some cases you may want to use the **pdfidx** type for a table of contents division (such as when you do not want marked text to also become PDF Bookmarks).
- Edit the CI Spec for the job and fill in the *PDF Level* fields and *PDF Optional Data* fields.
- Run *CITI* from the Job Processing menu. (CITI is located on the sub-menu of the pop-up menu that XPP displays when you right-click the job.)

Refer to *PDF Collapse* in the CI Spec to collapse fields in the TOC.

Note: *Restrictions in Adobe Acrobat Distiller software limit the length of Bookmark-text to 255 PDFDocEncoding (8-bit) characters or 126 Unicode (16-bit) values. Because both the BOOK and TOC XyMacros generate the same pdfmark for a PDF Bookmark, the same limit applies to TOC-text. If TOC-text or Bookmark-text exceed these limits, XPP truncates the text at that point to avoid Distiller errors. The Adobe pdfmark Reference Manual recommends "a practical limit of 32 characters...so that it can be read easily in the Acrobat viewer."*

Refer To Refer to Chapter 5 "XPP Specs and Processes" to generate tables of contents.

The following example shows entries in a CITI-generated PDF TOC .

```
《rsv;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;1;3;1;1;1;0;0》
《:pdfs;TOC;+1,4,01chap;/Color [0 0 1]》⟦cnchap⟧Chapter 1◘Getting
Started◘《:pdfe;TOC》
《rsv;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;2;2;3;1;1;2;0;0》
《:pdfs;TOC;+2,1,01chap;/Color [0 1 0]》⟦cnhead1⟧Overview◘
1-2《:pdfe;TOC》
```

- CITI inserts the *:pdfs;TOC* XyMacros for Chapter 1 and Overview after the corresponding *rsv* XyMacros.
- The **+** in front of a page number (+2) tells *psfmtdrv* the relative page number of the *Overview* within the division (01chap). The *psfmdrvt* program uses the relative page number and division name to calculate the sequence page number of an item within a job.

  For example, a job contains divisions with these total pages — cover has 1 page, notice has 1 page, table of contents has 4 pages, 01chap has 12 pages, 02chap has 6 pages, and 03chap has 8 pages.

  The fourth page of 03chap has a relative division page number of +4 and a job sequence page number of 28.
- The TOC XyMacro specifies an outline entry that produces an /OUT pdfmark operator. The count specifies the number and visual appearance of subordinate entries in the outline (for example, a chapter with several section headings).

  In this example, Chapter 1 (parent) has four subordinate levels (children), while Overview has only one subordinate level. When the Page Mode is set to *Outlines*, clicking open a parent displays the subordinate entries.
- Both the Chapter 1 and Overview entries contain optional RGB color data. /Color [0 0 1] produces a blue border around text and /Color [0 1 0] produces a green border around text.
- When *psfmtdrv* processes the PDF XyMacros, each PDF TOC XyMacro generates two pdfmark commands—one for the PDF-outline (external PDF table of contents) and one to create the "hot spot" for the entry itself.

- The TOC XyMacros generate the same *pdfmark* operator as Bookmark XyMacros (for a PDF Bookmark). CITI inserts TOC XyMacros when it generates a **pdftoc** table of contents division. If you do not use CITI to generate a PDF table of contents division, you can insert either TOC or BOOK XyMacros in text.
- If you insert the TOC XyMacro manually in text, you can specify the sequence page number relative to the whole job by omitting the preceding plus sign and the division name. For example, **《:pdfs;TOC;24》** identifies an item on the 24th page of the PostScript file.

BOOK Bookmark

Bookmark XyMacros mark pages or specific page views. The BOOK XyMacro generates the same *pdfmark* operator as TOC for creating a PDF outline entry (but does not generate a hot spot like TOC does).

You can insert bookmark XyMacros within the text of an XPP division using one of two methods: entering the page sequence number or entering an asterisk (*) to indicate the current page.

The numeric ID for BOOK is 2.

Note: Restrictions in Adobe Acrobat Distiller software limit the length of Bookmark-text to 255 PDFDocEncoding (8-bit) characters or 126 Unicode (16-bit) values. Because both the BOOK and TOC XyMacros generate the same pdfmark for a PDF Bookmark, the same limit applies to TOC-text. If TOC-text or Bookmark-text exceed these limits, XPP truncates the text at that point to avoid Distiller errors. The Adobe pdfmark Reference Manual recommends "a practical limit of 32 characters...so that it can be read easily in the Acrobat viewer."

Format

The format of the start and end bookmark XyMacros is:

《:pdfs;BOOK;*seq-page#***,***count***;***opt-data***》**Bookmark-text**《:pdfe;BOOK》**

Specifying Alternate Bookmark Text

You can specify alternate title text to appear in the PDF Bookmarks window instead of the BOOK text that appears in the main document. Using /Title ( ) in the optional data field instructs XPP to use that string instead of generating a title.

Comments

When the *Link to target position* field in the PDF spec is set to 'yes' and BOOK is being used, with content between the **《:pdfs》** and **《:pdfe》**, XPP will generate a pdfmark that includes a position attribute of the form /View [XYZ x y 0], where *x* is slightly left of the start of the target content, *y* is slightly above the target content, and *0* is a *zoom* factor that means no change to the current zoom factor and page display mode.

Note: *Whether or not the page actually scrolls to that point in the viewer window depends on limitations in the viewer behavior.*

Format The format for alternate bookmark text is:

⟪:pdfs;BOOK;**,-0;/Title (The Title I Want)***⟫⟪:pdfe;BOOK⟫**

Using Page Sequence Number

Enter the page sequence number relative to the job (without a preceding plus sign) and the count (number of subordinate entries or children).

When specifying a /Title (...) string in the pdfmark macro that itself contains parentheses, the parentheses should be escaped with a backslash.

⟪:pdfs;BOOK;**,-0;/Title (The \(Other\) Title)***⟫⟪:pdfe;BOOK⟫**

Refer To Refer to the TOC XyMacro for additional information and examples of specifying a title string.

Using an Asterisk

Insert bookmark XyMacros within the text of an XPP division. Enter an asterisk (*) to indicate the current page; then enter the count (number of subordinate entries or children). The program inserts the appropriate page number when it formats the document.

Format The format of the start and end bookmark XyMacros is:

⟪:pdfs;BOOK;*,*count***;***opt-data***⟫**Bookmark—text**⟪:pdfe;BOOK⟫**

Refer To Refer to the TOC XyMacro for additional information and examples of specifying the current page and the count.

INDEX Index

The index XyMacros mark text to include in a PDF index that is appended to the main document (or in some cases in a PDF table of contents that is prepended to the main document). Index entries are hot spots, with links to the appropriate pages in the PDF document.

The numeric ID for Index is 3.

Format The format of the PDF index XyMacros is:

⟪:pdfs;INDEX;+*rel-page#***,***div-name***;***opt-data***⟫**Index-text**⟪:pdfe;INDEX⟫**

Comments If you do not use CITI to generate a PDF index, you must enter PDF index XyMacros in the XPP division. Mark text to include in a PDF index by inserting the INDEX XyMacro pairs around each index entry.

If you use CITI to generate an index *citi* division, you can use the same CITI tags and codes to generate a *pdfidx* division that includes hot spot links, but does not generate an external PDF table of contents. You do not have to enter additional PDF XyMacros in text unless you want to mark entries that are not recognized by CITI.

When CITI processes the job, it inserts the PDF XyMacro pairs around the extracted index (or TOC) entries after the *rsv* Register Save XyMacro, and supplies the relative page number and division name.

To use CITI to generate a PDF index (or TOC) division:

- Edit the Document Assembly Ticket for the job and enter **pdfidx** in the Division Type field of the index (or TOC) division to generate.
- Edit the CI Spec for the job and fill in the *PDF Level* fields and *PDF Optional Data* fields.
- Run *CITI* from the Job Processing menu. (CITI is located on the sub-menu of the pop-up menu XPP displays when you right-click the job.)

Refer To

Refer to Chapter 5 “XPP Specs and Processes” for additional information.The following example shows a CITI-generated PDF index (or TOC) entry.

```
《rsv;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;7;7;3;1;5;7;0;0》
《:pdfs;INDEX;+7,05chap;/Color[0 0 1]》CI spec《in1》▣5-7
《:pdfe;INDEX》
```

- CITI inserts the *:pdfs;INDEX* XyMacro after the corresponding *rsv* XyMacro.
- The ”CI Spec”entry appears on page 5-7 in the XPP division, which is relative page +7 in *05chap*.
- The PDF index XyMacro contains optional RGB color data, /Color [0 0 1], which produces a blue border around text.
- The index XyMacros generate the same *pdfmark* operator as the 2PAGE XyMacros (to create a hot spot link). If you do not use CITI to generate a PDF index, you can insert either INDEX or 2PAGE XyMacros in text.
- If you insert the INDEX XyMacro manually in text, you can specify the sequence page number relative to the whole job by omitting the preceding plus sign and the division name. (For example, **《:pdfs;INDEX;24》** identifies an item on the 24th page of the PostScript file).

XREF Cross-reference

Use cross-reference PDF XyMacros to go to a location in the document that is marked with the *rx* (Mark Reference Spot) XyMacro, or to a named foot-note or pickup in the XPP job.

The numeric ID for XREF is 4.

Format The format of the cross-reference XyMacros is:

⟪:pdfs;XREF;*xref-name***;***opt-data***⟫**Hotspot-text**⟪:pdfe;XREF⟫**

—or—

⟪:pdfs;4;*xref-name***;***opt-data***⟫**Hotspot-text**⟪:pdfe;4⟫**

Comments During composition, XPP creates cross-reference information from *rx* (Mark Spot) XyMacros and from pickups and footnotes that are placed in the job. (To cross-reference an XPP story, use the ARTICLE and 2ART PDF XyMacros.)

To activate cross-references across multiple divisions in a job, you must use job processing options. You can activate cross-references in several ways:

Composing Divisions and Cross-References Together

1. Edit the Document Assembly Ticket and set the *Compose* field to **yes** and the *Resolve Xrefs across Job?* field to **yes**.
2. Select *Compose* from the Job Processing menu. First, XPP composes all *active* divisions in the Document Assembly Ticket. Then, XPP resolves cross-references in the active divisions by recomposing all lines with reference items.

Composing Divisions and Cross-References Separately

1. Edit the Document Assembly Ticket and set the *Compose* field to **yes** and the *Resolve Xrefs across Job?* field to **no**.
2. Select *Compose* from the Job Processing menu. (Right-click the job you want. XPP displays a pop-up menu. Select Processes from the pop-up menu. XPP displays a sub-menu, the Job Processing menu.) XPP composes all *active* divisions in the Document Assembly Ticket.
3. Select *Compose Xrefs* from the Job Processing menu. (Right-click the job you want. XPP displays a pop-up menu. Select Processes from the pop-up menu. XPP displays a sub-menu, the Job Processing menu.) XPP resolves cross-references within active divisions by composing only the lines with reference items.

List the target divisions as *active* PDF divisions in the Document Assembly Ticket so they are included in the PDF document. If a target division is missing from the PDF document, the PDF converter will fail with errors. For example, if you refer readers to a pickup in Chapter 9, the PDF document *must* contain the Chapter 9 division.

Note: *Reference items of the same type must have unique names across all divisions in a job. For example, you cannot have pickup "p1" in chap01 and pickup "p1" in chap05. However, different types of reference items can have the same name. For example, you can have a pickup named "monday," a footnote named "monday," and an* **《rx;monday》** *within the same job.*

XPP stores the cross-reference data in the *_job_xref* file at the job level, which lists the source reference name and type, and the "relative" target page in the PDF document. To check the names of reference items in your job, you can view the *_job_xref* file by selecting the *Show Xrefs* option from the Job Processing menu or running the *show_xref* utility from the command line.

Viewing the *_job_xref* file is helpful for checking whether a job contains duplicate reference item names or the target division for a reference item does not exist.

Refer To

Refer to Chapter 6 for information on processing cross-references in a job, on viewing the *_job_xref* file, and running the *show_xref* utility.
Refer to the XPP document *XyMacros* for information on the *rx* and *rf* XyMacros.

The following are examples of using PDF cross-reference XyMacros.

Example 1
Example 1 refers to a pickup named *messageA* in the XPP document. The pickup containing the message appears in Chapter 1. The reference to the message can appear in any division in the job.

```
This message will appear in the dialogue window: 《pick;
messageA;0;3》
```

```
You see a message similar to 《:pdfs;XREF;messageA;/Color [1 .5  0
]》Figure 1-1《:pdfe;XREF》 on page 《rf;messageA;3》 《ri;%refpg3》- 《
ri;%refpg4》 1-5.
```

Example 2
Example 2 refers to a marked spot named *using_alternate_keyboards* in Chapter 2 of the XPP document. The reference to the marked spot can appear in any division in the job.

```
{h1}Using Alternate Keyboards 《rx;using_alternate_keyboards》
```

```
《:pdfs;XREF;using_alternate_keyboards》Alternate keyboards
are《:pdfe;XREF》 listed on page 《rf;using_alternate_keyboards》
《ri;%refpg3》-《ri;%refpg4》 2-4.
```

Note: *In the preceding examples, the PDF cross-reference XyMacros* **⧼:pdfs;XREF;. . .⧽** *and* **⧼:pdfe;XREF⧽** *create hot links to marked spots within the PDF document.*

The XPP Reference Spot (rf) and Read Immediate (ri) XyMacros create cross-references to page numbers in the XPP divisions.

Refer To Refer to *XML Professional Publisher: XyMacros* for information about the Reference Spot and Read Immediate XyMacros.

ARTICLE Named Article

If you lay out a document to contain story blocks, or if your main text and pickups occur in a multi-column format or continue on non-consecutive pages, you can mark sections of the document that go together as PDF *articles*. When XPP processes the document, it creates threads linking sections of the articles, so when browsing the PDF document, the PDF viewer scrolls through the articles for you, moving from column to column or from page to page.

You create a PDF-threaded article in the XPP story stream by inserting start and end ARTICLE XyMacros at the beginning of the story, or on the page of the story where you want the article to begin. Once the reader goes to the article, PDF guides the reader through continued sections automatically until the end of the story.

You can also create an article that ties together XPP main text and pickups. When you enter the start and end article XyMacros in either the main text or pickup streams and name the article MAIN, the PDF viewer scrolls between main text and pickups on each page, starting from the page and location that first contains the article XyMacros until the end of the document.

In the main text and pickup streams the article name used in the article XyMacros is fixed and must be MAIN. In a story stream the article name used in the article XyMacros is fixed and must be the XPP story name.

Macros specifying a particular article name, including the name MAIN for main text and pickups, should be used only once in a document. Any other occurrences of a particular article name will be ignored.

The numeric ID for ARTICLE is 5.

Format The format of the start and end article XyMacros is:

⧼:pdfs;ARTICLE;*article-name***;***opt-data***⧽⧼:pdfe;ARTICLE⧽**

Examples In this example, once a reader goes to the start of the *weather* article in the *weather* story stream, the PDF viewer scrolls through columns and pages of the article until its end.

⧼:pdfs;ARTICLE;weather⧽⧼:pdfe;ARTICLE⧽"U.S. Weather Patterns"

In this example, once a reader goes to the start of the MAIN article, the PDF viewer scrolls between main text and pickups through the document to the last main text or pickup in the article.

《:pdfs;ARTICLE;MAIN》《:pdfe;ARTICLE》. . .

Comments The article XyMacros do not create a hot link in text. To create a hot link that takes a user to the start of a named article, use the 2ART XyMacro. Not supported with *Print Direct to PDF***.**

Note: *When browsing articles within a document, the PDF viewer expands each article block to fit the width of the window up to the maximum magnification allowed. In Adobe Reader, the reader can adjust this value by changing the Maximum ″Fit Visible″ Magnification field in the File → Preferences → General menu. The default is 800%; a smaller value, such as 200%, usually produces more satisfactory results for viewing articles.*

2ART Jump to Article

Use the Jump to Article XyMacros to create a hot link that takes you directly to an article within the PDF document that you have named with the Named Article XyMacros. Once you go to the start of an article, the PDF viewer guides you through continued article sections automatically when the reader moves to the end of a section. Not supported with *Print Direct to PDF.*

The numeric ID for 2ART is 10.

Format The format of the start and end Jump to Article XyMacros is:

《:pdfs;2ART;*article-name***;***opt-data***》**Hotspot-text**《:pdfe;2ART》**

Examples This sample XyMacro starts an article ″sunflower″ (in the *sunflower* story).

《:pdfs;ARTICLE;sunflower》《:pdfe;ARTICLE》Giant Sunflowers

Use the jump to article XyMacro to link “Tall plants” to the sunflower article (for the story named *sunflower*).

《:pdfs;2ART;sunflower》Tall Plants《:pdfe;2ART》

DEST Named Destination

Named destination XyMacros define a location that can be referenced by a link.

The numeric ID for DEST is 6.

Format The format of the start and end Named Destination XyMacros is:

《:pdfs;DEST;*dest-name***;***opt-data***》《:pdfe;DEST》**

Comments When the *Link to target position* field in the PDF spec is set to 'yes' and DEST is being used, with content between the **⟪:pdfs⟫** and **⟪:pdfe⟫**, XPP will generate a pdfmark that includes a position attribute of the form /View [XYZ x y 0], where *x* is slightly left of the start of the target content, *y* is slightly above the target content, and *0* is a *zoom* factor that means no change to the current zoom factor and page display mode.

Note: *Whether or not the page actually scrolls to that point in the viewer window depends on limitations in the viewer behavior.*

2DEST Go to Destination

The Go to Destination XyMacros jump you to a named destination that can be in the same PDF document or another PDF document.

The numeric ID for 2DEST is 9.

Format The format of the start and end Go to Destination XyMacros is:

⟪:pdfs;2DEST;*dest-name***;***opt-data***⟫**Hotspot-text**⟪:pdfe;2DEST⟫**

Examples These XyMacros name the *table4* destination in the same PDF document.

⟪:pdfs;DEST;table4⟫Table 4: Population Statistics⟪:pdfe;DEST⟫

These XyMacros take you to the destination named table4 in the same PDF document.

⟪:pdfs;2DEST;table4⟫Refer to table 4⟪:pdfe;2DEST⟫

You can also reference the named destination from a different document. Say that Table 4 is in *popstats.pdf*. Use the following XyMacros to reference Table 4 in *popstats.pdf* from a different document:

⟪:pdfs;2DEST;table4;/File (popstats.pdf)⟫See Table 4 in "Demographics of the Northeastern United States"⟪:pdfe;2DEST⟫

Specify the name of the PDF file (where you want the link to take you) in the optional data field of the XyMacro in the following format:
/File (filename.pdf)

NOTE Note

Notes provide a way for you to add comments to a PDF document. XPP supports the category of "simple notes" and uses the default size note window. The note text is contained within the Note XyMacro.

The numeric ID for NOTE is 7.

Format The format of the note XyMacro is:

《:pdfs;NOTE;*note-text***;***opt-data***》**Any/no-Text**《:pdfe;NOTE》**

Examples Here is a sample note XyMacro.

《:pdfs;NOTE;This technical note appears in red;/Color [1 0 0]》《:pdfe;NOTE》

Comments When the document is converted to PDF, a note icon indicates the presence of a note. To view note text from the PDF viewer, double-click on its icon. To help identify the type of comments in a note, you can color-code notes by category. For example, purple notes contain comments from the marketing department and green notes contain comments from the sales department.

2PAGE Go to Page

The Go to Page XyMacros jump you to a specific page in the PDF document.

The numeric ID for 2PAGE is 8.

Format The format of the start and end Go to Page XyMacros is:

《:pdfs;2PAGE;*seq-page#***;***opt-data***》**Hotspot-text**《:pdfe;2PAGE》**

Comments Be careful when using this XyMacro because the pages are numbered sequentially across all divisions that comprise a PDF document.

Examples You want to create a link that jumps to page 1-4 in Chapter 1. However, *psfmtdrv* prepends a title page and a three page Table of Contents to the PDF document. As a result, the fourth sequential page in the PDF document will be the last page of the Table of Contents, not page 1-4 of Chapter 1.

To jump to the fourth page in Chapter 1, you enter this PDF XyMacro:

《:pdfs;2PAGE;8》Getting Started《:pdfe;2PAGE》

If the table of contents increases to four pages, you would have to change the page number in the PDF XyMacro accordingly.

《:pdfs;2PAGE;9》Getting Started《:pdfe;2PAGE》

You can also specify a page number in a different document. To go to page 50 in *userguide.pdf* use the following XyMacro:

《:pdfs;2PAGE;50;/File (userguide.pdf)》See the ″User Guide″《:pdfe;2PAGE》

Specify the name of the PDF file (where you want the link to take you) in the optional data field of the XyMacro in the following format:

/File (filename.pdf)

LAUNCH Launch Another File

Launches another file or an application.

The numeric ID for LAUNCH is 11.

Format The format of the start and end XyMacros to Launch Another File is:

《:pdfs;LAUNCH;*filename***;***opt-data***》**Hotspot-text**《:pdfe;LAUNCH》**

Examples PDF Readers often have user preferences to determine whether cross-document links open in the same window or a different window and sometimes that behavior can be affected by whether the Ctrl key is pressed when clicking on the link.

For advanced users who know how to define a **pdfmark**, a command to the PDF converter, and who want the hotspot to always open in a new window, the optional data can be used to specify this.

《:pdfs;LAUNCH;file_name.pdf;/Action << /Subtype /Launch /File (file_name.pdf) /NewWindow true >>》Click Here《:pdfe;LAUNCH》

or if you also want the hotspot to have an orange colored border, the optional data can be as follows:

《:pdfs;LAUNCH;file_name.pdf;/Action << /Subtype /Launch /File (file_name.pdf) /NewWindow true >> /Color [1 .5 0]》Click Here《:pdfe;LAUNCH》

If you want to import the data using ToXSF or XyPerl, then you need to use special sequences for the double less-than and greater-than characters.

For example, <:pdfs;LAUNCH;.... @003c@003c ... @003e@003e ...>.

Another option for advanced users who know how to define a **pdfmark** is to use the RECT type.

WWW Link to WWW

Links to a location on the World Wide Web.

The numeric ID for WWW is 12.

Format The format of the Link to WWW start and end macros is:

《:pdfs;WWW;*URL***;***opt-data***》**Hotspot-text**《:pdfe;WWW》**

—or—

《:pdfs;12;*URL***;***opt-data***》**Hotspot-text**《:pdfe;12》**

Examples To link to the RWS home page, you enter the URL for RWS in the PDF XyMacro:

《:pdfs;WWW;https://www.rws.com》RWS《:pdfe;WWW》

Note: *If a URL includes a semi-colon, enter the hex value of a semi-colon—%3B—to avoid confusion with the semi-colon delimiter in XPP XyMacro syntax.*

Note: *There may be other characters, such as the &, that may be problematic with distiller. If the character is escaped, with a preceding \, the problem will likely be resolved.*

PS Pass-through PostScript Commands

The pass-through PostScript start XyMacro has one optional value that indicates whether level1 or level2 PostScript language code is being passed. If you omit the parameter, the default is level2. Not supported with *Print Direct to PDF*.

The numberic ID for PS is 13.

Format The format of the XyMacros is:

《:pdfs;PS;*pass-through-text***;***opt-data***》**Any/no-text**《:pdfe;PS》**

Note: *Beginning with Acrobat 5.0, the pass-through PostScript feature is no longer supported in any PDF files of version 1.4 or later.*

GEN Generic PDF Macros

XPP allows you to define a *pdfmark* directly by entering a complete pdfmark command, including starting bracket and ending keyword ’pdfmark’ as appropriate. The exact content of this pdfmark-string is written into the PostScript file.

The numeric ID for GEN is 14.

Format The format of the generic XyMacros is:

《:pdfs;GEN;*pdfmark-string***》**Any/No-text**《:pdfe;GEN》**

Comments This command is for advanced users who know how to define a **pdfmark**, a command to the PDF converter. Not supported with *Print Direct to PDF*.

Examples

This example generates a pdfmark that creates a PDF outline that opens the file named *test.doc*.

❰:pdfs;GEN;[/Action /Launch /File (test.doc) /Title (Open test.doc) /OUT pdfmark❱❰:pdfe;GEN❱

DOCINFO Pass Custom Information to the DOCINFO pdfmark

XPP allows you to change the data or add other data to the PDF Information Dictionary without changing the PDF Support Spec for each variation.

The numeric ID for DOCINFO is 15.

Format

The format of the XyMacro is:

❰:pdfs;DOCINFO;key-value pairs❱❰:pdfe;DOCINFO❱

Comments

Each key-value pair consists of a name and a string value in PDF syntax. The key-value pair must meet the following requirements:

- Each key must start with a slash and end with a space.
- Each value must be enclosed by parentheses or angle brackets.
- If there are multiple key-value pairs in a XyMacro, they must be separated by at least one space.
- If the key is one of the following—Title, Author, Subject, Keywords, Creator, Producer, CreationDate, ModDate—it overrides the corresponding value in the PDF Support Spec.

Note: *In divpdf, the Producer key is set to a fixed string defined by the PDFlib package, and cannot be changed.*

The PDF structure also lets you specify other key-value pairs for the Information Dictionary. XPP passes these through the PDF converter as well. Though these other key-value pairs do not appear in the PDF viewer's Document Information display, they may be available to specialized tools or Acrobat plug-ins.

Examples

PostScript requires that the text string of the value field be entered in either ASCII or ASCIIHex format, for example:

- ASCII strings contain the common 7-bit characters: A-Z, a-z, numerals and punctuation marks. In PostScript, an ASCII string is bounded with parentheses. For instance, the key/value part of a DOCINFO XyMacro might look like this:

  /Title (Zebras)
- ASCIIHex strings contain Unicode UCS-2 code point values, converted to an ASCII representation and bounded with angle brackets. Each Unicode character thus is expressed as four ASCII characters (0-9, A-F), and the string must start with the Unicode byte order mark

(FEFF). The title above, Zebras, could be written in an ASCIIHex string and look like this:

/Title <FEFF005A00650062007200610073>

- In XPP documents, however, there is often a need to use other languages and special characters that are entered as Unicode but cannot be stored in a PostScript ASCII string. XPP will sense when a parentheses-delimited value string of a DOCINFO XyMacro contains characters outside the ASCII range and convert it from the binary UTF-8 form to the ASCIIHex form so it will work in PostScript. For example, if the DOCINFO content is:

  /Subject (Café Français)

  there are non-ASCII characters that are not legal in a PostScript ASCII string. XPP will convert that to the ASCIIHex form:

  /Subject <FEFF00430061006600E90020004600720061006E00E7006100690073>

RECT Generic with hotspot rectangle PDF Macros

XPP allows you to define a *pdfmark* directly (like with the GEN type) by entering most of a complete pdfmark command, including the ending keyword 'pdfmark' as appropriate. The starting bracket and calculated hotspot rectangle information are written out first. Then the exact content of this pdfmark-string is written into the PostScript file.

The numeric ID for RECT is 16.

Format The format of the generic with hotspot rectangle XyMacros is:

《:pdfs;RECT;*pdfmark-string***》**Hotspot-text**《:pdfe;RECT》**

Comments This command, like the GEN type, is for advanced users who know how to define a **pdfmark**, a command to the PDF converter. The RECT type includes first writing out the calculated hotspot rectangle information for the pdfmark. Not supported with *Print Direct to PDF*.

Examples This example defines a hotspot with an orange colored border that always opens the file named *userguide.pdf* in a separate window.

《:pdfs;RECT;/Action << /Subtype /Launch /File (userguide.pdf) /NewWindow true >> /Color [1 .5 0] /Subtype /Link /ANN pdfmark》See the "User Guide"《:pdfe;RECT》

If you want to import the data using ToXSF or XyPerl, then you need to use special sequences for the double less-than and greater-than characters.

For example, <:pdfs;RECT;.... @003c@003c ... @003e@003e ...>.

Coloring or Changing Borders

As described previously in this section, you can enter RGB color values in the optional-data field in several of the PDF XyMacros to color borders around the text. For example, /Color [0 0 1] produces a blue border, /Color [0 1 0] produces a green border, and/Color [1 0 0] produces a red border. You can also enter /Color [1 1 1] to draw a white border, making it invisible on a white background.

If you want to suppress the border altogether, use the following syntax in the optional-data field: /Border [0 0 0].

If you want to import the data using ToXSF (in Classic mode), then it is important that you escape the square brackets.

For example, <:pdfs;2DEST;start;/Border \[0 0 0\]>text<:pdfe;2DEST>.

If you want to use a different border width than the default, use the following syntax in the optional data field: /Border [0 0 #]. The # is the desired border width in points, specified in XPP units. For example, a border width of 1q would be specified with /Border [0 0 353]; a border width of 2q would be specified with /Border [0 0 706]. The border width cannot be a fractional point (i.e. it must be specified in /Border as a multiple of 353).

Coloring or Setting Font Style of Bookmarks

For the TOC and BOOK PDF XyMacro types, you can also specify the color and font style of the bookmark text that appears in the PDF Bookmarks window.

If you want to specify the color of the bookmark text, use the following syntax in the optional data field: /C [# # #]. The # values are the RGB color values. The default is black.

If you want to specify the font style of the bookmark text, use the following syntax in the optional data field: /F #. The # value is 0 for plain (the default), 1 for italic, 2 for bold, and 3 for bold italic.

Note: For the TOC PDF XyMacro type, if you are using the /C optional data to color the bookmark text and you want the hotspot border to be a different color (including black), then you must also specify the /Color optional data for the color of the hotspot border (and *after* the /C optional data). In this case, use /Color [0 0 0] if you want a black hotspot border.

Chapter 4

PDF Support Spec

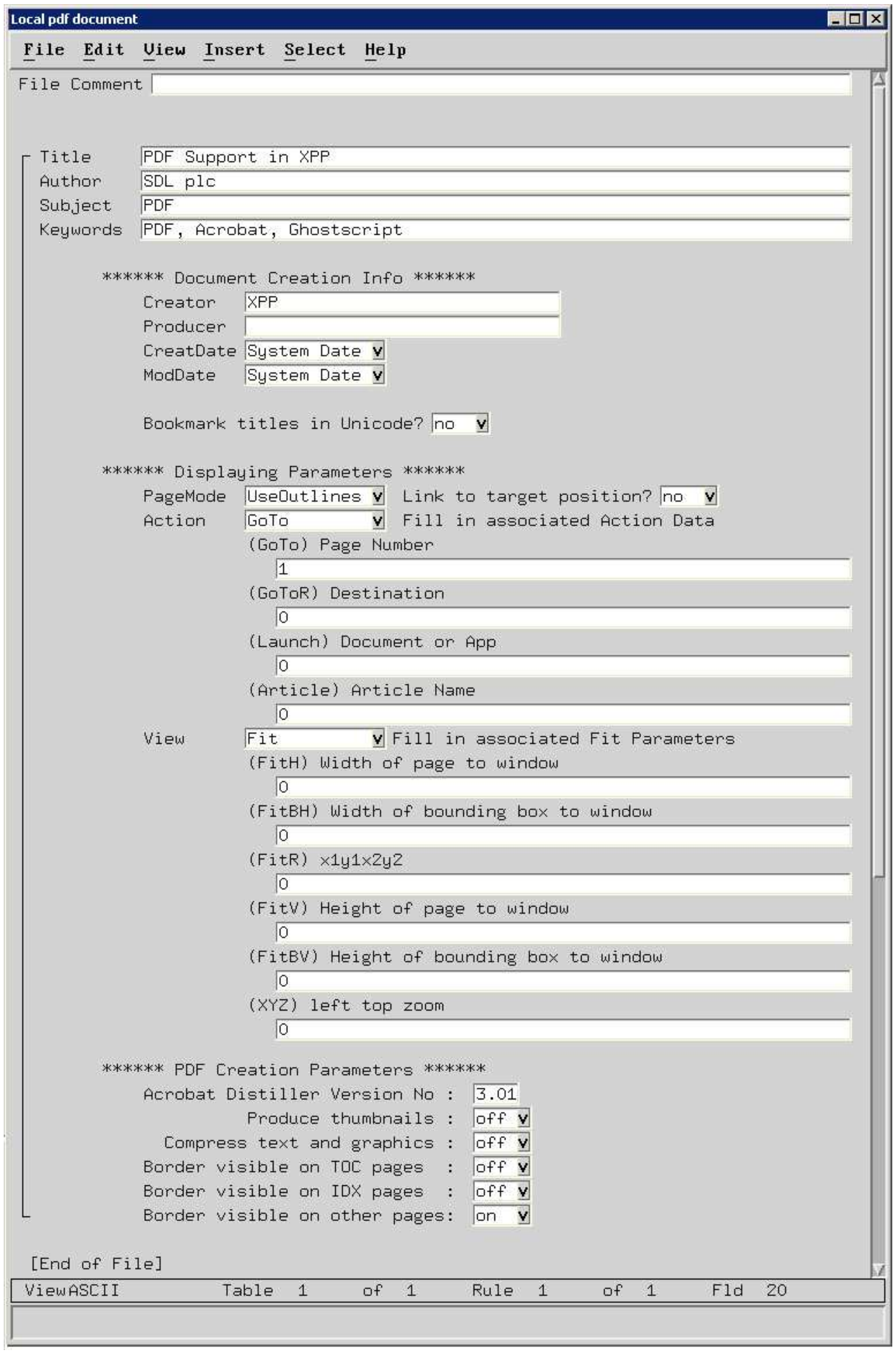
The PDF Support Spec specifies attributes for the resulting PDF document, including data for the PDF Information Dictionary, the Catalog Dictionary, and the PDF converter.

This chapter:

- Describes the function of the PDF Support Spec
- Explains how to access the spec
- Discusses the role of the PDF Support Spec when converting a document
- Shows a sample PDF Spec
- Describes how to fill in fields in the PDF Spec
- Describes the *_distill_parms* file
- Describes how to enable Ghostscript using the *gsdistill* file

PDF Support Spec

You must create a PDF Support Spec at the job level for each document that will be converted to PDF.



```
File Comment

Title       PDF Support in XPP
Author      SDL plc
Subject     PDF
Keywords    PDF, Acrobat, Ghostscript

      ****** Document Creation Info ******
          Creator    XPP
          Producer
          CreatDate  System Date
          ModDate    System Date

          Bookmark titles in Unicode? no

      ****** Displaying Parameters ******
          PageMode   UseOutlines   Link to target position? no
          Action     GoTo          Fill in associated Action Data
                     (GoTo) Page Number
                       1
                     (GoToR) Destination
                       0
                     (Launch) Document or App
                       0
                     (Article) Article Name
                       0
          View       Fit          Fill in associated Fit Parameters
                     (FitH) Width of page to window
                       0
                     (FitBH) Width of bounding box to window
                       0
                     (FitR) x1y1x2y2
                       0
                     (FitV) Height of page to window
                       0
                     (FitBV) Height of bounding box to window
                       0
                     (XYZ) left top zoom
                       0

      ****** PDF Creation Parameters ******
          Acrobat Distiller Version No :  3.01
                     Produce thumbnails :  off
            Compress text and graphics :  off
          Border visible on TOC pages   :  off
          Border visible on IDX pages   :  off
          Border visible on other pages:  on

 [End of File]
 ViewASCII          Table  1    of  1    Rule  1    of  1    Fld  20
```

Figure 4-1 *Sample PDF Spec*

Accessing the PDF Spec

Before creating a PDF-enhanced PostScript file from an XPP job, you must fill out a PDF Support Spec for the job. The PDF Spec contains four categories of fields:

- File Comment
- Information Dictionary
- Catalog Dictionary
- PDF Creation Parameters

Information for filling out the spec begins on page 4-6.

Create a New PDF Style Spec

You may want to create a new PDF Style Spec for a particular job.

To create a new PDF Style Spec:

1. Right-click the **job**.
   XPP displays a pop-up menu.
2. Select **New Style File > PDF** from the pop-up menus.
   XPP performs these tasks:
   - Follows the library chase looking for the PDF template called _pdf_sys.sde.
   - Displays the PDF template.
     You can use this template to change the defaults and store it at the job level.

About the Template Spec

A default PDF Spec, **pdf_sys**, resides in the *syslib* library. When you first open a PDF Spec for editing, XPP copies the template spec it locates by following your library chase (*syslib* is the default location) to the job level and saves your changes when you select *Store/Exit*. You now have a PDF Spec at the job with the changes, if any, that apply to that job.

If you want some PDF Spec fields to always have the same value regardless of the job, edit the template spec in the spec library listed in your Job Ticket. If XPP does not locate a template in the JT, it searches *syslib*. Set those fields to the appropriate value. For example, if you always want the PDF documents to open and display the same way, modify the *View* field in the template PDF Spec.

Name a PDF Spec in the Job Ticket

You must enter the name of the PDF Spec in the *PDF Support* field of the Job Ticket for XPP to access the PDF Support Spec and create a PDF version of your job.

Editing the PDF Spec at the Job Level

To edit the PDF Spec:

1. In PathFinder Tree View, select the **job** containing the PDF Spec you want to edit.
2. Click the **PDF** icon under the job.

   In the List View, PathFinder displays the PDF Specs stored with the job.
3. Right-click the **PDF** Spec you want to access.

   XPP displays a pop-up menu.
4. Select **Edit** from the pop-up menu.

   XPP displays the PDF Spec.
5. Enter information appropriate to the specific job in the spec fields.

PDF Spec in PostScript File Name

When XPP creates a PostScript disk file for conversion into a PDF document, it appends the name of the PDF Spec to the job name.

- The name of the PostScript disk file is *job_pdfspecname*.ps
- The name of the PDF document is *job_pdfspecname*.pdf

For example, an XPP job *geography,* using the PDF Spec *site-a*, the output produces the following files:

- **geography_site-a.ps**
- **geography_site-a.pdf**

Note: *If you create the PDF at the division level, the default also includes the division name.*

Refer to Chapter 2, page 2-12, for a complete explanation of naming conventions.

To override the default disk file name, you can set the *pn* (PostScript name) switch in the Print dialog box or in Device Options file (DOF Spec). Refer to Chapter 5 for information about setting device option switches.

File Comment Field

The File Comment field at the top of the spec contains general information about the PDF Support Spec. The comments do not appear in the PDF document.

File Comment

Enter a comment about the PDF Support Spec, such as the reason for editing the spec, the date, and your initials.

Entry	***Description***
[*string*]	Allows a maximum of 7½ lines (512 characters, including uppercase and lowercase characters, spaces, symbols (such as $, &, /), and the integers 0-9).

Information Dictionary Fields

The Information Dictionary contains general information about a PDF document such as its title, author, subject, keywords, creator, producer, creation date, and date last modified.

When viewing a PDF document with Adobe Reader, you can access this information by selecting *Document Information* from the *File* menu.

Fill in the following fields in the PDF Spec to add data to the Information Dictionary during the distillation process.

Note: *The DOCINFO XyMacro overrides the Information Dictionary fields.*

Title

The document name appears in the *Title* information field.

Entry	*Description*
[*string*]	Enter up to 512 alphanumeric characters.

Author

The name of the organization or person who created the document appears in the *Author* information field.

Entry	*Description*
[*string*]	Enter up to 512 alphanumeric characters.

Subject

The document subject or category appears in the *Subject* information field.

Entry	*Description*
[*string*]	Enter up to 512 alphanumeric characters.

Keywords

Words that summarize the main topic of the document, or that appear frequently in the document, appear in the *Keywords* information field. Keywords are used primarily for cross-document searches.

Entry	*Description*
[*string*]	Enter up to 512 alphanumeric characters.

Document Creation Information

This section refers to the software that created the document and the date it was used to create or modify the document.

Creator

The software application that created the source document appears in the *Creator* information field.

Entry	*Description*
XPP	Up to 30 alphanumeric characters. Defaults to XPP.

Producer

The software application that converted the source document to PDF appears in the *Producer* information field.

Entry	*Description*
string	Up to 3 alphanumeric characters.

Note: *In divpdf, the Producer key is set to a fixed string defined by the PDFlib package, and cannot be changed.*

CreatDate

The date the document was created appears in the *Created* information field.

Entry	*Description*
0 or **System Date**	The system supplies the date. (default)
mm/dd/yy	Enter a user-supplied date in this format.

ModDate

The date the document was last modified appears in the *Modified* information field.

Entry	*Description*
0 or **System Date**	The system supplies the date. (default)
mm/dd/yy	Enter a user-supplied date in this format.

Bookmark Titles in Unicode?

This field lets you generate PDF bookmarks in Unicode.

Entry	*Description*
yes	Specifies that the XyMacros that generate title text for PDF bookmarks (the BOOK and TOC XyMacro types) encode that text in Unicode.
no	Specifies that the XyMacros encode that title text in the default PDFDocEncoding (a superset of ISO Latin 1).

Comments Using Unicode lets you have titles in languages that are not supported in ISO Latin 1. However, Unicode requires twice the number of bytes per character; therefore, the Unicode title can contain only half as many characters as the default PDFDocEncoding. Consequently, the following limitations are in effect:

- Adobe Acrobat Distiller software limits the length of Bookmark-text to 255 PDFDocEncoding (8-bit) characters or 126 Unicode (16-bit) values.
- The same limit applies to TOC-text since BOOK and TOC XyMacros generates the same pdfmark.
- If BOOK-text or TOC-text exceed the limits—255 PDFDocEncoding characters or 126 Unicode encoding values—XPP truncates the text at that point to avoid PDF generation errors.
- The Adobe pdfmark Reference Manual recommends “a practical limit of 32 characters...so it can be read easily in the Acrobat Viewer.”

Catalog Dictionary Fields (Displaying Parameters)

The Catalog Dictionary contains information about the PDF document such as an action that occurs when the document opens and how to display the open document.

When viewing a PDF document with a PDF viewer, you can modify the way a document appears by selecting options from the *View* menu.

Fill in the following fields in the PDF Spec to add data to the Catalog Dictionary during the PDF generation process.

PageMode

How the PDF document appears when opened in a PDF viewer.

Entry	*Description*
UseNone	Opens the document without displaying bookmarks or thumbnail images. (default)
UseOutlines	Opens the document and displays bookmarks and table of contents text. If you use CITI to generate a PDF table of contents, the table of contents (in collapsible outline form) also appears to the left of the main PDF viewer window, with links and hot spots to help you navigate through the document.
UseThumbs	Opens the document and displays thumbnail images of pages. If you want to display thumbnails, you must activate thumbnails when distilling the file. Set the *Produce thumbnails* distillation field to **on** and/or set the *–thumbs on* switch in the _distill_parms file. For additional information, refer to the section "The _distill_parms and _gsdistill_parms Files" in Chapter 2. ***Note:*** *Acrobat Distiller 3.0 does not produce thumbnail images.*
FullScreen	Opens the document in full-screen mode.

Link to target position?

Allows user to set a target point when using TOC, BOOK, or DEST macros.

Entry	*Description*
yes	Allows a link to a target point on a page rather than the top of the page that contains the target. This applies to properly coded TOC, BOOK, or DEST passthrough macros.
no	Will link to the top of the page that contains the target.

Comments The TOC, BOOK, or DEST macros must be coded such that the target content occurs between the **《:pdfs》** and **《:pdfe》** macros. If there is no content between these macros, no point is defined and the link can only address the page.

When the option is selected in the PDF spec and a supported passthrough macro is encountered, XPP will generate a pdfmark that includes a position attribute of the form /View [XYZ x y 0], where *x* is slightly left of the start of the target content, *y* is slightly above the target content, and *0* is a *zoom* factor that means no change to the current zoom factor and page display mode.

Note: *Whether or not the page actually scrolls to that point in the viewer window depends on limitations in the viewer behavior.*

Action

Specifies an action to perform when the PDF document opens in a PDF viewer.

Entry	*Description*
GoTo	Fill in *(GoTo) Page Number* field. (default)
GoToR	Fill in *(GoToR) Destination* field.
Launch	Fill in *(Launch) Document or Application* field.
Article	Fill in *(Article) Name* field.

(GoTo) Page Number

Jump to a specific page within the current PDF document. This field is active when the *Action* field is set to **GoTo**.

Entry	*Description*
pagenumber or /Page pagenumber	The sequential number of a page in the PDF document. The default is **1**— the first page of the PDF document.

Comments	Use this field carefully. PDF accesses pages sequentially from the start of the document, which may contain all divisions in an XPP job. You may enter just the pagenumber or the PostScript format of **/Page pagenumber**.
Example	You want to jump to the fourth page in Chapter 1. If you enter Go to /Page 4, PDF counts to the fourth page from the start of the document. However, if you prepend a two-page table of contents, PDF opens the file on the fourth page from the start of the document, which is numbered page 2 in Chapter 1.

(GoToR) Destination

Name of another PDF document to open. This field is active when the *Action* field is set to **GoToR**.

Entry	*Description*
pathname	The device-independent path name of a PDF document.

Note: *For either the (GoToR) Destination or (Launch) Document or App fields, the path name entry is taken as a string so you can enter a path name of any format. It is advisable, however, to make the path name as unrestricted as you can, ideally, only the PDF file name. This instructs the PDF viewer to look for the file in the same directory.*

If you enter an absolute path name, then on any system where you want to view the PDF file, you must have a directory with that path name or the link will not be usable.

(Launch) Document or App

Name of a document or application to launch. This field is active when the *Action* field is set to **Launch**.

Entry	*Description*
path name	The device-independent path name of another PDF document to open or an application to execute.

(Article) Article Name

Name an article (story) to begin reading when the PDF document opens.

Entry	*Description*
string	The name of an article in the current PDF document. In XPP, an article is a story.

View

Defines which view of the destination page to display in the main window when the PDF document first opens.

You can use the zoom tool, the magnification box in the status bar or the toolbar buttons to change the magnification after the document opens. Maximum magnification is 800% and minimum magnification is 12%.

Entry	*Description*
Fit	Fit the page to the window. Has no parameters. (default)
FitB	Fit the bounding box of the page to the window. Has no parameters.
FitH	Fit the width of the page to the window. Set parameters in *(FitH) Width of page to window* field.
FitBH	Fit the width of the bounding box of the page contents to the window. Set parameters in the *(FitBH) Width of page to window* field.
FitR	Fit the rectangle to the window. Set parameters in the *(FitR)x1y1x2y2* field.
FitV	Fit the height of the page to the window. Set parameters in *(FitV) Height of Bounding Box to Window* field.
FitBV	Fit the height of the bounding box of the page contents to the window. Set parameters in the *(FitBV) Height of bounding box to window* field.
XYZ	Place the window as specified by coordinates and display at specified zoom factor. Set parameters in the *(XYZ) left top zoom* field.

Default User Space

Parameters for the options to the *View* field are specified in *default user space*, a device-independent, coordinate system used by PDF to ensure that files always look the same regardless of an output device's resolution.

The system maps a grid of x,y coordinates to the PDF page, that is, a single x,y pair identifies a point on the page. Measurement is in default user space units where *x* specifies horizontal measure and *y* specifies vertical measure.

The coordinates **0,0** (also known as the *page origin*) identify the extreme bottom and left of the page; measurement is relative to point 0,0. So point *32,121* is 32 units to the right of and 121 units above 0,0.

There are 72 units of default user space in one inch. One unit of default user space is equivalent to a typographic *point* (q); 12 units is equivalent to a *pica*, etc. An 8½ by 11 inch page is 612 by 792 default user space units.

(Fit) Fit page to window

Scale so the entire page displays in the PDF viewer window.

There are no parameters; select this option in the *View* field.

Note: *The zoom factor for the Fit and FitB options is determined by the size of the PDF viewer window when you open the document.*

(FitB) Fit Bounding Box of page to window

Scale so the Bounding Box of the page fits the PDF viewer window. There are no parameters; select this option in the *View* field.

The *bounding box* of a page is the smallest rectangle imposed over a page that includes all the objects printed on that page.

(FitH) Width of page to window

Display the full page width, and as much of the page depth, beginning from the point defined by *top*, that fits in the PDF viewer window.

Entry	*Description*
top	The distance in default user space above the page origin point.

Example If a document has an 8½ by 11 page (612 by 792 in default user space), and you choose *FitH* with 792 as the value for *top*, you see the full page width and as much page depth, beginning from the page *top* that fits in the PDF viewer window.

If you specify *top* as 400, you see the full page width. The depth display begins from approximately the middle of the page.

(FitBH) Width of bounding box to window

Display the full bounding box width and as much of the page depth, beginning from the point defined by *top*, that fits in the PDF viewer window.

Entry	*Description*
top	The distance in default user space above the page origin point.

Comments *FitBH* behaves like *FitH* with one exception— FitBH uses the bounding box width. See *FitB* for a description of bounding box width.

(FitR) x1y1x2y2.

Display the rectangle defined by coordinates **x1, y1** and **x2, y2** scaled to fit the PDF viewer window.

Entry	*Description*
x1 y1 x2 y2	The lower-left x, lower-left y, upper-right x, and upper-right y coordinates of a rectangle expressed in default user space units.

Comments All coordinates are relative to the page origin (0,0). The coordinates define the lower left and upper right corners of a rectangle from the page that you want to display.

Example To display the top right quadrant of a page in 7 by 9 inch format, identify the page's middle point for the bottom left corner of the rectangle and the page's top right corner for the top right corner of the rectangle. Since default user space units are 72 to the inch, the top right corner will be 7*72 and 9* 72 or 504 and 648 respectively. To find the bottom left corner (the pages' middle point) divide each of those values by 2.

Select *FitR* and enter 252 324 504 648.

(FitV) Height of page to window

Display the full page depth and as much of the page width, beginning from the point defined by *left*, that fits in the PDF viewer window.

Entry	*Description*
left	The distance in default user space to the right of the page origin where you want the left edge of the page display to begin when the full width can not fit in the PDF viewer window.

Comments This option is particularly useful when the pages of a document are printed in landscape format. *FitV* lets you specify a point along the x-coordinate axis to be the leftmost edge of the page display when the aspect ratio of the PDF viewer window does not allow the full page width to display. You can then scroll right or left using the horizontal scroll bar to see the entire page.

Example If you have a document with an 11 by 17 page that is printed in landscape format, and if the PDF viewer window cannot display the entire page width when it first comes up, you may want the leftmost edge of the page display to be the center point of the page. Select *FitV* and enter 612 ((17*72)/2) as the value for left.

(FitBV) Fit Bounding Box Height of page to window

Display the full bounding box depth and as much of the page width, beginning from the point defined by *left*, that fits in the PDF viewer window.

Entry	*Description*
left	The distance in default user space to the right of the page origin where you want the page display to begin when the full page width cannot fit in the PDF viewer window.

Comments *FitBV* behaves like *FitV* with one exception— FitBV uses the bounding box width. See *FitB* for a description of bounding box.

XYZ left top zoom

Display the page according to the coordinates specified and at the specified zoom factor.

Entry	*Description*
left top zoom	*left* and *top* specify a top left corner from which the display begins. *zoom* specifies the zoom factor, with 1 being 100% magnification, 1.25 being 125% magnification, etc.

Comments The specified zoom factor always applies, that is, it is not dependent on parameters of the Acrobat Reader window.

Example It is common to want to display from the upper left corner of the page at a magnification of 100%. If you had an 8½ by 11 format, select *XYZ* in the View field and enter -4 800 1.0 in *(XYZ) top left zoom*. This slightly offsets the top left corner of the page from the PDF viewer window and displays as much of the page, at 100% magnification, that fits in the window.

PDF Creation Parameters Fields

The PDF Support Spec contains fields where you can specify parameters used by the PDF converter, such as whether to produce thumbnail images, use compression algorithms, and display borders.

If you use job processing to create PDFs from the PostScript file, you can specify other parameters in the ASCII *_distill_parms* file or *_gsdistill_parms* file at the job level or the *gsdistill* file at the system level. During job processing, XPP passes parameters in the *_distill_parms* or *_gsdistill_parms* and *gsdistill* files to the PDF converter.

Values in the *_distill_parms* or *_gsdistill_parms* and *gsdistill* files override those in the PDF Support Spec. For example, if the *_distill_parms* file contains the entry **–thumbs off** and you set the *Produce thumbnails* field in the PDF Spec to **on**, the PDF converter will not produce thumbnail images for document pages.

Setting parameters can affect the amount of time it takes to generate the document. For example, producing thumbnails increases the generation time considerably.

See *Appendix B: PDF Creation Options in Ghostscript* for more information about distiller parameters and parmeter files.

Acrobat Distiller Version No

Enter the version of the Acrobat Distiller that you are running. The value in this field determines whether any action is taken on the *Produce thumbnails* and *Compress text and graphics* fields.

Note: *This field is ignored if you use Ghostscript to generate PDFs.*

Entry	*Description*
user specified	A four character field where you must specify the Distiller version.

Comments Enter only the digits of the version number and a period, such as *2.1* or *3.0*. Do not enter any alphabetic characters such as *V2.1*.

The default is 3.0. If you are running a version other than 3.0, you must explicitly enter the version number.

Produce thumbnails

Set this field to determine whether or not to produce thumbnail images of pages in the document. Thumbnails are miniature views of each page in a PDF document and are useful when navigating through the document.

Entry	*Description*
on	Produce thumbnails.
off	Do not produce thumbnails. (default)

This field is ignored if it is set to **on** and you are running Distiller V3.0 or later or if you use Ghostscript to generate PDFs. Distiller V4.0 and later use an Adobe PDF settings file to control this option.

Compress text and graphics

Set this field to determine whether to compress text and graphics using LZW compression.

Entry	*Description*
on	Compress text and graphics. (default)
off	Do not compress text and graphics.

This field is ignored if it is set to **on** and you are running Distiller V4.0 or later or if you use Ghostscript to generate PDFs. Distiller V4.0 and later use an Adobe PDF settings file to control this option.

Border visible on TOC pages

Set this field to determine whether to display a border around each hot spot on a table of contents page.

Entry	*Description*
on	Display a border. (default)
off	Do not display a border.

Border visible on IDX pages

Set this field to determine whether to display a border around each entry on an index page.

Entry	*Description*
on	Display a border. (default)
off	Do not display a border.

Border visible on other pages

Set this field to determine whether to display borders around hotspots on pages within the document body. This field functions just like the fields for TOC pages and IDS pages.

Entry	*Description*
on	Display a border. (default)
off	Do not display a border

Chapter 5

XPP Specs and Processes

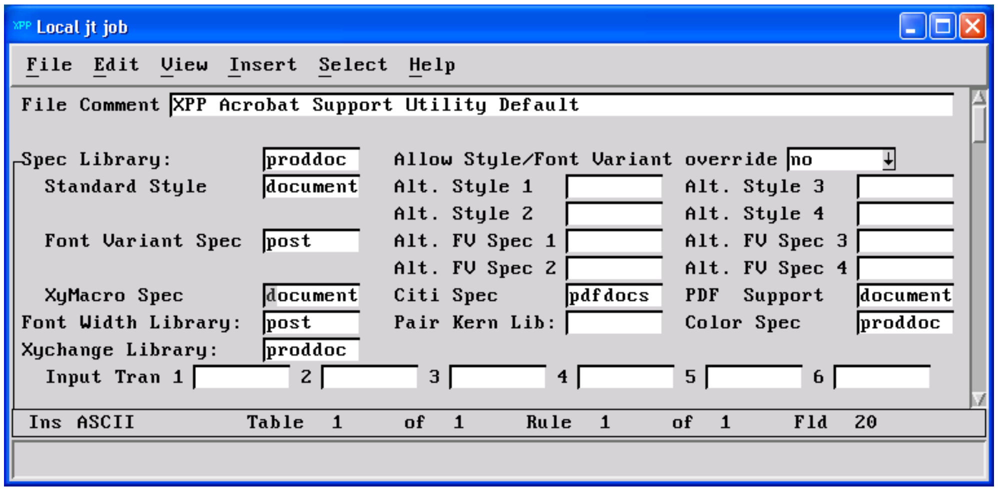
This chapter describes XPP specs and processes associated with creating PDFs , including the following:

- Job Ticket
- Document Assembly Ticket
- CI Spec
- Background Queue Spec
- Device Options File

Job Ticket

Before converting an XPP job to a PDF document, edit the Job Ticket and fill in these fields:

- *PDF Support* — Enter the name of the PDF Support Spec that specifies PDF attributes for the resulting PDF document.
- *Citi Spec* — If you are using CITI to generate a PDF table of contents or index, enter the name of the CI Spec in this field.



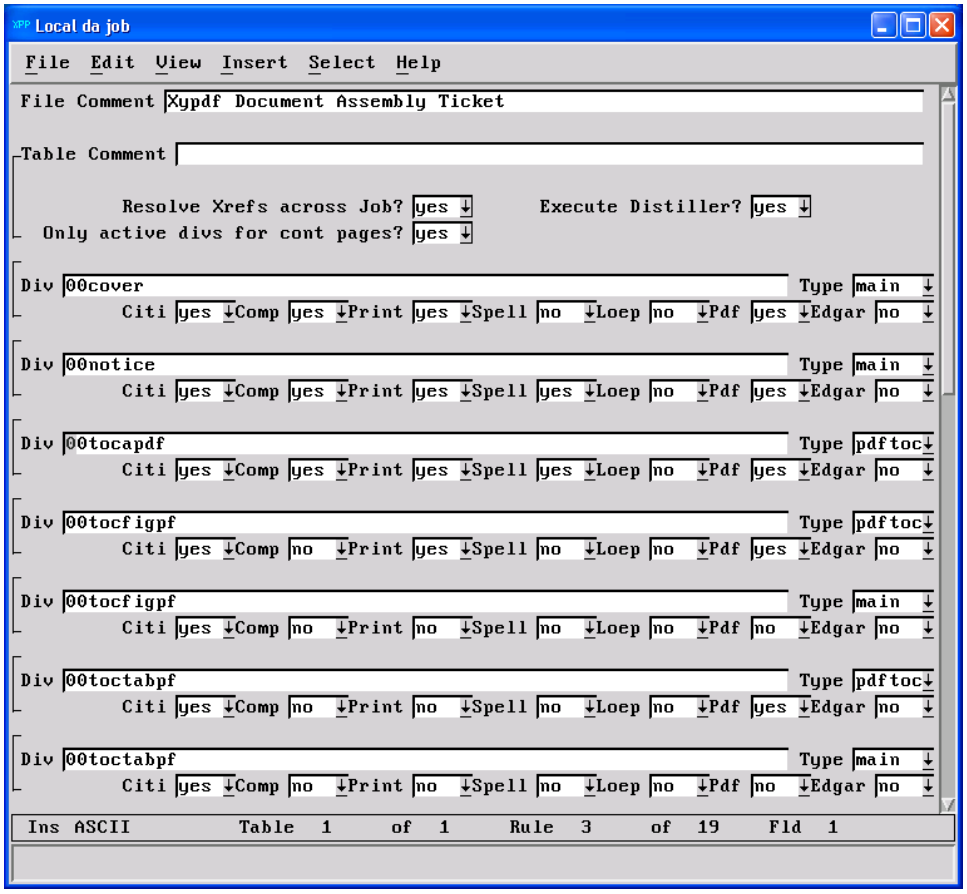
```
File Comment XPP Acrobat Support Utility Default

Spec Library:        proddoc    Allow Style/Font Variant override no
  Standard Style     document   Alt. Style 1              Alt. Style 3
                                Alt. Style 2              Alt. Style 4
  Font Variant Spec  post       Alt. FV Spec 1            Alt. FV Spec 3
                                Alt. FV Spec 2            Alt. FV Spec 4
  XyMacro Spec       document   Citi Spec      pdfdocs    PDF  Support  document
Font Width Library:  post       Pair Kern Lib:            Color Spec    proddoc
Xychange Library:    proddoc
  Input Tran 1          2          3          4          5          6
 Ins  ASCII          Table   1     of   1      Rule   1     of   1      Fld   20
```

Figure 5-1 *Partial Job Ticket for PDF Processing*

Document Assembly Ticket

The Document Assembly Ticket tells XPP the following:

- The names of main divisions that contain marked text to extract for a PDF table of contents or PDF index.
- The names of PDF table of contents and PDF index divisions that will be generated by CITI.
- Whether to resolve cross-references across the job by composing active divisions first, then recomposing all lines with reference items.
- Which divisions to include in the PostScript file.
- Whether to execute a PDF converter on the PostScript file.



```
File Comment Xypdf Document Assembly Ticket

Table Comment

        Resolve Xrefs across Job? yes       Execute Distiller? yes
  Only active divs for cont pages? yes

Div 00cover                                              Type main
       Citi yes Comp yes Print yes Spell no  Loep no Pdf yes Edgar no

Div 00notice                                             Type main
       Citi yes Comp yes Print yes Spell yes Loep no Pdf yes Edgar no

Div 00tocapdf                                            Type pdftoc
       Citi yes Comp yes Print yes Spell yes Loep no Pdf yes Edgar no

Div 00tocfigpf                                           Type pdftoc
       Citi yes Comp no  Print yes Spell no  Loep no Pdf yes Edgar no

Div 00tocfigpf                                           Type main
       Citi yes Comp no  Print no  Spell no  Loep no Pdf no  Edgar no

Div 00toctabpf                                           Type pdftoc
       Citi yes Comp no  Print no  Spell no  Loep no Pdf yes Edgar no

Div 00toctabpf                                           Type main
       Citi yes Comp no  Print no  Spell no  Loep no Pdf no  Edgar no

 Ins ASCII        Table  1    of  1     Rule  3    of  19    Fld  1
```

Figure 5-2 *Document Assembly Ticket for PDF Processing*

Follow standard procedures to set up the Document Assembly Ticket. In addition, enter data in these fields.

Division Type

Defines the division as main, citi, pdftoc, pdfidx, or loep.

Entry	***Description***
main	Identifies a main division (default) for input to CITI. All main (user-created) divisions must already exist in the job.
citi	Identifies a computer-generated division that does not include PDF XyMacros. A *citi* division can contain table of contents entries, index entries, or lists of figures or tables. The CITI software creates a CITI division when you select *CITI* from the Job Processing menu. (Right-click the job you want. XPP displays a pop-up menu. Select Processes. XPP displays a sub-menu listing all the job processes you can perform at the job level.) Normally, CITI divisions are not involved in PDF processing.
pdftoc	Identifies a computer-generated division in which CITI places TOC start and end PDF XyMacros around extracted text, after the *rsv* (Register Save) XyMacro. Use to create a PDF table of contents division. In some cases you may want to use the **pdfidx** type to create a PDF table of contents division. The CITI software creates a *pdftoc* division when you select *CITI* from the Job Processing menu. (Right-click the job you want. XPP displays a pop-up menu. Select Processes. XPP displays a sub-menu listing all the job processes you can perform at the job level.)
pdfidx	Identifies a computer-generated division in which CITI places INDEX start and end PDF XyMacros around extracted text, after the *rsv* (Register Save) XyMacro. Use to create a PDF index division, or in some cases a PDF table of contents division. The CITI software creates a *pdfidx* division when you select *CITI* from the Job Processing menu. (Right-click the job you want. XPP displays a pop-up menu. Select Processes. XPP displays a sub-menu listing all the Job processes you can perform at the job level.)
loep	Identifies an LOEP (list of effective pages) division. The Loose-leaf software creates the LOEP division when you select LOEP from the Job Processing menu. (Right-click the job you want. XPP displays a pop-up menu. Select Processes. XPP displays a sub-menu listing all the job processes you can perform at the job level.) Loep divisions are not involved in PDF processing.

Comments If the division type is *pdftoc* or *pdfidx*, CITI inserts the appropriate PDF XyMacro pairs after the *rsv* (Register Save) XyMacros. XPP composition does not act on the XyMacro contents. Instead, the PostScript formatter *psfmtdrv* converts the XyMacros to the appropriate *pdfmark* operators.

If you use CITI to generate non-PDF indexes and tables of contents (*citi* divisions), you can continue to do so. Or, you may choose to generate only *pdfidx* and *pdftoc* divisions that contain both *rsv* XyMacros and PDF XyMacros. The resulting XPP divisions are visually identical to *citi* divisions and they can be output to PostScript printers through normal processing. (The PDF XyMacros are ignored).

If a document contains cross-references across multiple divisions, list the target divisions as *active* PDF divisions in the Document Assembly Ticket so they are included in the PDF document. If a target division is missing from the PDF document, the PDF converter fails with errors. For example, if you refer readers to a pickup in Chapter 9, the PDF document *must* contain the Chapter 9 division.

CITI

Tells CITI whether to process a main division and whether to create a computer-generated division.

Entry	*Description*
yes	CITI processes this division when you select *CITI* from the Job Processing menu. (Right-click the job you want. XPP displays a pop-up menu. Select Processes. XPP displays a sub-menu listing all the job processes you can perform at the job level.) (default) Enter **yes** in this field for every *pdftoc*, *pdfidx*, and *citi* division you want CITI to create, and for each *main* division you want to process. CITI inserts a + (plus sign) before the page numbers of entries in *pdftoc* and *pdfidx* divisions, followed by the *main* division name, which tells the PostScript formatter *psfmtdrv* to calculate the sequence page numbers in the PDF document.
no	CITI does not process this division.

Comments There are three types of computer-generated divisions—*citi*, *pdftoc* and *pdfidx*. Be sure to run *CITI* from the Job Processing menu (right-click the job you want, XPP displays a pop-up menu, select Processes, XPP displays a sub-menu listing all the job processes you can perform at the job level) to create *pdftoc* and *pdfidx* divisions before selecting *PDF* from the Print dialog box (right-click the job you want, XPP displays a pop-up menu, select Print, XPP displays the Print dialog box).

PDF

To tell XPP whether to include the division in the PostScript file or PDF document, enter **yes** or **no** in the *PDF* column.

Entry	*Description*
yes	The utility adds this division to the PostScript file, in the order in which the division appears in the Document Assembly Ticket.
no	The utility does not include this division in the PostScript file. (default)

Comments XPP typically combines the output from multiple divisions in a job into a single PostScript file or PDF document. If the division types are *pdftoc*, *pdfidx*, or *citi*, run CITI before creating the PostScript file or PDF document.

Execute Distiller?

This field tells *psfmtdrv* whether to execute the PDF converter to create a PDF document from the resulting PDF-ready PostScript file.

This field gives you the choice of generating PDFs from the file as part of job processing or creating the PostScript disk file on one system and copying it to another system for PDF conversion.

Entry	*Description*
yes	Run Distiller or Ghostscript on the resulting PostScript file as part of job processing and uses parameters in the *_distill_parms* file or *gsdistill* and *_gsdistill_parms* files.
no	Do not run Distiller or Ghostscript on the resulting PostScript file, and do not use parameters in the *_distill_parms* file or *gsdistill* and *_gsdistill_parms* files. (default)

Resolve Xrefs across Job?

During composition, XPP creates cross-reference information from the *rx* (Mark Spot) XyMacros and from pickups and footnotes that are placed in the job.

To activate cross-references across all divisions in a job, set the *Resolve Xrefs across Job?* field to **yes** and set the *Compose* field to **yes**. If the *Resolve Xrefs across Job?* field is set to **yes** and you select *Compose* from the Job Processing menu. (Right-click the job you want. XPP displays a pop-up menu. Select Processes. XPP displays a sub-menu listing all the job processes you can perform at the job level.) XPP first composes all *active* divisions in the job, then it resolves all cross-references across the divisions by recomposing all lines with reference items.

Entry	*Description*
yes	Resolve all cross-references in divisions with an entry of **yes** in the *Compose* field.
no	Do not resolve cross-references across multiple divisions. (default)

Comments After you have composed all divisions, you can also resolve cross-references across divisions within a job by composing only the lines with reference items. Right-click the job you want. XPP displays a pop-up menu. Select Processes. XPP displays a sub-menu listing all the job processes you can perform at the job level. Click Compose. XPP displays the Compose dialog box. Select the Job Proc tab. Click the "Resolve xrefs across jobs" radio button.

XPP stores the cross-reference data in the *_job_xref* file at the job level that identifies the reference name and type, and the "relative" target page in the PDF document.

Note: *Reference items of the same type must have unique names across all divisions in a job. For example, you cannot have pickup "p1" in chap01 and pickup "p1" in chap05. However, different types of reference items can have the same name. For example, you can have a pickup named "monday," a footnote named "monday," and an* **《rx;monday》** *within the same job.*

Refer To Refer to the section "Cross-references in a Job" on page 6-6 for additional information.

CI Spec

You can use CITI to generate tables of contents and indexes with PDF XyMacros that will be converted to *pdfmarks* in the PostScript file. When the PostScript disk file is converted to a PDF document, the hypertext links and hot spots will be in place to help navigate readers through the document.

When CITI processes the job, it inserts the appropriate PDF XyMacro pairs around the extracted table of contents or index entry, after the *rsv* (Register Save XyMacro), and supplies the sequential division page number, division name, and subordinate level numbers. You do not have to insert PDF TOC and INDEX XyMacros in the XPP document or perform additional markup.

To use CITI to generate a PDF index or table of contents (or regular) division, edit the CI Spec for the job and fill in the following fields:

Citi Division Name

Specify the name of a PDF table of contents or index (or regular) division that CITI will generate.

You must create a table in the CI Spec with the same name as the PDF table of contents (*pdftoc*) or PDF index (*pdfidx*) or regular (*citi*) division specified in the Document Assembly Ticket.

PDF Level

Allows building the PDF table of contents and index levels independently of the levels for building the document.

Entry	*Description*
1 through 9	Creates a PDF entry with that level number
0	Does not use a differnt PDF level. Uses the *Level In Document* field. (default #)

Comments The PDF level numbers are important for creating the multi-level collapsible PDF outline form of the table of contents. The format of the table of contents in the PDF document is not affected by the PDF Level field.

You also can use the PDF Levels to insert optional data in the PDF XyMacros for both tables of contents and indexes.

The external PDF table of contents appears in the Overview window, on the left side of the PDF viewer main window when the Page Mode is set to *UseOutlines*.

Example For a table of contents, you assign the following PDF levels — book title (PDF Level 1), chapter title (PDF Level 2), Section headings (PDF Level 3), and Sub-section headings (PDF Level 4).

When the table of contents opens in the PDF reader navigation window, the PDF entries are indented as follows:

```
Book Title
  Chapter 1
    Section
      Sub-sectionA
      Sub-sectionA
  Chapter 2
    Section
```

Note: *Clicking on a down-pointing triangle to the left of an entry "hides" all its children and rotates the triangle to point to the entry.*

PDF Collapse

This field controls the initial display of the PDF table of contents (TOC) in the navigation window. *Collapsed* levels are ones that are defined in the TOC, but do not appear when you open a document for the first time.

For example, the previous example had only the book title and chapter titles displayed with the Sections and Sub-sections *collapsed*. When the TOC opens in the navigation window, the collapsed levels do not appear:

```
Book Title
  Chapter 1
  Chapter 2
```

The triangles pointing to the TOC entries means that each entry has one or more collapsed sublevels. You click on a triangle to display the sublevels.

To specify whether to collapse a level, enter **yes** or **no** in the PDF Collapse field:

Entry	*Description*
no	If there are entries subordinate to this one, continue the display. This is the default entry.
yes	Do not display entries subordinate to this one, that is, ″collapse″ the display at this entry.

Comments The default display in the navigation window is the entire hierarchy of a document’s TOC; all levels are set to **no** in the *PDF Collapse* field. A **yes** entry at any level ″collapses″ (hides the display of) any subordinate levels. You can collapse up to nine levels, but only those that correspond to a level in a particular document have an effect. For a document with three levels, only the collapse fields for *Level 1* and *Level 2* have an effect. Entries beyond *Level 2* are ignored.

If you use more than one CITI division to generate your document table of contents, set the *Collapse* fields to the same values for all of those divisions if you want the TOC’s initial display to be consistent. For example, if you generate your document table of contents with a separate PDFTOC division for chapter, figure, and table entries, and you want your TOC to display only *Level 1* entries initially, then you must go to the table for each of these divisions in your CITI Spec and set the *Level 1* field to **yes**.

Example For a table of contents, you assign the following PDF levels: chapter titles (PDF Level 1), main chapter headings (PDF Level 2), sections of main headings (PDF Level 3), and subsections of main headings (PDF Level 4).

You could produce the following variations of the initial TOC display using the indicated settings in the Collapse field:

Note: *Because Level 4 is the lowest level in the hierarchy, entries for Level 4-9 are ignored.*

Collapse Settings	*Result*
Level 1 - no **Level 2 - no** **Level 3 - no**	Displays the entire table of contents
Level 1 - yes **Level 2 - no** **Level 3 - no**	1. Displays only chapter titles 2. Click on a chapter title to see all subordinate levels
Level 1 - yes **Level 2 - yes** **Level 3 - no**	1. Displays only chapter titles 2. Click on a chapter title to see its main headings 3. Click on a main heading to see sections and subsections
Level 1 - yes **Level 2 - yes** **Level 3 - yes**	1. Displays only chapter titles 2. Click on a chapter title to display its main headings 3. Click on a main heading to display its sections 4. Click on a section to display its subsections

For additional information, refer to the TOC XyMacro in Chapter 3.

PDF Optional Data, Levels 1 through 9

Enter optional PDF data strings (for specific PDF levels) that are added to the generated <:pdfs;TOC> and <:pdfs;INDEX> XyMacros.

For example, you can use the optional data fields to specify different *rgb* color values for different levels.

Entry	***Description***
	A blank field means do not output optional data for this level. (default)
PDF data strings	User-definable optional data for this level.

Comments Optional data is data that is passed to the PostScript file for use by the PDF converter, such as color values.

The PDF Optional Data fields for the document are used by all the rules that have the PDF Level field with that number.

Example This example outputs PDF Level 1 entries with blue borders, PDF Level 2 entries with red borders, and PDF Level 3 entries with green borders.

```
PDF Optional Data
   Level 1     /Color [0 0 1]
   Level 2     /Color [1 0 0]
   Level 3     /Color [0 1 0]
```

Format of CITI-inserted PDF Macros

CITI outputs TOC and INDEX pass-though XyMacros in the *pdftoc* and *pdfidx* divisions, in the following formats, depending on the setting in the Division Ticket or Job Ticket:

XPP standard processing:

<:pdfs;TOC;+*page*,*count*,*div*;*opt-data*>[*insert*]*extracted-text*%**<:pdfe;TOC>**

<:pdfs;INDEX;+*page*,*div*;*opt-data*>{*insert*}*extracted-text*%**<:pdfe;INDEX>**

XML processing:

<?xpp :pdfs;TOC;+*page*,*count*,*div*;*opt-data?*>[*insert*]*extracted-text*%**<?xpp :pdfe;TOC?>**

<?xpp :pdfs;INDEX;+*page*,*div*;*opt-data?*>{*insert*}*extracted-text*%**?<?xpp :pdfe;INDEX?>**

SGML processing:

<?xpp :pdfs;TOC;+*page*,*count*,*div*;*opt-data*>[*insert*]*extracted-text*%**<?xpp :pdfe;TOC>**

<?xpp :pdfs;INDEX;+*page*,*div*;*opt-data*>{*insert*}*extracted-text*%**<?xpp :pdfe;INDEX>**

Table 5-1 *Format of CITI-inserted PDF Macros*

Entry	***Description***
:pdfs;TOC	Marks the start of the pass-through PDF XyMacro for a table of contents entry.
:pdfs;INDEX	Marks the start of the pass-through PDF XyMacro for an index (or in some cases table of contents) entry.
+page	The relative page number in an XPP main division for which the text was extracted. The **+** in front of a page number (+13) tells the PostScript formatter *psfmtdrv* that the page position is relative to a division and to calculate the sequence page number for the PDF document. For example, an entry appears on the fifth page of the *08chap* division, so the PDF INDEX XyMacro would be <:pdfs;INDEX;+5,08chap>*entry*<:pdfe;INDEX>. However, when XPP combines all divisions in the job (including CITI-generated *pdfidx* divisions) into a PDF-enhanced PostScript file, the entry would appear on sequence page number 128.

Table 5-1 *Format of CITI-inserted PDF Macros (Continued)*

Entry	*Description*
count —or— –count	For the PDF TOC XyMacro only, the count of the number of children (immediate subordinate entries) under this level entry. The number of immediate children (next level) collapsing all sublevels.
div	The name of the main division from which the text was extracted.
opt-data	Optional PDF data strings, if any, such as color values, from the *PDF Optional Data, Levels 1 through 9* field in the CI Spec.
[insert]	Entry from the Insert String field of the CI Spec. (optional)
extracted-text%	Text that was extracted for the table of contents or index entry appears between the start and end pass-through XyMacros. A pgraf mark (%) marks the end of the extracted text.
:pdfe;TOC	Marks the end of the pass-through PDF TOC XyMacro.
:pdfe;INDEX	Marks the end of the pass-through PDF INDEX XyMacro.
?	A question mark at the beginning ***and*** the end of both the pdfs and pdfe XyMacros indicates XML processing A question mark at the beginning of the pdfs and pdfe XyMacros, but ***not*** at the end of the XyMacro indicates SGML processing.
xpp	Indicates to the parser that this is an XPP processing instruction for an XML/SGML document.

Converting Pre-existing CITI Divisions to PDF

XPP can process files that exist at the job level to produce PDF documents automatically.

Pre-existing XPP divisions that use CITI to produce tables of contents and indexes do not need additional markup to produce PDF tables of contents and indexes.

The resulting PDF tables of contents and indexes are visually identical to pre-existing tables of contents and indexes divisions and there is no difference when output to a PostScript printer.

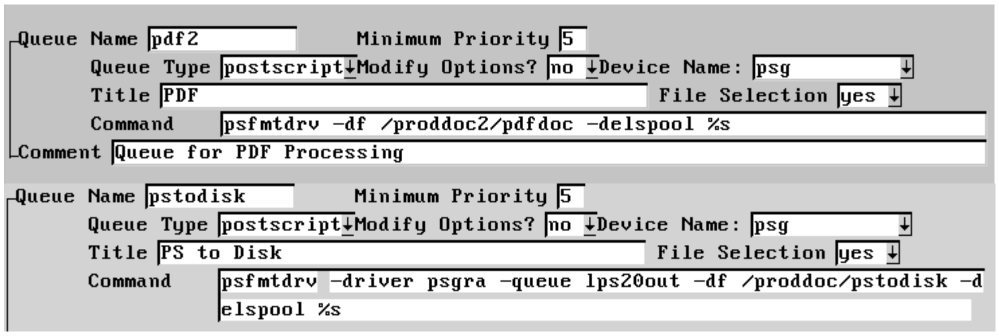
To convert pre-existing CITI divisions to PDF versions:

1. Change the Division Type in the Document Assembly Ticket from *citi* to *pdftoc* or *pdfidx*.
2. Edit the CI Spec to add PDF Levels and PDF Optional Data fields. (optional)
3. Recompose the divisions in the job.
4. Delete the existing CITI divisions.
5. Run *CITI* from the Job Processing menu. (Right-click the job you want. XPP displays a pop-up menu. Select Processes. XPP displays a sub-menu listing all the job processes you can perform at the job level.)
6. Run *PDF* from the Print dialog box.

Background Queue Spec

When creating background queues for printing, RWS recommends that you set up separate queues for PDF documents and for standard PostScript disk files. If you do not set up separate queues for PDF processing, standard PostScript disk file tasks can wait in the queue for PDF document creation.

Edit the Background Queue Spec and set up format and output queues for PDF processing, as shown in the following example:



```
Queue Name pdf2            Minimum Priority 5
      Queue Type postscript Modify Options? no  Device Name: psg
      Title PDF                                     File Selection yes
      Command    psfmtdrv -df /proddoc2/pdfdoc -delspool %s
Comment Queue for PDF Processing
Queue Name pstodisk        Minimum Priority 5
      Queue Type postscript Modify Options? no  Device Name: psg
      Title PS to Disk                              File Selection yes
      Command    psfmtdrv -driver psgra -queue lps20out -df /proddoc/pstodisk -d
                 elspool %s
```

Figure 5-3 *Queues for PDF and PostScript disk files*

Set *Modify Options* to **yes** to access the Device Options (DOF) file for the queue.

For information about accessing and editing the Background Queue Spec and the Device Options file, refer to the XPP document *XML Professional Publisher: Managing XPP*.

Setting Device Options

RWS recommends that you set device options carefully for the PDF document queue. Often the only device option switches you need to set are the *nhdr* (no header), *trctl* (tray control), and *ogb* (output graphic box) options.

To download fonts into the PostScript file, set the *efd* (enable font download) switch.

Table 5-2 describes device option switches that apply to PDF processing.

Table 5-2 *Device Option Switches for PDF Processing*

Switch	*Description*
pdfmark	This option tells the program to include markup needed by the PDF converter to produce a PDF file.

Table 5-2 *Device Option Switches for PDF Processing (Continued)*

Switch	***Description***
pdfps name	When creating a PostScript file, the program normally uses the PDF Support Spec listed in the Job Ticket. To use a different PDF Support Spec (overriding the spec in the Job Ticket), set the *pdfps* switch and name a PDF Support Spec to use.
pn name	Use the PostScript name switch to override the default naming convention of disk files. Either specify the name you want or use XPP variables to include the class, group, job, division, and pdfspec names. See Chapter 2 for information on output file names.

Related Information

Refer to these XPP documents:

- *XML Professional Publisher: Managing XPP*
- *XML Professional Publisher: Using CITI*

Chapter 6

PDF Job Processing

This chapter describes job processing options to create a PDF document, including:

- Composing divisions
- Running CITI
- Verifying PDF markup in XPP divisions
- Checking cross-reference items
- Converting an XPP job to a PostScript file
- Running the PDF converter

Job Processing

Creating a PDF document directly from an XPP job is a print process, that is, you print the XPP job directly into a PDF file by selecting the appropriate print parameters.

Before printing the PDF file, however, you must ensure that all other tasks are accomplished first, that is, all divisions are composed, the Document Assembly Ticket is prepared, etc. Printing directly to PDF is like any other print process for a job.

Tip! Refer to Chapter 13, ″Printing″ in the *XPP User Guide* for information on PDF print options.

To create PDF versions of XPP jobs:

1. In the PathFinder TreeView, right-click the job you want to output as a PDF file and select *Print* from the job pop-up menu.

   From the Print dialog box you may need to make selections on each of the following tabs: *Print, Output,* and *PS/PDF*.

2. **Print Tab**

   Click the *PDF file* or *PS to PDF file* box to tell XPP that it is creating an output file and not sending the file to the printer. Click the *PS to file* box if you want the print job output to be a PostScript file, which you can distill manually.

3. **Output**

   Set the PS output options that you want.

   Click the *Path/file name:* if you want to specify a particular file name and/or output destination for the file. The default is *jobname_pdfspecname*.ps output to the directory defined by the *XYV_TMPS* variable.

4. **PS/PDF**

   Set the PDF options that you want.

   Click the *Use PDF markup* field to produce the PDF attributes that you want in the output file.

   By default, the print process uses the PDF spec named in the Job Ticket. Click the *PDF Spec* field and enter the spec name you want if you don’t want the default.

   In the *PDF Security (Direct to PDF only)* frame, select the PDF security options to be applied to the output file.

   Refer to ″PDF Security Options (Direct-to-PDF only)″ on page 8-27 for details about the available PDF security options.

5. When you have completed all the tab selections you want, click the *Run* button at the bottom of the Print process dialog box.

XPP creates the output file you specified by your selections.

Sample Job Processing Sequence

Here is an example of a typical job processing sequence to create a PDF document. Access the job where you want to create the PDF output file and select the processes you need from the *Process* selection on the job pop-up menu:

1. Select *Process/Compose*.

   XPP looks in the *COMPOSE* field of the Document Assembly Ticket and composes all main divisions with an entry of **yes**.

   If the *Resolve Xrefs across Job?* field in the Document Assembly Ticket is set to **yes**, XPP recomposes reference items in active divisions within the job.

2. Run *CITI* to create PDF tables of contents and indexes.

3. Run *PDF Check* to verify that the PDF markup in all XPP divisions is correct.

   If the program reports errors, edit the XPP divisions to correct the errors.

4. Run *Show Xrefs* to ensure that cross-references are correct and divisions are fully composed.

   If the program reports errors, edit the XPP divisions to correct the errors.

5. Select *Print* and make the appropriate selections to produce the PDF output file.

Verifying PDF Markup

To verify PDF markup in a job, run the *showpdf* utility by selecting *PDF Check* from the Job Processing menu. (Right-click the job you want. XPP displays a pop-up menu. Select Processes. XPP displays a sub-menu listing all the processes you can perform from the job level.) Or, you can enter the *showpdf* command at the operating system command line.

The *showpdf* utility identifies many of the common errors that affect document distillation, such as missing start or end XyMacros or incorrect XyMacro syntax. If the program reports errors, edit the XPP divisions to correct the problems, then distill the document.

For detailed information about running *showpdf* from the operating system command line, see the *XML Professional Publisher: Command Line Utilities* manual.

Cross-references in a Job

During composition, XPP creates cross-reference information for:

- Locations marked in text with the *rx* (Mark Spot) XyMacro.
- Pickups and footnotes placed in the job.

XPP stores the cross-reference data in the *_job_xref* file at the job level with such information as the multi-part page number, the relative target page in a PDF document, and whether or not there are reference items in the job with duplicate names.

Unique Names for Reference Items

Reference items of the same type must have unique names across all divisions in a job. For example, you *cannot* have pickup "p1" in chap01 and pickup "p1" in chap05. However, different types of reference items can have the same name. For example, you can have a pickup named "monday," a footnote named "monday," and an **《rx;monday》** within the same job.

Activating Cross-References

Because divisions can contain references to locations in the same division or to locations in other divisions within the job, you must activate the cross-references across multiple divisions in a job.

Edit the Document Assembly Ticket and set the *Resolve Xrefs across Job?* field to **yes** and the *Compose* fields to **yes**.

- When you select *Compose* from the Job Processing menu, (Right-click the job you want. XPP displays a pop-up menu. Select Processes. XPP displays a sub-menu listing all the processes you can perform from the job level.) XPP first composes all *active* divisions, then it recomposes all lines with reference items to resolve cross-references across the divisions.
- You can also resolve cross references in the divisions of a job by composing only the lines with reference items, not whole divisions. Right-click the job you want. XPP displays a pop-up menu. Select processes. XPP displays a sub-menu listing all the processes you can perform from the job level. Select Compose. XPP displays the Compose dialog box. On the Compose tab, click the radio button in front of ″Cross references only″.

When using cross-references across multiple divisions in an XPP job that will be converted to a PDF document, you must list the target divisions (containing the reference) as *active* PDF divisions in the Document Assembly Ticket.

If a target division is missing from the PDF document, the Acrobat Distiller will fail with errors. For example, if you reference a pickup in Chapter 9, the PDF document must contain the Chapter 9 division.

Example This example shows an XPP job with cross-references across multiple divisions.

In Chapter 3, the *rx* XyMacro marks a section named "Precipitation Amounts."

```
{h1}Precipitation Amounts <rx;chap03.precipitation-amounts>
```

In Chapter 5, the *rf* XyMacro and *:pdfs;XREF* XyMacros refer to the ″Precipitation Amounts″ section in Chapter 3.

```
{text}Heavy rainfall is common in the springtime. Refer to the
section <:pdfs;XREF;chap03.precipitation-amounts> "Precipita-
tion Amounts" <:pdfe;XREF> <rf;chap03.precipitation-amounts>
on page <ri;%refpg3>-<ri;%refpg4>.
```

After using job processing to compose divisions and resolve cross-references, Chapter 5 contains this text:

> Heavy rainfall is common in the springtime. Refer to the section "Precipitation Amounts" on page 3-5.

The _job_xref File

You can activate references to a spot, footnote, or pickup that exists in another division within the job only by using job processing.

Every time a division is composed and saved, XPP updates the *__job_xref* file with the names of reference elements included in the division. When you reference an item that is not in the current division, XPP checks the *_job_xref* file to see if the reference was defined in a previously composed division in the job or if there is a duplicate name.

If the job contains duplicate reference items, you can edit the division(s) and change the item names before creating a disk file. You can archive/restore and copy the *_job_xref* file with a job.

Note: *Reference items of the same type must have unique names across all divisions in a job. For example, you cannot have pickup "p1" in chap01 and pickup "p1" in chap05. However, different types of reference items can have the same name. For example, you can have a pickup named "monday," a footnote named "monday," and an* **《rx;monday》** *within the same job.*

When is the _job_xref File Updated?

XML Professional Publisher:

- Creates the *_job_xref* file at the job level the first time a full composition is performed on any division in the job.
- Updates the *_job_xref* file whenever a full or partial composition of a division occurs.

What information is in the _job_xref file?

The *_job_xref* file contains:

- The names of each division in the job, the number of pages in the division, and the composition state (none, full, partial).
- Total number of references in the job.
- Reference names, in alphabetical order, for footnotes, pickups, and spots marked with the *rx* XyMacro.
- Information on whether there are duplicate reference names in the job.
- Division where the reference is defined.
- Relative page number of the reference.
- Six-part page number of the reference.
- Values of the 256 number registers.

Use Show_Xrefs to Check Cross-references

To view the contents of the *_job_xref* file:

1. Right-click the **job** you want.

   XPP displays a pop-up menu.

2. Select **Processes** from the pop-up menu.

   XPP displays a submenu listing all the processes you can perform from the job level.

3. Select **Show_Xrefs** from the submenu.

   XPP displays the *Show_Xrefs* dialog box.

4. Check the Command Line Options check box if you want to enter commands unique to the job.

5. Check the Preview Command check box to see the commands (optional).

   When you click the Run button, XPP displays a message box listing the commands to be included and presents the option to continue running the process.

6. Select **yes** to run the process.

   —or—

   Select **no** to abort the process.

For detailed information on running *show_xref* from the command line, see the *XPP Command Line Utilities* manual.

If the program reports errors, such as reference items with duplicate names, edit the division and correct the errors, then recompose the job.

Related Information

For additional information, refer to these XPP documents:

- *XML Professional Publisher: Managing XPP*
- *XML Professional Publisher: Using CITI*

Chapter 7

Special PDF Features: Postscript Format Driver (psfmtdrv)

This chapter describes how to use some of the special features in PDF, namely:

- Using psfmtdrv PDF Accessibility Features
- Defining Page Regions

Using psfmtdrv PDF Accessibility Features

PDF accessibility features provide people with disabilities, such as vision, hearing, or motor impairments with the ability to access electronic information. The Postscript Format Driver (psfmtdrv) program includes PDF accessibility features to help provide such users access to the content of your PDF output.

Print Option for Word-Space Requirement

You can insert most of the tagged PDF commands for PDF Accessibility features using the PDF passthru macro, ◀:pdfs;GEN▶. With this macro, you mark the start and end of paragraphs, sections, and so forth.

When XPP generates PostScript output, it uses a typographic coding style. That is, it calculates word spaces and explicitly places the text. However, PDF accessibility tools require that space characters be present to separate words.

Software tools that read Tagged PDF require these spaces to locate word breaks. The *psfmtdrv* utility has a **–pdfacc** option to fulfill this requirement.

When you use this option, and the utility encounters a space or computer-generated space, it generates a zero-length space character in addition to a horizontal move.

Defining Page Regions

XPP generates structured comments in the PostScript output to define page regions. You can also use options for the *psfmtdrv* utility to set the PDF CropBox and BleedBox.

Structured Comments

This section describes the structured comments as well as related options for the *psfmtdrv* utility.

The PostScript output contains the following structured comments relating to page regions:

- %%BoundingBox (per file)
- %%HiResBoundingBox (per file)
- %%CropBox (per file)
- %%PageBoundingBox (per page)
- %%PageHiResboundingBox (per page)
- %%PageCropBox (per page)

If you add **–pdfmark** as an additional option to the *psfmtdrv* utility, the PostScript output contains the following pdfmark: /TrimBox (per page).

If you add **–bleed** to **–pdfmark** as an additional option, the PostScript output contains a /BleedBox pdfmark per page. **–bleed** takes a numeric argument followed by any Xyvision qualified units (defaults to points). This option specifies the offset of the BleedBox from the TrimBox.

PDF CropBox

The PDF CropBox values are set to the page (or frame) bounding box rather than to the media box, when appropriate.

If you are outputting text or marks outside the page bounding box (without using frame frills) or outside the frame bounding box, and you do not want these marks to be cropped, you can add **–mediacrop** as an option to the *psfmtdrv* utility. This option forces the PDF CropBox to be set to the media box.

PDF TrimBox

The PDF TrimBox values are set to the page layout rather than to the media box, when appropriate.

If you want to produce PostScript for PDFs where the TrimBox is determined by the the page control file, you can use the **–mediatrim** option with the *psfmtdrv* utility.

This option forces the PDF TrimBox to be set to the media box.

Using this option is similar to using the **–mediacrop** option to set the Boundingbox to the media box dimensions instead of the page layout dimensions.

Chapter 8

Special PDF Features: Direct-to-PDF (divpdf)

This chapter describes how to use some of the special features in PDF, namely:

- XPP PDF Tagging Features
- PDF/UA Tagging Tips and Tricks
- Support for 3D PDF images
- PDF security options for Direct-to-PDF (divpdf) output

XPP PDF Tagging Features

You can use the PDF tagging features of the Direct to PDF (divpdf) program to help provide people with disabilities, such as vision, hearing, or motor impairments the ability to access electronic information.

CSS is used to tag the textual content and generate bookmarks/annotations (links). Output options allow for creating a fully PDF/UA-compliant document or using CSS to generate bookmarks and annotations only, without having to tag all the content.

These properties work with both CSS-XML and XML Divisions, provided that the XML is well-formed and the markup is based on the logical reading order of the document.

Tagged PDF for PDF/UA

PDF tags provide a hidden structured, textual representation of the PDF content that is presented to screen readers.They exist for accessibility purposes only and have no visible effect on the PDF file. For PDFs to be PDF/UA compliant, they need to be tagged based on the logical reading order (WCAG 2.0) of the document.

Tagged PDFs allow various assistive technologies (AT), such as screen readers, to interact with the content of a document and make sense of it. Tagged PDFs also allow for document reflow, as well as allowing for export/conversion from PDF to other formats such as HTML.

Bookmarks and Annotations

XPP includes XPP-specific CSS properties to control the creation of PDF tags, bookmarks, and annotations by the Direct to PDF program divpdf.

The `-pdfba` divpdf option can be used to avoid having to use PDF marks to generate bookmarks and annotations. For more information about using the `-pdfba` option, see ”Command Line Options” on page 8-4.

Document Information Properties

Document information metadata can also be set with CSS properties. For example, the values representing the title and author may be referenced from the XML to set the corresponding PDF properties. This functionality is available for both PDF/UA and for creating bookmarks and annotations in non-PDF/UA documents.

PDF CSS Properties

XPP CSS properties fall into the following categories:

- PDF Tagging properties: `-xpp-pdf-tag` and `-xpp-pdf-tag-options` for PDF/UA only
- PDF Bookmark properties: `-xpp-pdf-bookmark` and `-xpp-pdf-bookmark-options`
- PDF Annotation/Link properties: `-xpp-pdf-annotation` and `-xpp-pdf-annotation-options`
- PDF Document Information property: `-xpp-pdf-docinfo-options`

Example *PDF/UA Tagging*

The following is an example of CSS PDF/UA tagging for simple XML document using an XPP style bundle named 'mystyle':

```
<doc>

    ...

    <title>My Title</title>

    ...

    <p>Paragraph text</p>

    ...

</doc>
```

You can create a `mystyle.css` CSS file at the Job level that has the following CSS properties:

```
doc { -xpp-pdf-tag: ″Part″; }

doc > title { -xpp-pdf-tag: ″H1″; }

p { -xpp-pdf-tag: ″P″; }
```

After running `divpdf -pdfua` against your XML Division, the result would be a tagged PDF file that conforms to PDF/UA, without having to write macros or re-compose the division.

Example *CSS Bookmark Tagging*

The following is an example of CSS Bookmark tagging:

XML Source:

```
<h1>Proposal for PDF Bookmarks and Annotations
(divpdf) in XPP 9.2</h1>
<h2>Overview</h2>
```

CSS:

```
h1 {
  -xpp-pdf-bookmark: ″type={book} level={1}″;
  -xpp-pdf-bookmark-options: ″textcolor={rgb 1 0 0}″;
  /* Create Red Bookmark */
}
h2 {
  -xpp-pdf-bookmark: ″type={book} level={2}″;
}
```

After running `divpdf -pdfba` against your XML Division, the result would be a PDF file that contains bookmarks created from CSS rules, without having to write macros or re-compose the division, and without having to generate a PDF/UA tagged document.

Command Line Options

The divpdf command has options to create tagged PDF and/or bookmarks and annotations as follows:

```
divpdf [ -pdfba ] [ -pdfua ] [ -pdfcss name ] [-
pdfmark ] [-pdfuaimg type ] [ -pdfuapng # ] [ -
pdfuatif # ] [ -strong ] [ -lang code ]
```

where:

- **-pdfba** = Generate bookmarks and annotations only using CSS
- **-pdfua** = Generate a PDF/UA compliant document (including bookmarks and annotations) using CSS
- **-pdfcss** ***name*** = Use the named CSS file instead of the style name specified in Job Ticket and/or Division Ticket.
- **-pdfmark** = Honor the PDF Pass-through macros.

The following options are for PDF/UA only:

- **-pdfuaimg *type*** = Specify image type (defaults to png16m) for raster conversion of EPS|CGM|non-PDF/UA-1-compatible PDF images. There are PNG and TIFF options; for the full list of types, refer to *XML Professional Publisher: Command Line Utilities*. Note that the (default) PNG type will result in any CMYK colors being converted to RGB.
- -**pdfuapng #** = Specify resolution (d.p.i: defaults to 600) for PNG conversion of EPS|CGM|non-PDF/UA-1-compatible PDF images.
- -**pdfuatif #** = Specify resolution (d.p.i: defaults to 600) for TIFF conversion of EPS|CGM|non-PDF/UA-1-compatible PDF images.
- **-strong** = Declare document as strongly structured (allow only "H" tags for headings).
- **-lang *code*** = Specify document language, such as `-lang en` or `-lang es-MX`. If no language is specified, the default document language is the first language in the Job Ticket.

With `-pdfua`, CSS properties for PDF tags, Bookmarks/Annotations, and Document information are processed (and properties for PDF tags are required). With `-pdfba`, only Bookmarks/Annotations and Document information CSS properties are processed (and properties for PDF tags are not required).

The `-pdfmark` option allows Annotations and Bookmarks coded using `<:pdfs>` and `<:pdfe>` macros to be used in conjunction with the CSS properties. However, it is recommended that for future use for PDF/UA PDF documents, they be implemented using the Annotation and Bookmark CSS properties (see Bookmark and Annotation/Link section).

The PDF pass-through Macros may continue to be used by divpdf to produce non-PDF/UA PDF documents and PDF documents for SGML and Classic XPP divisions.

However, the `<:pdfs:GEN>` and `<:pdfe:GEN>` macros are not supported because they generate postscript. The primary use of these macros was to implement a PDF tag structure, and they have therefore been superseded by the XPP PDF tagging properties.

Note: *By default, divpdf uses the style bundle for the name and location of the CSS spec. If the division is in CSS mode, the appropriate CSS properties can be added to the current CSS file. If the division is in XML mode, a CSS file can be created at the job or library level with the same name as the style bundle. In either case, a CSS spec with a different name can be used by overriding the active CSS spec by using the* `-pdfcss <name>` *option.*

Important: *Examples in the following sections illustrate some of the XPP functionality supported by PDFlib. For more comprehensive inforation about the PDFlib library, consult the latest PDFlib API Reference.*

Option Lists

In each category there is an `options` property that is a CSS string value. The syntax of the contents of each `options` property is defined by the PDFlib software library, which is used by divpdf to create tags, bookmarks, annotations, and other objects in the PDF.

Option lists are strings that can contain an arbitrary number of options. In most cases, an option list will contain one or more key=value elements.

If there are multiple elements, they should be separated by spaces. If a value itself contains spaces, surround the value with curly braces.

An outermost pair of enclosing braces is not part of the element. The sequence `{ }` designates an empty element. If an element contains brace characters, these must be protected with a preceding backslash (`\`) character.

The following are examples of option lists:

- `-xpp-pdf-tag-options: ″ListNumbering=Decimal″;`
- `-xpp-pdf-bookmark-options: ″textcolor={rgb 1 0 0}″;`
- `-xpp-pdf-annotation: ″type={www} text={attr(href)}″;`
- `-xpp-pdf-docinfo-options: ″Title={Testing \2014 -xpp-pdf-annotation et al.}″;`

Note: *Unicode characters can be entered using utf-8, or as* `\xxxx` *hex unicode values. In this case,* `\2014` *inserts an em-dash into the* `Title` *value.*

Consult Section 1 of the latest *PDFlib API Reference* for more detail about the syntax for option lists used by PDFlib function calls.

CSS Properties in Detail

XPP supports the following CSS properties:

- `-xpp-pdf-tag: none | <`*`string`*`>`
- `-xpp-pdf-tag-options: none | <`*`string`*`>`
- `-xpp-pdf-bookmark: none | <`*`string`*`>`
- `-xpp-pdf-bookmark-options: none | <`*`string`*`>`
- `-xpp-pdf-annotation: none | <`*`string`*`>`
- `-xpp-pdf-annotation-options: none | <`*`string`*`>`
- `-xpp-pdf-docinfo-options: none | <`*`string`*`>`

Note that in each of these properties, you can insert the `attr(`*`attribute_name`*`)` function to insert the value of the specified attribute from the corresponding XML element. The examples later in this chapter illustrate the use of the `attr()` function.

XPP PDF Tagging Properties (PDF/UA Only)

The `-xpp-pdf-tag` and `-xpp-pdf-tag-options` CSS properties create PDF tags in the tag tree (logical structure tree). The following is the syntax for the `-xpp-pdf-tag` option:

`-xpp-pdf-tag: none | <`*`string`*`>`

where: <*string*> is the PDF tag name.

For a complete list of tag options supported by PDFLib, consult Table 14.1 in the *PDFlib API Reference*. The following table shows some of the tag name options commonly used with XPP:

Table 8-1 *Tag Name Options Commonly Used with XPP*

Category	*Tag Names (case-sensitive)*
`group`	`Document`, `Part`, `Art`*, `Sect`, `Div`, `BlockQuote`*, `Caption`, `TOC`*, `TOCI`*, `Index`*, `NonStruct`*, `Private`*.
`head_para`	`P`, `H`, `H1`, `H2`, `H3`, `H4`, `H5`, `H6`, `H7`, `H8`...
`list`	`L`, `LI`, `Lbl`, `LBody`
`table`	`Table`, `TR`, `TH`, `THead`, `TBody`, `TFoot`.
`inline`	`Span`, `Quote`*, `Note`***, `Reference`, `BibEntry`*, `Code`*, `Link`, `Annot`.
`illustration`	`Figure`, `Formula`.
`Japanese`	`Ruby (grouping)`, `RB`, `RT`, `RP`, `Warichu (grouping)`, `WT`, `WP`.
`user`	Not supported at this time.
`pseudo`	`Artifact`, `ASpan`, `ReversedChars` (not recommended), `Clip` (not recommended).

* Deprecated in PDF 2.0/ISO 32000-2.

*** The `Note` tag is used for footnotes.

The following is the syntax for the `-xpp-pdf-tag-options` option:

`-xpp-pdf-tag-options: none | <`*`string`*`>`

where: <*string*> is an *oplist* to bes passed to the `PDF_begin_item()` function.

For a complete list of tag options supported by PDFLib, consult Tables 14.2 and 4.3 in the *PDFlib API Reference*.

The following table shows some of the PDF tag options commonly used with XPP:

Table 8-2 *PDF Tag Options Commonly Used with XPP*

Option	***Description***
`ActualText`	Equivalent replacement text for the content item and its children.
`Alt`	Word or phrase as alternate description for the content item and its children. It should be provided for figures, images, etc. that cannot be recognized as text.

`direct`	Boolean; only for tag names `Artifact`, `Code`, `FENote`, `BibEntry, Em, Note, Quote, Reference, Span, Strong, Sub; for Lbl as child of BibEntry, TOCI or Note; and for the pseudo items ASpan, ReversedChars, Clip.`
	`If true, the content item is written inline and no structure element is created.`
	`Default: true.`

`Lang`	String; not for pseudo tags except `ASpan`. Two- or three-character language code according to ISO 639-1/639-2, optionally followed by a hyphen and a two-character ISO 3166 region code (e.g. `en-us`, `en-gb`, `es-mx`) to override the document's default language/country codes for a content item and its associated options `ActualText`, `Alt`, and `E`. Not case-sensitive.

Example *PDF CSS Tagging*

The following is an example of PDF CSS tagging:

XML Source:

```
<p lang="de">Mein Hut, der hat drei Ecken</p>
```

CSS:

```
p[lang] {
    -xpp-pdf-tag: "P";
    -xpp-pdf-tag-options: "Lang={attr(lang)}";
}
```

XPP PDF Bookmark Properties

The XPP-specific `-xpp-pdf-bookmark` and `-xpp-pdf-bookmark-options` CSS properties will create a PDF Bookmark. The following are descriptions of these properties.

The following is the syntax of the `-xpp-pdf-bookmark` option:

`-xpp-pdf-bookmark: none | <string>`

where: <*string*> contains "<*type*> <*text*> <*page*> <*level*> <*state*> <*target*>"

The allowed values for each property are shown in the following table:

Table 8-3 *Property Values for -xpp-pdf-bookmark*

Property	*Allowed Value*	*Default*
type	`book | toc`	none
text	`none | <`*string*`> | content() | attr()`	`content()`
page	`current | <`*integer*`> | -xpp-page(`*+rel-page#,div-name*`) | attr()`	`current`
level	`none | <`*integer*`> | attr()`	none
state	`open | closed | attr()`	`closed`
target	`self | url(`*target-url*`) | dest(`*named destination*`) | attr()`	`self`

The following is the syntax of the `-xpp-pdf-bookmark-options` option:

`-xpp-pdf-bookmark-options: none | <`*`string`*`>`

where:

<*`string`*> contains options which are passed to PDF_create_bookmark(), as defined in Table 12.1 in the *PDFlib API Reference*.

It is possible to use the attr() function within the *`string`* to insert the value of an attribute.

The more commonly used bookmark options are shown in the following table:

Table 8-4 *Common Bookmark Options*

Option	***Description***
fontstyle	(Keyword) Specifies the font style of the bookmark text: normal, bold, italic, bolditalic. Default: normal
textcolor	(Color) Specifies the color of the bookmark text. Supported color spaces: none, gray, rgb. Default: rgb {0 0 0} (=black)

Example *XPP PDF Bookmark Tagging*

The following is an example PDF bookmark tagging:

XML Source:

```
<h1>Proposal for PDF/UA support (by divpdf) in XPP
9.2</h1>
<h2>Overview</h2>
```

CSS:

```
h1 { -xpp-pdf-tag: "H1"; /* PDF/UA Only */
   -xpp-pdf-bookmark: "type={book} level={1}";
   -xpp-pdf-bookmark-options: "textcolor={rgb 1 0 0}";
   /* Create Red Bookmark */
}
h2 { -xpp-pdf-tag: "H2"; /* PDF/UA Only */
   -xpp-pdf-bookmark: "type={book} level={2}";
}
```

Note: *The "level" value controls how bookmarks are nested within each other. In this case, an h2 bookmark that follows an h1 bookmark will be created as a "child" of the h1 bookmark. This means that the user does not have to specify the # of children of any bookmarks, in contrast to the PDF pass-through method.*

XPP PDF Annotation/Link Properties

The XPP-specific `-xpp-pdf-annotation` and `-xpp-pdf-annotation-options` CSS properties will create a PDF Annotation.

`-xpp-pdf-annotation: none | <string>`

where: *`<string>`* contains "<*`type`*> <*`text`*> <*`file`*> <*`page`*>"

and where the allowed values for each property are as shown in the following table:

Table 8-5 *Property Values for -xpp-pdf-annotation*

Type	*Text*	*File*	*Page*	*Comment*
dest	*<named dest>*	N/A	N/A	define a named destination
2dest	*<named dest>*	(opt)	N/A	create link to named dest (in named file)
2page	*<page #>*	(opt)	(opt)	create link to page # (in named file/div)

Table 8-5 *Property Values for -xpp-pdf-annotation (Continued)*

Type	*Text*	*File*	*Page*	*Comment*
note	`<note text>`	N/A	N/A	create a PDF note
launch	`<program name>`	N/A	N/A	create link to launch a file/ application
www	`<URL>`	N/A	N/A	create link to a location in the WWW
xref	`<xref name>`	N/A	N/A	create link to (xref) marked spot/foot/pick
index	`<page #>`	N/A	(opt)	create link to page # (in named div)
toc	`<page #>`	N/A	(opt)	create link to page # (in named div)

-xpp-pdf-annotation-options: none | <*string*>

where: <*string*> contains options that are passed to the PDF_create_annotation() function, as defined in Table 12.3 in the *PDFlib API Reference*.

The more commonly used annotation options are shown in the following table:

Table 8-6 *Common Annotation Options for -xpp-pdf-annotation-options*

Option	*Description*
annotcolor	(Color) The color of the background of the annotation's icon when closed, the title bar of the annotation's pop-up window, and the border of a link annotation. Supported color spaces: none, gray, rgb, and (in PDF 1.6) cmyk. Default: none
borderstyle	(Keyword) Style of the annotation border: solid, beveled, dashed, inset, or underline. Note that the beveled, inset, and underline styles do not work reliably in Acrobat. Default: solid
contents	Text to be displayed for the annotation or (if the annotation does not display text) an alternate description of its contents in human-readable form.
linewidth	(Integer) Width of the annotation border in default units (=points). If linewidth=0 the border will be invisible. Default: 1

Table 8-6 *Common Annotation Options for -xpp-pdf-annotation-options (Continued)*

Option	*Description*
open	(Boolean; only for type=Text, Popup); If true, the annotation will initially be open. Default: false

Example *XPP PDF Annotation/Link Tagging*

The following is an example PDF annotation/link tagging:

XML Source:

```
<a href="https://www.section508.gov/content/
accessibility">Section 508</a> is a US Federal Law.
```

CSS:

```
a[href] { -xpp-pdf-tag: "Link"; /* PDF/UA Only */
    -xp-pdf-annotation: "type={www}
    text={attr(href)}";
    -xpp-pdf-annotation-options: "contents={jump to
    WWW}
    linewidth={1}";
}
```

XPP PDF Document Information Property

The following CSS property will update the PDF Document Information metadata using the PDFLib set_info(key,value) API call, as specified in the *PDFlib API Reference:*

-xpp-pdf-docinfo-options: none | <*string*>

where: <*string*> consists of name={value} pairs.

Example *XPP PDF Document Information Tagging*

The following is an example of document information tagging:

XML source:

```
<doc title="PDF/UA support (Section 508 Compliance) in
XPP 9.2" author="A. Wilmot and S. Piercey"
subject="Section 508 support in XPP". . .
```

CSS:

```
doc {
  -xpp-pdf-tag: "Part"; /* PDF/UA Only */
  -xpp-pdf-docinfo-options: "Title={attr(title)}
  Author={attr(author)} Subject={attr(subject)}";
}
```

Keys and descriptions for document information fields are shown in the following table:

Table 8-7 *Document Information Keys and Descriptions*

Key	***Description***
Subject	Subject of the document
Title	Title of the document
Creator	Software used to create the document. Acrobat will display this entry as »Application«. This key is distinct from the Producer key, which is always set to a string that indicates PDFlib produced the PDF output.
Author	Author of the document
Keywords	Keywords describing the contents of the document
Trapped	Indicates whether trapping has been applied to the document. Allowed values are: True, False, and Unknown.
any other name	User-defined document information field. PDFlib supports an arbitrary number of fields. A custom field name should only be supplied once. Custom document info fields must not contain any of the following characters if XMP metadata is created (via the autoxmp or metadata options): & \ < >" space Fields which are used for standard identification are not allowed.

The XPP User Interface (Pathfinder)

The XPP Pathfinder Print interface supports the PDF/UA and Bookmarks/ Annotation Only options. When the PDF File option is selected on the Print tab, the following choices are enabled on the “PS/PDF” Tab:

- **Generate PDF/UA:** Enable PDF/UA output by reading the CSS style file.
- **Generate Bookmarks/Annotations Only:** Generate annotated output by reading the CSS file.

Note: *You can select only one (or neither) of these options.*

Note: *The Generate PDF/UA option automatically enables “Download Fonts” since the PDF/UA spec requires that the fonts be embedded in the PDF. The “Disable Font subsetting” option will include the whole font, rather than including only characters used from the font.*

- **Use PDF markup:** Allows the PDF pass-through macros `<:pdfs>` and `<:pdfe>` to be used.
- **CSS spec override:** Override the spec name for the CSS spec (defaults to the style name)
- **Strongly structured** (PDF/UA only)**:** The document is marked as ″strongly-structured″, which allows only the ″H″ PDF tag to be used. H1, H2, etc are considered illegal.
- **Document language** (PDF/UA only)**:** The document language e.g. ″en″ or or ″es-MX″ as the ISO language/country code.

PDF/UA Tagging Tips and Tricks

Overview

When debugging issues with PDF tagging, you may encounter PDFlib exceptions in cases where, for example, an assigned PDF tag is not valid in the context in which it is assigned. In such cases, it can be useful to execute divpdf with the `-dump` option, which creates a debugging output file named `divpdf.log` in the same directory where the PDF is created.

Figure Tagging

An image/graphic should be tagged using the ″Figure″ PDF tag. The exception is when the graphic does not represent meaningful content, in which case the graphic should be tagged as ″Artifact″. According to PDF/UA, the user must provide an alternative representation or replacement text that represents the contents marked with the ″Figure″ tag; this should be done using the ″Alt″ option as in the following example:

Example *Figure Tagging*

XML:

```
<figure id=″my_id″ alt=″my alternate text″> . . .
</figure>
```

CSS:

```
figure {
    -xpp-pdf-tag: ″Figure″;
    -xpp-pdf-tag-options: ″Alt={attr(alt)}″;
}
```

If there is a caption with the figure, it should be tagged as ″Caption″.

Automatic Figure Tagging

If an image is neither tagged as an ″Artifact″ nor as a ″Figure″, divpdf will automatically tag the image as a ″Figure″.

In such a case, if there is a caption associated with the image, you can use the `-xpp-alt={figure alternate text}` in the caption's `-xpp-pdf-tag-options` CSS property to specify the alternate text for the figure as in the following example:

Example *Using the Caption to Set Alternate Text*

The following is an example of using the caption attribute to set alternative text for a figure:

XML:

```
<figure id="foo"> . . . <caption figalt="figure alter
nate text">. . .
```

CSS:

```
caption {
    -xpp-pdf-tag: "Caption";
    -xpp-pdf-tag-options: "-xpp-alt={attr(figalt)}";
}
```

Example *Caption Source XML Elements with Text Content*

Sometimes, the source XML element for the figure caption contains direct text. In this case, assigning the "Caption" PDF tag will result in a PDFlib exception, since "Caption" is a grouping element and cannot contain text directly as in the following example:

XML:

```
<figure . . . > . . . <caption . . .>This is the
caption text.</caption> . . .</figure>
```

In this case, the following is a way of creating a parent plus a child PDF tag from a single source XML element:

CSS:

```
caption { -xpp-pdf-tag: "Caption+P"; }
```

This CSS causes divpdf to create both a "Caption" and a (child) "P" PDF tag from the 'caption' source XML element.

Headings Tagging

A conforming PDF/UA PDF document shall use heading tags for all headings. PDF/UA allows for either a weakly- or strongly-structured document. Divpdf sets the PDF document by default to ″weakly-structured″. With this setting, H1, H2, etc. can be used as follows:

- If any heading tags are used, H1 shall be the first.
- A document may use more than one instance of any specific tag level. A tag level may be repeated if document content requires it. For example, H1 H2 H3 H3 is a valid sequence if the content has one top-level heading, one second-level heading, and two consecutive third-level headings.
- If document semantics require a descending sequence of headers, such a sequence shall proceed in strict numerical order and shall not skip an intervening heading level. That is, H1 H2 H3 is permissible, while H1 H3 is not.
- A document may increment its heading sequence without restarting at H1 if document semantics require it. For example, H1 H2 H3 H4 H3 H4 H3 H4 H2 H3 is a permissible sequence.

Strongly-structured PDF Documents

The `-strong` command line option in divpdf declares that the document is ″strongly-structured″, which allows for only the H PDF tag to be used. When the `-strong` command line option is used, it is invalid to have any H1..H*n* PDF tags.

Table Tagging

Tables should always include headers, and can contain column headers, row headers or both.

Table elements should be tagged as in the following table:

Table 8-8 *Table Tagging Structures*

Structure Type	*Description*
Table	A two-dimensional layout of rectangular data cells. It contains either one or more table rows (structure type TR) as children; or an optional table head (structure type THead) followed by one or more table body elements (structure type TBody) and an optional table footer (structure type TFoot). In addition, a table may have a caption (structure type Caption) as its first or last child.

Table 8-8 *Table Tagging Structures (Continued)*

Structure Type	*Description*
TR	(Table row) A row of headings or data in a table. It may contain table header cells and table data cells (structure types TH and TD).
TH	(Table header cell) A table cell containing header text describing one or more rows or columns of the table.
TD	(Table data cell) A table cell containing data that is part of the table's content.
THead	(Table header row group) A group of rows that constitute the header of a table.
TBody	(Table body row group) A group of rows that constitute the main body portion of a table.
TFoot	(Table footer row group) A group of rows that constitute the footer of a table.

As much information as possible about the structure of tables needs to be available when assistive technology is relied upon. Headers play a key role in providing structural information.

Example

Elements with a Scope Attribute

Structure elements of type TH have a Scope attribute, so for example, your CSS may look like the following for an element tagged as a TH:

```
table tbody tr th {
    -xpp-pdf-tag: ″TH″;
    -xpp-pdf-tag-options: ″Scope=Column″;
}
```

It should be noted that it is invalid in PDF/UA to have a Table that contains a TBody but no THead.

List Tagging

According to the PDF/UA spec, a list should be tagged using ′L′, which must conform to the following rules:

- A list must consist of sequence of items of like meaning and importance.
- Its immediate children should be an optional caption (structure type ″Caption″; a "Grouping Element") followed by one or more list items (structure type ″LI″).

- An explicit `ListNumbering` attribute shall be used for L tags: `Disc` | `Circle` | `Square` | `Decimal` | `UpperRoman` | `LowerRoman` | `UpperAlpha` | `LowerAlpha`.
- Individual list items shall be specified by LI tags. LI (List item) is an individual member of a list. Its children may be one or more labels, list bodies, or both (structure types ″Lbl″ or ″LBody″).

Problems Tagging Lists

If in the XML source there is a list with XML tags which correspond to each list item, label and list body, there is no problem tagging such a list using the ″LI″, ″Lbl″ and ″LBody″ tags for PDF/UA compliance.

However, sometimes list items in XML are marked up using a single ″list item″ tag that contains the list body text as direct content, and the label is ′generated content′ as in the following examples:

Example 1 *List Item Tagging with Direct Text Content*

The following is an example of a list item tag containing direct text content.

Source XML:

```
<ol>
    <li>This is the first list item.</li>
etc.
```

In this case, if the `<li>` XML tag is assigned a ″LI″ PDF tag, this is not valid PDF/UA, since the ″LI″ tag must have either a ″Lbl″ or an ″LBody″ as a child. Direct text (or a generated label) is not permitted.

Another common XML markup strategy for lists is to have a list item tag which has a single ′para′ child. A label, if present, is generated between the list item start tag and the child start tag as in the following example:

Example 2 *List Item Tagging with a Single Child Element*

The list item tag always has a single child element, as in the following example:

Source XML:

```
<list type=″numbered″>
    <list-item><p>This is the first list
    item.<p></list-item>
etc.
```

In this case, if the `<list-item>` XML tag is assigned a ″LI″ pdf tag, this is also not valid PDF/UA. Even if the `<p>` is tagged as a ′LBody′, the (XPP-generated) label will cause a PDFLib exception because the label text is not valid as a direct child of the ′LI′ tag.

Automatic List Tagging

To handle these cases, divpdf supports an option called `-xpp-auto`, which takes a value of `1` or `2`. Example 1 includes a numbered list where each list item source XML tag has direct text content. Thus your CSS should be:

```
ol {
    -xpp-pdf-tag: ″L″;
    -xpp-pdf-tag-options: ″ListNumbering=Decimal″;
}
li {
    -xpp-pdf-tag: ″LI″;
    -xpp-pdf-tag-options: ″-xpp-auto={1}″;
}
```

In Example 2, there is a numbered list with each list item source XML tag having a single child element. Your CSS should be:

```
list[type=″numbered″]{
    -xpp-pdf-tag: ″L″;
    -xpp-pdf-tag-options: ″ListNumbering=Decimal″;
}
list-item{
    -xpp-pdf-tag: ″LI″;
    -xpp-pdf-tag-options: ″-xpp-auto={2}″;
}
```

In Example 1, use of the `-xpp-auto={1}` option causes divpdf to search for a computer-generated label following the list item tag. If a label is found, it is wrapped with a ′Lbl′ PDF tag. Divpdf then automatically generates an ′LBody′ pdf tag to contain the remainder of the list item contents.

In Example 1, if this is an unlabelled list, you should also use `-xpp-auto={1}` for the list item tag in the CSS, but use `ListNumbering=None` for the tag-options on the parent (″L″) element.

In Example 2, using the `-xpp-auto={2}` option automatically generates a ′Lbl′ pdf tag as a container for the list item label, and an ′LBody′ pdf tag around the list item body element, such that the result is valid PDF/UA tagging.

In general, it is always safe to use `-xpp-auto={1}` in cases where there is no source XML element for the list item label.

Footnote Tagging

Footnotes and endnotes should be tagged using the ″Note″ PDF tag.

Each Note tag shall have a unique entry in the ID key, as specified in the PDF 1.7 specification. Divpdf automatically creates a unique ID using the footnote (DLD) id. A Note may have a label (structure type Lbl) as a child, which would be for the footnote reference number or symbol.

Note: *Beginning with PDFLib version 10, the* `Placement` *option is not supported for the* `Note` *PDF tag.*

Footnotes should have the option `direct={false}` in the `-xpp-pdf-tag-options` CSS property, which promotes the Note from an inline element to a block structure element. Your XML and CSS for tagging a footnote would then resemble the following:

XML:

```
<foot id=″f1″> . . . text of footnote --- </foot>
```

CSS:

```
foot {
    -xpp-pdf-tag: ″Note″;
    -xpp-pdf-tag-options: ″direct={false}″;
}
```

This option enables divpdf to place each footnote as the next child of the document (that is, root) element, thus avoiding PDF/UA tag nesting errors. Otherwise, you may encounter an exception reported by divpdf (from PDFlib) such as the following:

```
PDFlib exception occurred:
[3096] PDF_begin_item: Element type ’Document’ not
allowed as parent for item ’Note’
```

Footnote Request/Reference Tagging

If the footnote request or reference in XPP is a xymacro, there is no way to tag it. If the footnote reference is an XML element, however, the ″Reference″ PDF tag can be used.

Alternatively, you can create a `<Span>` tag with the `ActualText` option as in the following example.

Example *Footnote Request/Reference Tagging*

XML:

```
<footref id="f1"/>
```

CSS:

```
footref[id] {
    -xpp-pdf-tag: "Span";
    -xpp-pdf-tag-options: "ActualText={reference to
footnote attr(id)}";
}
```

Footnote and Pickup Placement in the PDF tag Structure Tree

Although PDF/UA does not prescribe where the footnote/pickup is placed within the tag structure tree, divpdf must avoid inserting them under a parent where it would be invalid according to the PDF/UA nesting rules.

Consequently, divpdf always creates each footnote/pickup as the next child of the document (that is, root) element.

Tag Nesting Rules

Various rules must be observed for creating PDF tags (structure elements). These rules are summarized in the following table:

Table 8-9 *Tag Nesting Rules*

Item	*Rule*
direct content	The following elements may contain direct page content, that is, text, image, or vector graphics: • H, H1, H2, ... • P • Lbl,LBody • TH,TD • Span, Quote, Note, Reference, BibEntry, Code • Link, Annot • Figure, Formula • RB, RT, RP, WT, WP • Artifact, ASpan, ReversedChars, Clip All other elements require an intermediate structure element before direct page content can be added. PDFlib throws an exception when attempting to add direct content.

Table 8-9 *Tag Nesting Rules (Continued)*

Item	*Rule*
grouping elements	The following grouping elements must not contain direct content, ASpan or inline elements as children, that is, a block-level element must be created before content can be created: Document, Part, Art, Sect, Div, BlockQuote, Caption, TOC, TOCI, Index, NonStruct, Private.
block-level elements	The following block-level elements must not contain direct content as children, that is, a suitable grouping element or block-level element must be created before content can be created: L, LI, Table, TR, THead, TFoot, TBody **Strict rule:** The P element may not contain grouping elements.
pseudo and inline elements	Pseudo elements such as Artifact, ASpan, ReversedChars, Clip, and the following inline elements cannot have any descendants if direct=true: Code, BibEntry, Note, Quote, Reference, Span. However, these elements may have children if direct=false. Artifacts cannot be created within pseudo elements and the following elements with direct=true: Span, Quote, Note, Reference, BibEntry, Code.
table elements	Table elements may contain one or more TR elements, or an optional THead followed by one or more TBody elements and an optional TFoot. In addition, Table elements may have a Caption element as its first or last child. TR elements may contain TH and TD elements. TH and TD elements may not contain TR, TH, TD, THead, TBody, or TFoot THead, TBody, TFoot elements can contain only TR elements, and can only have Table as parent. TR may only have Table, THead, TBody, or TFoot as parent.
list elements	L elements may optionally contain a Caption element and one or more LI elements. LI elements may contain one or more Lbl or LBody elements or both. LI may only have L as parent. LBody may only have LI as parent.
table of contents	TOC elements may contain an optional Caption element as first child, and one or more TOCI and TOC elements (also in combination). TOCI elements may contain only Lbl, Reference, NonStruct, P, and TOC elements. TOCI may only have TOC as parent.
Label element	Lbl may only have LI, TOCI, BibEntry, or Note as parent.

Table 8-9 *Tag Nesting Rules (Continued)*

Item	*Rule*
interactive elements: links and annotations	The following element types are not allowed as parent for Link, Annot, and Form: table elements, inline elements, Ruby and Warichu elements, pseudo elements. Annot elements cannot be nested. Link elements cannot be nested.
Japanese Ruby and Warichu	Ruby can have RB, RT, and RP as children, but not any other element types. Warichu can have WT and WP as children, but not any other element types.

Importing PDF/UA-1 PDF Files

If you import a PDF file which is labeled as PDF/UA-1-compliant into the `divpdf` (`-pdfua`) output, `divpdf` first checks that there is at least one ″top level tag″ in the imported PDF, otherwise an error is output and the PDF is rasterized and imported as a graphic (Artifact).

Otherwise, `divpdf` attempts to import the tags of the imported PDF file in the original place.

If that fails, and if the currently active tag is a ″grouping″ tag, `divpdf` tries to import the PDF file by inserting a ″P″ as the top level tag.

If that fails, an error message is output and the imported PDF is rasterized and imported as an ″Artifact″.

If the tags of the PDF/UA-1 file cannot be imported, confirm the following in the imported file:

- The root tag of the imported file is a valid child of the current tag in `divpdf` when the file is imported.
- The imported file is correctly labeled as PDF/UA-1 in the PDF metadata. This can be verified in Adobe Acrobat DC Pro and other third-party tools.
- The imported PDF passes the ″accessibility checks″ using Acrobat DC Pro, or PAC3, or other third party PDF/UA-1 validity checker.

Support for 3D PDF Images

When a PDF is imported into a Job or Division, divpdf preserves and outputs multimedia or 3D objects embedded within it, such as:

- Model Trees
- Custom Views
- Rendering Modes
- Colors

Other PDF Objects

All embedded Page Actions are preserved in divpdf output, overriding any previously imported actions.

Most types of Annotations are preserved, including the following:

- 3D
- Caret
- Circle
- FileAttachment
- FreeText
- Highlight
- Ink
- Line
- Link
- Movie1
- Polygon
- PolyLine
- Popup
- Projection1
- Redact1
- RichMedia
- Screen
- Sound1
- Square
- Squiggly
- Stamp
- StrikeOut
- Text
- Underline

Limitations

Due to limitations in the PDF drawing package (PDFlib), PDF form field objects (such as checkbox, combobox, listbox, pushbutton, radiobutton, signature, or textfield) are not imported if embedded in a PDF.

In addition, because the PDFlib package does not support GIF images, divpdf converts embedded GIFs to PNG format.

PDF Security Settings for Direct-to-PDF output

Options available in the PDF Security (Direct to PDF only) frame on the PS/PDF tab of the Print dialog allow you to apply security settings to Direct-to-PDF (divpdf) output. The following table describes the PDF security options enabled on the PS/PDF tab when you select `PDF file` on the Print tab:

Table 8-10 *PDF Security Settings for Direct-to-PDF Output*

Option	***Description***
`noprint`	Acrobat prevents users from printing the file.
`nomodify`	Acrobat prevents users from adding form fields or making other changes to the file.
`nocopy`	Acrobat prevents users from copying or extracting text or graphics, and disables accessibility features.
`noannots`	Acrobat prevents users from adding or changing comments or form fields.
`noforms`	Implies `noannots` and prevents users from filling form fields, even if `noannots` was not specified.
`noaccessible`	Acrobat prevents users from extracting text or graphics from the file for accessibility purposes.
`noassemble`	Implies `nomodify` and prevents users from inserting, deleting, or rotating pages, or creating bookmarks and thumbnails, even if `nomodify` was not specified.
`nohiresprint`	Acrobat prevents high-resolution printing. Unless `noprint` was specified, users can print a low-resolution rendition of the page using the Adobe ”Print as Image” feature.
`plainmetadata`	Available only for files rendered in PDF 1.5 (and later) format, this option causes PDF metadata to remain unencrypted, even for encrypted documents.

Command line switches equivalent to each of these options are available for the `divpdf` command. For further information, refer to *XML Professional Publisher: Command Line Utilities*.

Applying PDF Security Settings

To apply PDF security settings to Direct-to-PDF output, you must specify a master password in the `Master Password` field in the PDF Security (Direct to PDF only) frame of the PS/PDF tab. The master password grants a user full access to the PDF file, based on the security permissions (copy, modify, print, etc.) but includes the ability to change its security settings using Adobe Acrobat DC Pro (or an equivalent tool). Attempting to set any security options without specifying a master password causes the divpdf program to fail with the following error:

```
Cannot create PDF file; Empty master password not
allowed.
```

In addition, you can specify a user password in the `User Password` field in the PDF Security (Direct to PDF) frame. The user password will be required to open the PDF document. Once opened using the user password, the user can copy, modify, annotate, and print the PDF document, subject to the security restrictions set, but cannot change the security settings.

Specifying passwords causes the divpdf program to encrypt the PDF output with the following algorithms:

- PDF 1.6 (Acrobat 7) and PDF 1.7: ISO 32000-1, (Acrobat 8): AES-128
- PDF 1.7ext3 or later: AES-256

Note: *You can set the PDF version in the Command dialog using the* `-pdfversion` *option. If you specify an invalid version, the divpdf program fails with the following error:*

```
PDF_begin_document: Option ’compatibility’ has bad
keyword XXXX.
```

Appendix A

Showpdf Messages

This appendix contains the text of error messages generated by the Showpdf program that checks PDF markup in a job and describes their meaning.

There are three categories of messages:

- Progress Messages
- Warning Messages
- Fatal Error Messages

Progress Messages

The following messages show the progress of the *showpdf* program.

Table A-1 *Showpdf Progress Messages*

Message	***Description***
"showpdf Vn.n XSFREV=nnn"	Startup message identifies the showpdf program version and XSF revision number.
Processing job/division"	Names the job and divisions to process.
*"Pg p-n TYPE **text**"*	Output when the PDF start and end macros are on the same line. Lists the division page and line number of PDF start and end macros, the PDF type, and optionally the text (if the –t switch is specified)
*"Pg p-n TYPE **text** Pg p-n"*	Output when the PDF start and end macros are on a different page or line. Lists the division page and line number of a PDF start macro, the PDF type, optionally the text (if the –t switch is specified), and the page and line number of a PDF end macro.
*"**div** Scanned Page p"*	Completed processing of division page number without finding any PDF macros.
*"**div** Located Page p: n PDF macros"*	Completed processing of division page number and found *n* number of PDF macros.
"n PDF macros on n line on n pages"	At end of job, reports the total number of PDF macros found, on the total number of lines, on the total number of pages across all divisions.
"n divisions processed, n PDF errors"	At end of job, reports the total number of divisions processed and the total number of PDF macro errors across all divisions.
"...exit showpdf..."	Completion message
"Opened division"	With the –X switch, names the division being processed
"Next Page p"	With the –X switch, after finding a PDF start macro, scanning ahead to Page p looking for the matching PDF macro end.

Warning Messages

The following error messages warn about invalid information found while processing PDF start and end macros. After displaying a message, the program continues.The word ERROR: precedes each message. If job processing is active, the division name precedes the message.

To help you locate and correct an error before distilling the document, “Pg p-n” indicates the division page type/number and the line number within a stream on that page where the error occurred. For example, “chap01 Pg 8-10” tells you an error exists in the chap01 division, on page 8, on line 10.

Table A-2 uses the following conventions:
divnam is a division name and appears only if you are doing job processing.
p represents a page number.
n represents a line number.
TYPE represents a value in the PDF macro Type field.

Table A-2 *Showpdf Warning Messages*

Message	***Description***
Pg p-n: <:pdfs;TYPE; ... missing pdfe.	A **<:pdfs>** macro is specified without an ending **<:pdfe>** macro in the same stream. Macros must be used in pairs.
Pg p-n: <:pdfs;TYPE; ... missing ‘>’	The end macro character (**>**) is missing from either a PDF start or end macro.
Pg p-n: <:pdfs;TYPE; ... invalid delimiter	A semi-colon (;) is the required delimiter between macro fields. Another character was used.
Pg p-n: <:pdfs;TYPE; ... missing delimiter	A semi-colon (;) delimiter is required and is missing from a macro.
Pg p-n <:pdfs;TYPE; ... extra delimiter	An extraneous semi-colon (;) delimiter is present in a macro.
Pg p-n: <:pdfs;TYPE; ... invalid type	The *type* field is not a valid PDF macro type. (See the –types switch.)
Pg p-n: <:pdfs;TYPE; ... end w/o start	A **<:pdfe>** end macro was not paired with a **<:pdfs>** beginning macro.
Pg p-n: <:pdfs;TYPE; ... invalid macro	The passthrough macro is not **<:pdfs>** or **<:pdfe>**.

Table A-2 *Showpdf Warning Messages*

Message	***Description***
Pg p-n: <::pdfs;TYPE; ... nested start	After a PDF start macro was found, another start macro was found in the same stream that was of a *type* that cannot be nested, without first having a corresponding PDF end macro. Only certain types of PDF macros can be nested.
[Pg p-n: <::pdfs;TYPE; ... start/end types	The *type* field for a PDF start macro is not the same *type* as the ending macro. (See the –e switch.)
Pg p-n: <:pdfs;TYPE; ... no count delim	For a TOC type macro, the semi-colon delimiter (;) to start the required count field (number of children) is missing.
Pg p-n: <::pdfs;TYPE; ... no div name delim	For a *TOC* or a *INDEX* macro type, the semi-colon delimiter to start the required division name field is missing.
Pg p-n: <:pdfs;TYPE; ... unknown div name	The division name field in a TOC or INDEX macro is not the name of a division in the process list. It may necessary to run CITI again.
Pg p-n: <:pdfs;TYPE; ... color field error	The optional data field contains /Color... but does not have a matching pair of brackets([ ]) following it.
Pg p-n:/Color...color data error 'c'	The optional data field contains /Color...but the data between the open bracket ([ ) and the close bracket ( ]) contains the illegal character **c**. Legal data characters are 0 through 9, periods, and spaces.
Pg p-n:/Color...missing space delim	The optional data field contains /Color...but the data between the open bracket ([ ) and the close bracket ( ]) does not contain two or more spaces to separate the three required numeric fields.
Pg p-n:/File...missing 1)'	The optional data for 2PAGE or 2DEST PDF type contains /File...but the following filespec does not start with an open parenthesis (( ).

Table A-2 *Showpdf Warning Messages*

Message	***Description***
Pg p-n:/File...missing ‘)’	The optional data for 2PAGE or 2DEST PDF type contains /File...but the following filespec does not end with a close parenthesis.
Pg p-n: <:pdfs;TYPE; ... bad optional data	The optional data field contains a mismatched pair of brackets, that is, an open bracket ([ ) without a close bracket ( ]) or a close bracket ( ]) without an open bracket ([ ).
Pg p-n: <:pdfs;TYPE; ... missing destination name	A destination name is required and is missing from a macro.

Fatal Error Messages

The following error messages are fatal. After the message displays, Showpdf processing stops.

Table A-3 *Fatal Showpdf Error Messages*

Message	***Description***
*Cannot chdir to "**divnam**" (errno=#)*	A specified division does not exist where *divnam* is the division name; *errno* is the system error number.
Illegal begin/end page numbers	A –p and/or –P switch is illegal. Invalid page number(s) or different start/end page types.
–div without –job is illegal	A list of division names cannot be specified unless job processing is enabled.
***option1** without **option2** is illegal*	Some processing options must be used together, option *1* was specified but option *2* was not.
*Illegal argument -div **divnam***	*divname* is not a legal division name(s) list.
Cannot access :x2a_default.p	The XSF to ASCII conversion table needed for processing could not be found.
***option1** with **option2** is illegal*	The combination of switches is invalid.
*Cannot open Page **#***	Where # is a page number, the specified page file could not be opened (Maybe it was deleted from the division directory.)

Appendix B

PDF Creation Options in Ghostscript

This appendix contains the list and descriptions of PDF creation options that Ghostscript supports.

Tip! Refer to Chapter 13, ″Printing″ in the *XPP User Guide* for information on PDF print options.

Common Options

The following table lists some common options for Ghostscript:

Option	*Description*
–r	Sets the resolution for pattern fills and for fonts that must be converted to bitmaps.
–dProcessColorModel	Sets the color space to be used for device-dependent colors in the output. This value may be /DeviceGray, /DeviceRGB, or /DeviceCMYK. /DeviceRGB is the default value. ***Note:*** *This does not affect images. For more information, see Limitations.*
–dCompressFonts	Defines whether Ghostscript compresses embedded fonts in the output. The default value is true. Set this to false only for debugging.
–dFastWebView	Specifies whether Ghostscript optimizes PDFs.
–dUseFastColor	Specifies whether Ghostscript uses ICC color profiles. For improved performance, XPP has ICC color profiles off. To use ICC color profiles, use the -dUseFastColor=false option.
–dMaxInlineImageSize	Specifies the maximum size of an inline image, in bytes. For images larger than this size, gs creates an XObject instead of embedding the image into the context stream. The default value is 4000. Redundant inline images must be embedded each time they occur in the document, while multiple references can be made to a single XObject image. If the source document has multiple identical images, set the value to zero or a small integer to reduce the PDF file size. ***Note:*** *Values greater than 8100 may cause page failure in Adobe Reader. Because the largest "safe" value is data-dependent, the user should increase the parameter value from the default (4000) in increments of 1000 until the best tradeoff between generation time and viewing fidelity is achieved.*

Option	*Description*
–dPDFSETTINGS	Presets the distiller parameters to one of the following predefined settings: • /screen selects low-resolution output similar to the Acrobat Distiller ″Smallest File Size″ setting. • /ebook selects medium-resolution output similar to the Acrobat Distiller ″Standard-Classic″ setting. • /printer selects output similar to the Acrobat Distiller ″High Quality Print″ setting. • /prepress selects output similar to Acrobat Distiller ″Press Quality″ setting. • /default selects output intended to be useful across a wide variety of uses, possibly at the expense of a larger output file, similar to Acrobat Distiller ″Standard″ setting.

Security Options

The following table describes the switches you can use to create encrypted documents:

Option	***Description***
–sOwnerPassword	Defines that the document be encrypted with the specified owner password.
–sUserPassword	Defines the user password for opening the document. If empty, the user can open the document with no password, but the owner password is required to edit it.
–dPermissions	Defines the PDF permissions flag field. Negative values can represent unsigned integers with the highest bit set. Refer to *XML Professional Publisher: PDF Support in XPP* for more information about flag bits.
–dEncryptionR	Defines the encryption method revision number; the value can be either 2 or 3.
–dKeyLength	Defines the length, in bits, of the encryption key. The value must be a multiple of 8 in the interval (40, 128). If the length is not 40, set dEncryptionR to 3.

Other Options

The following table describes Ghostscript-supported optional control values for PDF. Add them to the command line using -dVALUE=xxxx in the *gsdistill* file or in the *_gsdistill_parms* file. The values in the Note column correspond to the notes that follow this table.

Parameter Name	***Note***	***Default***	***Screen***	***eBook***	***Printer***	***Prepress***
AlwaysEmbed		[]	=	=	=	=
AntiAliasColorImages	(0)	false	=	=	=	=
AntiAliasGrayImages	(0)	false	=	=	=	=
AntiAliasMonoImages	(0)	false	=	=	=	=
ASCII85EncodePages		false	=	=	=	=
AutoFilterColorImages		true	=	=	=	=
AutoFilterGrayImages		true	=	=	=	=
AutoPositionEPSFiles	(0)	true	=	=	=	=
AutoRotatePages		/PageBy Page	/PageBy Page	/All	/None	/None
Binding	(0)	/Left	=	=	=	=
CalCMYKProfile	(0)	()	=	=	=	=
CalGrayProfile	(0)	()	=	=	=	=
CalRGBProfile	(0)	()	=	=	=	=
CannotEmbedFontPolicy	(0)	/Warning	/Warning	/Warning	/Warning	/Error
ColorACSImageDict		(note 7)	(note 10)	(note 10)	(note 8)	(note 9)
ColorConversionStrategy	(6)	/Leave Color Unchanged	/RGB	/RGB	/Use Device Independent Color	/LeaveColor Unchanged
ColorImageAutoFilter Strategy		/JPEG	=	=	=	=
ColorImageDepth		-1	=	=	=	=
ColorImageDict		(note 7)	=	=	=	=
ColorImageFilter		/DCTEncode	=	=	=	=
ColorImageDownsample Threshold		1.5	=	=	=	=
ColorImageDownsample Type	(3)	/Bicubic	/Average	/Average	/Average	/Bicubic

Parameter Name	*Note*	*Default*	*Screen*	*eBook*	*Printer*	*Prepress*
ColorImageResolution		150	72	150	300	300
CompatibilityLevel		1.7	1.5	1.5	1.7	1.7
CompressPages		true	=	=	=	=
ConvertCMYKImagesTo RGB		false	=	=	=	=
ConvertImagesToIn-dexed	(0)	true	=	=	=	=
CoreDistVersion		5000	=	=	=	=
CreateJobTicket	(0)	false	false	false	true	true
DefaultRenderingIntent		/Default	=	=	=	=
DetectBlends	(0)	true	=	=	=	=
DoThumbnails	(0)	false	false	false	false	true
DownsampleColor Images		false	true	true	true	true
DownsampleGrayImages		false	true	true	true	true
DownsampleMono Images		false	true	true	true	true
EmbedAllFonts		true	=	=	=	=
EmitDSCWarnings	(0)	false	=	=	=	=
EncodeColorImages		true	=	=	=	=
EncodeGrayImages		true	=	=	=	=
EncodeMonoImages		true	=	=	=	=
EndPage	(0)	{--.call endpage--}	=	=	=	=
FastWebView	(5)	false	=	=	=	=
GrayACSImageDict		(note 7)	(note 7)	(note 10)	(note 8)	(note 9)
GrayImageAutoFilter Strategy		/JPEG	=	=	=	=
GrayImageDepth		-1	=	=	=	=
GrayImageDict		(note 7)	=	=	=	=
GrayImageDownsample Threshold		1.5	=	=	=	=
GrayImageDownsample Type	(3)	/Subsample	/Average	/Average	/Average	/Bicubic
GrayImageFilter		/DCTEncode	=	=	=	=

Parameter Name	*Note*	*Default*	*Screen*	*eBook*	*Printer*	*Prepress*
GrayImageResolution		150	72	150	300	300
ImageMemory	(0)	524288	=	=	=	=
LockDistillerParams		false	=	=	=	=
LZWEncodePages	(2)	false	=	=	=	=
MaxSubsetPct		100	=	=	=	=
MonoImageDepth		-1	=	=	=	=
MonoImageDict		<</K -1>>	=	=	=	=
MonoImageDownsample Threshold		1.5	=	=	=	=
MonoImageDownsample Type		/Subsample	=	=	=	=
MonoImageFilter		/CCITTFax Encode	=	=	=	=
MonoImageResolution		300	300	300	1200	1200
NeverEmbed		(notes 11, 12)	(notes 11, 12)=	(notes 11, 12)	[] (note 12)	[] (note 12)
OffOptimizations		0	=	=	=	=
OPM		1	=	=	=	=
Optimize	(0,5)	false	true	true	true	true
ParseDSCComments		true	=	=	=	=
ParseDSCCommentsFor DocInfo		true	=	=	=	=
PassThroughJPEGImages	(14)	true	=	=	=	=
PDFEndPage	(0)	-1	=	=	=	=
PreserveCopyPage	(0)	true	=	=	=	=
PreserveEPSInfo	(0)	true	false	false	true	true
PreserveHalftoneInfo		false	=	=	=	=
PreserveOPIComments	(0)	true	false	false	true	true
PreserveOverprint Settings		true	false	false	true	true
sRGBProfile	(0)	()	=	=	=	=
StartPage	(0)	1	=	=	=	=
SubsetFonts		true	=	=	=	=
TransferFunctionInfo	(4)	/Preserve	/Preserve	/Preserve	/Apply	/Preserve

Parameter Name	*Note*	*Default*	*Screen*	*eBook*	*Printer*	*Prepress*
UCRandBGInfo		/Preserve	/Remove	/Remove	/Preserve	/Preserve
UseFastColor	(13)	false	=	=	=	=
UseFlateCompression	(2)	true	=	=	=	=
UsePrologue	(0)	false	=	=	=	=

Note 0: You can set and query this parameter, but it currently has no effect.

Note 1: No longer applies.

Note 2: Ghostscript does not use LZW compression: all requests for LZW compression are ignored. UseFlateCompression is treated as always on, but the switch CompressPages can be set to false to turn off page level stream compression.

Note 3: The xxxDownsampleType parameters can also have the value /Bicubic (a Distiller 4 feature), this will use a Mitchell filter. Note: if a non-integer downsample factor is used the code will clamp to the nearest integer (if the difference is less than 0.1) or will silently switch to the old bicubic filter, NOT the Mitchell filter.

Note 4: The default for transfer functions is to preserve them, this is because transfer functions are a device-dependent feature, a set of transfer functions designed for an RGB device will give incorrect output on a CMYK device for instance. PDF 2.0 deprecates the use of transfer functions, and so when producing PDF 2.0 compatible output if the TransferFunctionInfor is set to /Preserve it will be silently replaced with /Apply. You can instead specifically set TransferFunctionInfo to /Remove when producing PDF 2.0 in order to avoid the transfer function being applied.

Note 5: Ghostscript implements optimization (linearization) using the -dFastWebView option to the distiller. This is a single pass optimization, which is initiated with the -pdfoptgs switch to psfmtdrv via the command line, or through PathFinder by checking Fast web view (**File > Print > PS/PDF** tab).

Note 6: The value UseDeviceIndependentColorForImages works the same as UseDeviceIndependentColor. The value sRGB actually converts to RGB with the default Ghostscript conversion. The new Ghostscript-specific value Gray converts all colors to DeviceGray. It is no longer necessary to set ProcessColorModel when selecting Gray, RGB or CMYK. It is also no longer necessary to set UseCIEColor for UseDeviceIndependentColor to work properly, and the use of UseCIEColor is now strongly discouraged.

Note 7: The Default image parameter dictionary is
<</QFactor 0.9/Blend 1 /HSamples [2 1 1 2] /VSamples [2 1 1 2]>>

Note 8: The Printer ACS image parameter dictionary is <</QFactor 0.4 /Blend 1 /ColorTransform 1 /HSamples [1 1 1 1] /VSamples [1 1 1 1]>>

Note 9: The Prepress ACS image parameter dictionary is <</QFactor 0.15 /Blend 1 /ColorTransform 1 /HSamples [1 1 1 1] /VSamples [1 1 1 1]>>

Note 10: The Screen and eBook ACS image parameter dictionary is </QFactor 0.76 /Blend 1 /ColorTransform 1 /HSamples [2 1 1 2] /VSamples [2 1 1 2]>>

Note 11: The Default, Screen, and eBook settings never embed the 14 standard fonts (Courier, Helvetica, and Times families; Symbol; and ZapfDingbats).

Note 12: NeverEmbed can include CID font names. If you substitute a CID font in lib/cidfmap, Ghostscript uses the substitute font name when the CID font is embedded, and uses the original CID font name when it is not embedded. NeverEmbed should always specify the original CID font name.

Note 13: Specifies whether Ghostscript uses ICC color profiles. For improved performance, XPP has ICC color profiles off. To use ICC color profiles, use the -dUseFastColor=false option.

Note 14: When true, image data in the source which is encoded using the DCT (JPEG) filter will not be decompressed and then recompressed on output. This prevents the multiplication of JPEG artefacts caused by lossy compression. PassThroughJPEGImages currently only affects simple JPEG images. It has no effect on JPX (JPEG2000) encoded images, or masked images. In addition this parameter will be ignored if the source data is going to be modified. This can happen if the image is being downsampled, changing color space, or having transfer functions applied. Note that this parameter essentially overrides the EncodeColorImages and EncodeGrayImages parameters if they are false; the image will still be written with a DCTDecode filter.

Index

RWS Group
201 Edgewater Drive
Wakefield, MA 01880-6216
www.rws.com