Portable Document Format Publishing with GNU Troff

Keith Marshall
<keith.d.marshall@ntlworld.com>



A GNU MANUAL

Table of Contents

1. Introduction 1

2. Exploiting PDF Document Features 3

2.1. The `pdfmark` Operator 3
2.2. Selecting an Initial Document View 3
2.3. Adding Document Identification Meta-Data 4
2.4. Creating a Document Outline 4

2.4.1. A Basic Document Outline 4
2.4.2. Hierarchical Structure in a Document Outline 5
2.4.3. Associating a Document View with an Outline Reference 5
2.4.4. Folding the Outline to Conceal Less Significant Headings 6
2.4.5. Outlines for Multipart Documents 6
2.4.6. Delegation of the Outline Definition 7

2.5. Adding Reference Marks and Links 7

2.5.1. Optional Features of the `pdfhref` Macro 8
2.5.2. Marking a Reference Destination 10

2.5.2.1. Mapping a Destination for Cross Referencing 10
2.5.2.2. Associating a Document View with a Reference Mark 11

2.5.3. Linking to a Marked Reference Destination 11

2.5.3.1. References within a Single PDF Document 11
2.5.3.2. References to Destinations in Other PDF Documents 12

2.5.4. Linking to Internet Resources 13

2.5.5. Establishing a Format for References 13

2.5.5.1. Using Colour to Demarcate Link Regions 13
2.5.5.2. Specifying Reference Text Explicitly 14
2.5.5.3. Using Automatically Formatted Reference Text 14
2.5.5.4. Customising Automatically Formatted Reference Text 14

2.5.6. Problematic Links 14

2.5.6.1. Links with a Page Transition in the Active Region 14

2.6. Annotating a PDF Document using Pop-Up Notes 15

2.7. Synchronising Output and `pdfmark` Contexts 15

3. PDF Document Layout 16

3.1. Using `pdfmark` Macros with the `ms` Macro Package 16

3.1.1. `ms` Section Headings in PDF Documents 16

3.1.1.1. The `XN` Macro 16

4. The PDF Publishing Process 16

4.1. Resolving Cross References 16

4.1.1. Creating a Document Reference Map 16
4.1.2. Deploying a Document Reference Map 16

1. Introduction

It might appear that it is a fairly simple matter to produce documents in Adobe® "Portable Document Format", commonly known as PDF, using GNU Troff (`groff`) as the document formatter. Indeed, `groff`'s default output format is the native Adobe® PostScript® format, which PDF producers such as Adobe® Acrobat® Distiller®, or GhostScript, expect as their input format. Thus, the PDF production process would seem to entail simply formatting the document source with `groff`, to produce a PostScript® version of the document, which can subsequently be processed by Acrobat® Distiller® or GhostScript, to generate the final PDF document.

For many PDF production requirements, the production cycle described above may be sufficient. However, this is a limited PDF production method, in which the resultant PDF document represents no more than an on screen image of the printed form of the document, if `groff`'s PostScript® output were printed directly.

The Portable Document Format provides a number of features, which significantly enhance the experience of reading a document on screen, but which are of little or no value to a document which is merely printed. It *is* possible to exploit these PDF features, which are described in the Adobe® "pdfmark Reference Manual", with some refinement of the simple PDF production method, provided appropriate "feature implementing" instructions can be embedded into `groff`'s PostScript® rendering of the document. This, of course, implies that the original document source, which `groff` will process to generate the PostScript® description of the document, must include appropriate markup to exploit the desired PDF features. It is this preparation of the `groff` document source to exploit a number of these features, which provides the principal focus of this document.

The markup techniques to be described have been utilised in the production of the PDF version of this document itself. This has been formatted using `groff`'s `ms` macro package; thus, usage examples may be found in the document source file, `./contrib/pdfmark/pdfmark.ms`, to which comments have been added, to help identify appropriate markup examples for implementing PDF features, such as:–

- Selecting a default document view, which defines how the document will appear when opened in the reader application; for example, when this document is opened in Acrobat® Reader, it should display the top of the cover sheet, in the document view pane, while a document outline should appear to the left, in the "Bookmarks" pane.
- Adding document identification "meta-data", which can be accessed, in Acrobat® Reader, by inspecting the "File/Document Properties/Summary".
- Creating a document outline, which will be displayed in the "Bookmarks" pane of Acrobat® Reader, such that readers may quickly navigate to any section of the document, simply by clicking on the associated heading in the outline view.
- Embedding active links in the body of the document, such that readers may quickly navigate to related material at another location within the same document, or in another PDF document, or even to a related Internet resource, specified by its URI.
- Adding annotations, in the form of "sticky notes", at strategic points within the PDF document.

All of the techniques described have been tested on *both* GNU/Linux, and on Microsoft® Windows™2000 operating platforms, using `groff 1.19.1`,[1] in association with `AFPL GhostScript 8.14`.[2] Other tools employed, which should be readily available on *any* Unix™ or GNU/Linux system, are `sed`, `awk` and `make`, together with an appropriate text editor, for creating and marking up the `groff` input files. These additional utilities are not provided, as standard, on the Microsoft® Windows™ platform, but several third party implementations are available. Some worth considering include the MKS® Toolkit,[3] Cygwin,[4] or MSYS.[5] This list is by no means exhaustive, and should in no way be construed as an

1. Later versions should, and some earlier versions may, be equally suitable. See http://www.gnu.org/software/groff for information and availability of the latest version.
2. Again, other versions may be suitable. See http://ghostscript.com for information and availability.
3. A commercial offering; see http://mkssoftware.com/products/tk/default.asp for information.
4. A *free* but comprehensive POSIX emulation environment and Unix™ toolkit for 32-bit Microsoft® Windows™ platforms; see http://cygwin.com for information and download.
5. Another free, but minimal suite of common Unix™ tools for 32-bit Microsoft® Windows™, available for download from http://www.mingw.org; it *does* include those tools listed above, and is the package which was actually used when performing the Windows™2000 platform tests referred

endorsement of any of these packages, nor to imply that other similar packages, which may be available, are in any way inferior to them.

---

to in the text.

2. Exploiting PDF Document Features

To establish a consistent framework for adding PDF features, a `groff` macro package, named `pdfmark.tmac`, has been provided. Thus, to incorporate PDF features in a document, the appropriate macro calls, as described below, may be placed in the `groff` document source, which should then be processed with a `groff` command of the form

```
groff -Tps [-m name] -m pdfmark [-options ...] file ...
```

(Or use the PDF post-processor to avoid using ghostscript, `-Tpdf`).

It may be noted that the `pdfmark` macros have no dependencies on, and no known conflicts with, any other `groff` macro package; thus, users are free to use any other macro package, of their choice, to format their documents, while also using the `pdfmark` macros to add PDF features.

2.1. The `pdfmark` Operator

All PDF features are implemented by embedding instances of the **`pdfmark`** operator, as described in the Adobe® "pdfmark Reference Manual", into `groff`'s PostScript® output stream. To facilitate the use of this operator, the `pdfmark` macro package defines the primitive `pdfmark` macro; it simply emits its argument list, as arguments to a `pdfmark` operator, in the PostScript® output stream.

To illustrate the use of the `pdfmark` macro, the following is a much simplified example of how a bookmark may be added to a PDF document outline

```
.pdfmark \
    /Count 2 \
    /Title (An Example of a Bookmark with Two Children) \
    /View  [/FitH \n[PDFPAGE.Y]] \
    /OUT
```

In general, users should rarely need to use the `pdfmark` macro directly. In particular, the above example is too simple for general use; it *will* create a bookmark, but it does *not* address the issues of setting the proper value for the `/Count` key, nor of computing the `PDFPAGE.Y` value used in the `/View` key. The `pdfmark` macro package includes a more robust mechanism for creating bookmarks, (see section 2.4, "Creating a Document Outline"), which addresses these issues automatically. Nevertheless, the `pdfmark` macro may be useful to users wishing to implement more advanced PDF features, than those currently supported directly by the `pdfmark` macro package.

2.2. Selecting an Initial Document View

By default, when a PDF document is opened, the first page will be displayed, at the default magnification set for the reader, and outline and thumbnail views will be hidden. When using a PDF reader, such as Acrobat® Reader, which supports the `/DOCVIEW` class of the `pdfmark` operator, these default initial view settings may be overridden, using the `pdfview` macro. For example

```
.pdfview /PageMode /UseOutlines
```

will cause Acrobat® Reader to open the document outline view, to the left of the normal page view, while

```
.pdfview /PageMode /UseThumbs
```

will open the thumbnail view instead.

Note that the two `/PageMode` examples, above, are mutually exclusive — it is not possible to have *both* outline and thumbnail views open simultaneously. However, it *is* permitted to add `/Page` and `/View` keys, to force the document to open at a page other than the first, or to change the magnification at which the document is initially displayed; see the "pdfmark Reference Manual" for more information.

It should be noted that the view controlling meta-data, defined by the `pdfview` macro, is not written immediately to the PostScript® output stream, but is stored in an internal meta-data "cache", (simply implemented as a `groff` diversion). This "cached" meta-data must be written out later, by invoking the `pdfsync` macro, (see section 2.7, "Synchronising Output and `pdfmark` Contexts").

2.3. Adding Document Identification Meta-Data

In addition to the `/DOCVIEW` class of meta-data described above, (see section 2.2, “Selecting an Initial Document View”), we may also wish to include document identification meta-data, which belongs to the PDF `/DOCINFO` class.

To do this, we use the `pdfinfo` macro. As an example of how it is used, the identification meta-data attached to this document was specified using a macro sequence similar to:–

```
.pdfinfo /Title     PDF Document Publishing with GNU Troff
.pdfinfo /Author    Keith Marshall
.pdfinfo /Subject   How to Exploit PDF Features with GNU Troff
.pdfinfo /Keywords  groff troff PDF pdfmark
```

Notice that the `pdfinfo` macro is repeated, once for each `/DOCINFO` record to be placed in the document. In each case, the first argument is the name of the applicable `/DOCINFO` key, which *must* be named with an initial solidus character; all additional arguments are collected together, to define the value to be associated with the specified key.

As is the case with the `pdfview` macro, (see section 2.2, “Selecting an Initial Document View”), the `/DOCINFO` records specified with the `pdfinfo` macro are not immediately written to the PostScript® output stream; they are stored in the same meta-data cache as `/DOCVIEW` specifications, until this cache is explicitly flushed, by invoking the `pdfsync` macro, (see section 2.7, “Synchronising Output and `pdfmark` Contexts”).

2.4. Creating a Document Outline

A PDF document outline comprises a table of references, to “bookmarked” locations within the document. When the document is viewed in an “outline aware” PDF document reader, such as Adobe® Acrobat® Reader, this table of “bookmarks” may be displayed in a document outline pane, or “Bookmarks” pane, to the left of the main document view. Individual references in the outline view may then be selected, by clicking with the mouse, to jump directly to the associated marked location in the document view.

The document outline may be considered as a collection of “hypertext” references to “bookmarked” locations within the document. The `pdfmark` macro package provides a single generalised macro, `pdfhref`, for creating and linking to “hypertext” reference marks. This macro will be described more comprehensively in a later section, (see section 2.5, “Adding Reference Marks and Links”); the description here is restricted to its use for defining document outline entries.

2.4.1. A Basic Document Outline

In its most basic form, the document outline comprises a structured list of headings, each associated with a marked location, or “bookmark”, in the document text, and a specification for how that marked location should be displayed, when this bookmark is selected.

To create a PDF bookmark, the `pdfhref` macro is used, at the point in the document where the bookmark is to be placed, in the form

`.pdfhref O` *`<level> descriptive text ...`*

in which the reference class “**`O`**” stipulates that this is an outline reference.

Alternatively, for those users who may prefer to think of a document outline simply as a collection of bookmarks, the `pdfbookmark` macro is also provided — indeed, `pdfhref` invokes it, when processing the “**`O`**” reference class operator. It may be invoked directly, in the form

`.pdfbookmark` *`<level> descriptive text ...`*

Irrespective of which of the above macro forms is employed, the *`<level>`* argument is required. It is a numeric argument, defining the nesting level of the “bookmark” in the outline hierarchy, with one being the topmost level. Its function may be considered analagous to the *heading level* of the document’s section headings, for example, as specified with the `NH` macro, if using the `ms` macros to format the document.

All further arguments, following the *`<level>`* argument, are collected together, to specify the heading text which will appear in the document’s outline view. Thus, the outline entry for this section of this document, which has a level three heading, might be specified as

```
.pdfhref O 3 2.4.1. A Basic Document Outline
```

or, in the alternative form using the `pdfbookmark` macro, as

```
.pdfbookmark 3 2.4.1. A Basic Document Outline
```

2.4.2. Hierarchical Structure in a Document Outline

When a document outline is created, using the `pdfhref` macro as described in section 2.4.1, and any entry is added at a nesting level greater than one, then a hierarchical structure is automatically defined for the outline. However, as was noted in the simplified example in section 2.1, the data required by the `pdfmark` operator to create the outline entry may not be fully defined, when the outline reference is defined in the `groff` document source. Specifically, when the outline entry is created, its `/Count` key must be assigned a value equal to the number of its subordinate entries, at the next inner level of the outline hierarchy; typically however, these subordinate entries will be defined *later* in the document source, and the appropriate `/Count` value will be unknown, when defining the parent entry.

To resolve this paradox, the `pdfhref` macro creates the outline entry in two distinct phases — a destination marker is placed in the PostScript® output stream immediately, when the outline reference is defined, but the actual outline entry is stored in an internal "outline cache", until its subordinate hierarchy has been fully defined; it can then be inserted in the output stream, with its `/Count` value correctly assigned. Effectively, to ensure integrity of the document outline structure, this means that each top level outline entry, and *all* of its subordinates, are retained in the cache, until the *next* top level entry is defined.

One potential problem, which arises from the use of the "outline cache", is that, at the end of any document formatting run, the last top level outline entry, and any subordinates defined after it, will remain in the cache, and will *not* be automatically written to the output stream. To avoid this problem, the user should follow the guidelines given in section 2.7, to synchronise the output state with the cache state, (see section 2.7, "Synchronising Output and `pdfmark` Contexts"), at the end of the `groff` formatting run.

2.4.3. Associating a Document View with an Outline Reference

Each "bookmark" entry, in a PDF document outline, is associated with a specific document view. When the reader selects any outline entry, the document view changes to display the document context associated with that entry.

The document view specification, to be associated with any document outline entry, is established at the time when the outline entry is created. However, rather than requiring that each individual use of the `pdhref` macro, to create an outline entry, should include its own view specification, the actual specification assigned to each entry is derived from a generalised specification defined in the string `PDFBOOKMARK.VIEW`, together with the setting of the numeric register `PDFHREF.VIEW.LEADING`, which determine the effective view specification as follows:–

`PDFBOOKMARK.VIEW`

Establishes the magnification at which the document will be viewed, at the location of the "bookmark"; by default, it is defined by

```
.ds PDFBOOKMARK.VIEW /FitH \\n[PDFPAGE.Y] u
```

which displays the associated document view, with the "bookmark" location positioned at the top of the display window, and with the magnification set to fit the page width to the width of the window.

`PDFHREF.VIEW.LEADING`

Specifies additional spacing, to be placed between the top of the display window and the actual location of the "bookmark" on the displayed page view. By default, it is set as

```
.nr PDFHREF.VIEW.LEADING 5.0p
```

Note that `PDFHREF.VIEW.LEADING` does not represent true "leading", in the typographical sense, since any preceding text, set in the specified display space, will be visible at the top of the document viewing window, when the reference is selected.

Also note that the specification of `PDFHREF.VIEW.LEADING` is shared by *all* reference views defined by the `pdfhref` macro; whereas `PDFBOOKMARK.VIEW` is applied exclusively to outline references, there is no independent `PDFBOOKMARK.VIEW.LEADING` specification.

If desired, the view specification may be changed, by redefining the string PDFBOOKMARK.VIEW, and possibly also the numeric register PDFHREF.VIEW.LEADING. Any alternative definition for PDFBOOKMARK.VIEW *must* be specified in terms of valid view specification parameters, as described in the Adobe® "pdfmark Reference Manual".

Note the use of the register PDFPAGE.Y, in the default definition of PDFBOOKMARK.VIEW above. This register is computed by pdfhref, when creating an outline entry; it specifies the vertical position of the "bookmark", in basic groff units, relative to the *bottom* edge of the document page on which it is defined, and is followed, in the PDFBOOKMARK.VIEW definition, by the grops "u" operator, to convert it to PostScript® units on output. It may be used in any redefined specification for PDFBOOKMARK.VIEW, (or in the analogous definition of PDFHREF.VIEW, described in section 2.5.2.2, "Associating a Document View with a Reference Mark"), but *not* in any other context, since its value is undefined outside the scope of the pdfhref macro.

Since PDFPAGE.Y is computed relative to the *bottom* of the PDF output page, it is important to ensure that the page length specified to troff correctly matches the size of the logical PDF page. This is most effectively ensured, by providing *identical* page size specifications to groff, grops and to the PostScript® to PDF converter employed, and avoiding any page length changes within the document source.

Also note that PDFPAGE.Y is the only automatically computed "bookmark" location parameter; if the user redefines PDFBOOKMARK.VIEW, and the modified view specification requires any other positional parameters, then the user *must* ensure that these are computed *before* invoking the pdfhref macro.

2.4.4. Folding the Outline to Conceal Less Significant Headings

When a document incorporates many subheadings, at deeply nested levels, it may be desirable to "fold" the outline such that only the major heading levels are initially visible, yet making the inferior subheadings accessible, by allowing the reader to expand the view of any heading branch on demand.

The pdfmark macros support this capability, through the setting of the PDFOUTLINE.FOLDLEVEL register. This register should be set to the number of heading levels which it is desired to show in expanded form, in the *initial* document outline display; all subheadings at deeper levels will still be added to the outline, but will not become visible until the outline branch containing them is expanded. For example, the setting used in this document:

```
.\" Initialise the outline view to show only three heading levels,
.\" with additional subordinate level headings folded.
.\"
.nr PDFOUTLINE.FOLDLEVEL 3
```

results in only the first three levels of headings being displayed in the document outline, *until* the reader chooses to expand the view, and so reveal the lower level headings in any outline branch.

The initial default setting of PDFOUTLINE.FOLDLEVEL, if the document author does not choose to change it, is 10,000. This is orders of magnitude greater than the maximum heading level which is likely to be used in any document; thus the default behaviour will be to show document outlines fully expanded, to display all headings defined, at all levels within each document.

The setting of PDFOUTLINE.FOLDLEVEL may be changed at any time; however, the effect of each such change may be difficult to predict, since it is applied not only to outline entries which are defined *after* the setting is changed, but also to any entries which remain in the outline cache, *at* this time. Therefore, it is recommended that PDFOUTLINE.FOLDLEVEL should be set *once*, at the start of each document; if it *is* deemed necessary to change it at any other time, the outline cache should be flushed, (see section 2.7, "Synchronising Output and pdfmark Contexts"), *immediately* before the change, which should immediately preceed a level one heading.

2.4.5. Outlines for Multipart Documents

When a document outline is created, using the pdfhref macro, each reference mark is automatically assigned a name, composed of a fixed stem followed by a serially generated numeric qualifier. This ensures that, for each single part document, every outline reference has a uniquely named destination.

As the overall size of the PDF document increases, it may become convenient to divide it into smaller, individually formatted PostScript® components, which are then assembled, in the appropriate order, to create a composite PDF

document. While this strategy may simplify the overall process of creating and editing larger documents, it does introduce a problem in creating an overall document outline, since each individual PostScript® component will be assigned duplicated sequences of "bookmark" names, with each name ultimately referring to multiple locations in the composite document. To avoid such reference naming conflicts, the `pdfhref` macro allows the user to specify a "tag", which is appended to the automatically generated "bookmark" name; this may be used as a discriminating mark, to distinguish otherwise similarly named destinations, in different sections of the composite document.

To create a "tagged" document outline, the syntax for invocation of the `pdfhref` macro is modified, by the inclusion of an optional "tag" specification, *before* the nesting level argument, i.e.

```
.pdfhref O [-T <tag>] <level> descriptive text ...
```

The optional `<tag>` argument may be composed of any characters of the user's choice; however, its initial character *must not* be any decimal digit, and ideally it should be kept short — one or two characters at most.

By employing a different tag in each section, the user can ensure that "bookmark" names remain unique, throughout all the sections of a composite document. For example, when using the `spdf.tmac` macro package, which adds `pdfmark` capabilities to the standard `ms` package, (see section 3.1, "Using `pdfmark` Macros with the `ms` Macro Package"), the table of contents is collected into a separate PostScript® section from the main body of the document. In the "body" section, the document outline is "untagged", but in the "Table of Contents" section, a modified version of the `TC` macro adds an outline entry for the start of the "Table of Contents", invoking the `pdfhref` macro as

```
.pdfhref O -T T 1 \\*[TOC]
```

to tag the associated outline destination name with the single character suffix, "`T`". Alternatively, as in the case of the basic outline, (see section 2.4.1, "A Basic Document Outline"), this may equally well be specified as

```
.pdfbookmark -T T 1 \\*[TOC]
```

2.4.6. Delegation of the Outline Definition

Since the most common use of a document outline is to provide a quick method of navigating through a document, using active "hypertext" links to chapter and section headings, it may be convenient to delegate the responsibility of creating the outline to a higher level macro, which is itself used to define and format the section headings. This approach has been adopted in the `spdf.tmac` package, to be described later, (see section 3.1, "Using `pdfmark` Macros with the `ms` Macro Package").

When such an approach is adopted, the user will rarely, if ever, invoke the `pdfhref` macro directly, to create a document outline. For example, the structure and content of the outline for this document has been exclusively defined, using a combination of the `NH` macro, from the `ms` package, to establish the structure, and the `XN` macro from `spdf.tmac`, to define the content. In this case, the responsibility for invoking the `pdfhref` macro, to create the document outline, is delegated to the `XN` macro.

2.5. Adding Reference Marks and Links

Section 2.4 has shown how the `pdfhref` macro may be used to create a PDF document outline. While this is undoubtedly a powerful capability, it is by no means the only trick in the repertoire of this versatile macro.

The macro name, `pdfhref`, which is a contraction of "PDF HyperText Reference", indicates that the general purpose of this macro is to define *any* type of dynamic reference mark, within a PDF document. Its generalised usage syntax takes the form

```
.pdfhref <class> [-options ...] [--] [descriptive text ...]
```

where `<class>` represents a required single character argument, which defines the specific reference operation to be performed, and may be selected from:–

- **`O`** Add an entry to the document outline. This operation has been described earlier, (see section 2.4, "Creating a Document Outline").
- **`M`** Place a "named destination" reference mark at the current output position, in the current PDF document, (see section 2.5.2, "Marking a Reference Destination").
- **`D`** Specify the content of a PDF document reference dictionary entry; typically, such entries are generated automatically, by transformation of the intermediate output resulting from the use of

`pdfhref` "**M**", with the "**-X**" modifier, (see section 4.1.1, "Creating a Document Reference Map"); however, it is also possible to specify such entries manually, (see section 2.5.5.2, "Specifying Reference Text Explicitly").

L Insert an active link to a named destination, (see section 2.5.3, "Linking to a Marked Reference Destination"), at the current output position in the current PDF document, such that when the reader clicks on the link text, the document view changes to show the location of the named destination.

W Insert an active link to a "web" resource, (see section 2.5.4, "Linking to Internet Resources"), at the current output position in the current PDF document. This is effectively the same as using the "**L**" operator to establish a link to a named destination in another PDF document, (see section 2.5.3.2, "References to Destinations in Other PDF Documents"), except that in this case, the destination is specified by a "uniform resource identifier", or `URI`; this may represent any Internet or local resource which can be specified in this manner.

F Specify a user defined macro, to be called by `pdfhref`, when formatting the text in the active region of a link, (see section 2.5.5, "Establishing a Format for References").

Z Define the absolute position on the physical PDF output page, where the "hot-spot" associated with an active link is to be placed. Invoked in pairs, marking the starting and ending PDF page co-ordinates for each link "hot-spot", this operator is rarely, if ever, specified directly by the user; rather, appropriate `pdfhref` "**Z**" specifications are inserted automatically into the document reference map during the PDF document formatting process, (see section 4.1.1, "Creating a Document Reference Map").

I Initialise support for `pdfhref` features. The current `pdfhref` implementation provides only one such feature which requires initialisation — a helper macro which must be attached to a user supplied page trap handler, in order to support mapping of reference "hot-spots" which extend through a page transition; (see section 2.5.6.1, "Links with a Page Transition in the Active Region").

2.5.1. Optional Features of the `pdfhref` Macro

The behaviour of a number of the `pdfhref` macro operations can be modified, by including "*option specifiers*" after the operation specifying argument, but *before* any other arguments normally associated with the operation. In *all* cases, an option is specified by an "*option flag*", comprising an initial hyphen, followed by one or two option identifying characters. Additionally, *some* options require *exactly one* option argument; for these options, the argument *must* be specified, and it *must* be separated from the preceding option flag by one or more *spaces*, (tabs *must not* be used). It may be noted that this paradigm for specifying options is reminiscent of most Unix™ shells; however, in the case of the `pdfhref` macro, omission of the space separating an option flag from its argument is *never* permitted.

A list of *all* general purpose options supported by the `pdfhref` macro is given below. Note that not all options are supported for all `pdfhref` operations; the operations affected by each option are noted in the list. For *most* operations, if an unsupported option is specified, it will be silently ignored; however, this behaviour should not be relied upon.

The general purpose options, supported by the `pdfhref` macro, are:–

-N <***name***>
Allows the <*name*> associated with a PDF reference destination to be defined independently from the following text, which describes the reference. This option affects only the "**M**" operation of the `pdfhref` macro, (see section 2.5.2, "Marking a Reference Destination").

-E Also used exclusively with the "**M**" operator, the **-E** option causes any specified *`descriptive text`* arguments, (see section 2.5.2, "Marking a Reference Destination"), to be copied, or *echoed*, in the body text of the document, at the point where the reference mark is defined; (without the **-E** option, such *`descriptive text`* will appear *only* at points where links to the reference mark are placed, and where the standard reference display format, (see section 2.5.5, "Establishing a Format for References"), is used).

-D <***dest***>
Specifies the `URI`, or the destination name associated with a PDF active link, independently of the

following text, which describes the link and demarcates the link "hot-spot". This option affects the behaviour of the pdfhref macro's "**L**" and "**W**" operations.

When used with the "**L**" operator, the *<dest>* argument must specify a PDF "named destination", as defined using pdfhref with the "**M**" operator.

When used with the "**W**" operator, *<dest>* must specify a link destination in the form of a "uniform resource identifier", or URI, (see section 2.5.4, "Linking to Internet Resources").

-F *<file>*
When used with the "**L**" pdfhref operator, *<file>* specifies an external PDF file in which the named destination for the link reference is defined. This option *must* be specified with the "**L**" operator, to create a link to a destination in a different PDF document; when the "**L**" operator is used *without* this option, the link destination is assumed to be defined within the same document.

-P *<"prefix-text">*
Specifies *<"prefix-text">* to be attached to the *start* of the text describing an active PDF document link, with no intervening space, but without itself being included in the active area of the link "hot-spot"; it is effective with the "**L**" and "**W**" pdfhref operators.

Typically, this option would be used to insert punctuation before the link "hot-spot". Thus, there is little reason for the inclusion of spaces in *<"prefix-text">*; however, if such space is required, then the enclosing double quotes *must* be specified, as indicated.

-A *<"affixed-text">*
Specifies *<"affixed-text">* to be attached to the *end* of the text describing an active PDF document link, with no intervening space, but without itself being included in the active area of the link "hot-spot"; it is effective with the "**L**" and "**W**" pdfhref operators.

Typically, this option would be used to insert punctuation after the link "hot-spot". Thus, there is little reason for the inclusion of spaces in *<"affixed-text">*; however, if such space is required, then the enclosing double quotes *must* be specified, as indicated.

-T *<tag>*
When specified with the "**O**" operator, *<tag>* is appended to the "bookmark" name assigned to the generated outline entry. This option is *required*, to distinguish between the series of "bookmark" names generated in individual passes of the groff formatter, when the final PDF document is to be assembled from a number of separately formatted components; (see section 2.4.5, "Outlines for Multipart Documents").

-X This pdfhref option is used with either the "**M**" operator, or with the "**L**" operator.

When used with the "**M**" operator, (see section 2.5.2, "Marking a Reference Destination"), it ensures that a cross reference record for the marked destination will be included in the document reference map, (see section 2.5.2.1, "Mapping a Destination for Cross Referencing").

When used with the "**L**" operator, (see section 2.5.3, "Linking to a Marked Reference Destination"), it causes the reference to be displayed in the standard cross reference format, (see section 2.5.5, "Establishing a Format for References"), but substituting the *descriptive text* specified in the "pdfhref L" argument list, for the description specified in the document reference map.

-- Marks the end of the option specifiers. This may be used with all pdfhref operations which accept options, to prevent pdfhref from interpreting any following arguments as option specifiers, even if they would otherwise be interpreted as such. It is also useful when the argument list to pdfhref contains special characters — any special character, which is not valid in a groff macro name, will cause a parsing error, if pdfhref attempts to match it as a possible option flag; using the "--" flag prevents this, so suppressing the groff warning message, which would otherwise ensue.

Using this flag after *all* sequences of macro options is recommended, even when it is not strictly necessary, if only for the entirely cosmetic benefit of visually separating the main argument list from the sequence of preceding options.

In addition to the pdfhref options listed above, a supplementary set of two character options are defined. These supplementary options, listed below, are intended for use with the "**L**" operator, in conjunction with the **-F** *<file>*

option, to specify alternate file names, in formats compatible with the file naming conventions of alternate operating systems; they will be silently ignored, if used in any other context.

The supported alternate file name options, which are ignored if the **-F** <***file***> option is not specified, are:–

-DF <***dos-file***>
Specifies the name of the file in which a link destination is defined, using the file naming semantics of the MS-DOS® operating system. When the PDF document is read on a machine where the operating system uses the MS-DOS® file system, then <*dos-file*> is used as the name of the file containing the reference destination, overriding the <*file*> argument specified with the **-F** option.

-MF <***mac-file***>
Specifies the name of the file in which a link destination is defined, using the file naming semantics of the Apple® Macintosh® operating system. When the PDF document is read on a machine where the operating system uses the Macintosh® file system, then <*mac-file*> is used as the name of the file containing the reference destination, overriding the <*file*> argument specified with the **-F** option.

-UF <***unix-file***>
Specifies the name of the file in which a link destination is defined, using the file naming semantics of the Unix™ operating system. When the PDF document is read on a machine where the operating system uses POSIX file naming semantics, then <*unix-file*> is used as the name of the file containing the reference destination, overriding the <*file*> argument specified with the **-F** option.

-WF <***win-file***>
Specifies the name of the file in which a link destination is defined, using the file naming semantics of the MS-Windows® 32-bit operating system. When the PDF document is read on a machine where the operating system uses any of the MS-Windows® file systems, with long file name support, then <*win-file*> is used as the name of the file containing the reference destination, overriding the <*file*> argument specified with the **-F** option.

2.5.2. Marking a Reference Destination

The pdfhref macro may be used to create active links to any Internet resource, specified by its URI, or to any "named destination", either within the same document, or in another PDF document. Although the PDF specification allows link destinations to be defined in terms of a page number, and an associated view specification, this style of reference is not currently supported by the pdfhref macro, because it is not possible to adequately bind the specification for the destination with the intended reference context.

References to Internet resources are interpreted in accordance with the W3C standard for defining a URI; hence the only prerequisite, for creating a link to any Internet resource, is that the URI be properly specified, when declaring the reference; (see section 2.5.4, "Linking to Internet Resources"). In the case of references to "named destinations" in PDF documents, however, it is necessary to provide a mechanism for creating such "named destinations". This may be accomplished, by invoking the pdfhref macro in the form

.pdfhref M [**-N** <*name*>] [**-X**] [**-E**] [*descriptive text ...*]

This creates a "named destination" reference mark, with its name specified by <*name*>, or, if the **-N** option is not specified, by the first word of ***descriptive text***; (note that this imposes the restriction that, if the **-N** option is omitted, then *at least* one word of ***descriptive text*** *must* be specified). Additionally, a reference view will be automatically defined, and associated with the reference mark, (see section 2.5.2.2, "Associating a Document View with a Reference Mark"), and, if the **-X** option is specified, and no document cross reference map has been imported, (see section 4.1.2, "Deploying a Document Reference Map"), then a cross reference mapping record, (see section 2.5.2.1, "Mapping a Destination for Cross Referencing"), will be written to the stdout stream; this may be captured, and subsequently used to generate a cross reference map for the document, (see section 4.1.1, "Creating a Document Reference Map").

When a "named destination" reference mark is created, using the pdfhref macro's "**M**" operator, there is normally no visible effect in the formatted document; any ***descriptive text*** which is specified will simply be stored in the cross reference map, for use when a link to the reference mark is created. This default behaviour may be changed, by specifying the **-E** option, which causes any specified ***descriptive text*** to be "echoed" in the document text, at the

point where the reference mark is placed, in addition to its inclusion in the cross reference map.

2.5.2.1. Mapping a Destination for Cross Referencing

Effective cross referencing of *any* document formatted by `groff` requires multiple pass formatting. Details of how this multiple pass formatting may be accomplished, when working with the `pdfmark` macros, will be discussed later, (see section 4.1, "Resolving Cross References"); at this stage, the discussion will be restricted to the initial preparation, which is required at the time when the cross reference destinations are defined.

The first stage, in the process of cross referencing a document, is the generation of a cross reference map. Again, the details of *how* the cross reference map is generated will be discussed in section 4.1; however, it is important to recognise that *what* content is included in the cross reference map is established when the reference destination is defined — it is derived from the reference data exported on the `stderr` stream by the `pdfhref` macro, when it is invoked with the "**`M`**" operator, and is controlled by whatever definition of the string `PDFHREF.INFO` is in effect, when the `pdfhref` macro is invoked.

The initial default setting of `PDFHREF.INFO` is

```
.ds PDFHREF.INFO page \\n% \\$*
```

which ensures that the cross reference map will contain at least a page number reference, supplemented by any *`descriptive text`* which is specified for the reference mark, as defined by the `pdfhref` macro, with its "**`M`**" operator; this may be redefined by the user, to export additional cross reference information, or to modify the default format for cross reference links, (see section 2.5.5, "Establishing a Format for References").

2.5.2.2. Associating a Document View with a Reference Mark

In the same manner as each document outline reference, defined by the `pdfhref` macro with the "**`O`**" operator, (see section 2.4, "Creating a Document Outline"), has a specific document view associated with it, each reference destination marked by `pdfhref` with the "**`M`**" operator, requires an associated document view specification.

The mechanism whereby a document view is associated with a reference mark is entirely analogous to that employed for outline references, (see section 2.4.3, "Associating a Document View with an Outline Reference"), except that the `PDFHREF.VIEW` string specification is used, in place of the `PDFBOOKMARK.VIEW` specification. Thus, the reference view is defined in terms of:–

`PDFHREF.VIEW`

A string, establishing the position of the reference mark within the viewing window, and the magnification at which the document will be viewed, at the location of the marked reference destination; by default, it is defined by

```
.ds PDFHREF.VIEW /FitH \\n[PDFPAGE.Y] u
```

which displays the reference destination at the top of the viewing window, with the magnification set to fit the page width to the width of the window.

`PDFHREF.VIEW.LEADING`

A numeric register, specifying additional spacing, to be placed between the top of the display window and the actual position at which the location of the reference destination appears within the window. This register is shared with the view specification for outline references, and thus has the same default initial setting,

```
.nr PDFHREF.VIEW.LEADING 5.0p
```

as in the case of outline reference views.

Again, notice that `PDFHREF.VIEW.LEADING` does not represent true typographic "leading", since any preceding text, set in the specified display space, will be visible at the top of the viewing window, when the reference is selected.

Just as the view associated with outline references may be changed, by redefining `PDFBOOKMARK.VIEW`, so the view associated with marked reference destinations may be changed, by redefining `PDFHREF.VIEW`, and, if desired, `PDFHREF.VIEW.LEADING`; such changes will become effective for all reference destinations marked *after* these definitions are changed. (Notice that, since the specification of `PDFHREF.VIEW.LEADING` is shared by both outline

reference views and marked reference views, if it is changed, then the views for *both* reference types are changed accordingly).

It may again be noted, that the `PDFPAGE.Y` register is used in the definition of `PDFHREF.VIEW`, just as it is in the definition of `PDFBOOKMARK.VIEW`; all comments in section 2.4.3 relating to its use, and indeed to page position computations in general, apply equally to marked reference views and to outline reference views.

2.5.3. Linking to a Marked Reference Destination

Any named destination, such as those marked by the `pdfhref` macro, using it's "**`M`**" operator, may be referred to from any point in *any* PDF document, using an *active link*; such active links are created by again using the `pdfhref` macro, but in this case, with the "**`L`**" operator. This operator provides support for two distinct cases, depending on whether the reference destination is defined in the same document as the link, (see section 2.5.3.1, "References within a Single PDF Document"), or is defined as a named destination in a different PDF document, (see section 2.5.3.2, "References to Destinations in Other PDF Documents").

2.5.3.1. References within a Single PDF Document

The general syntactic form for invoking the `pdfhref` macro, when creating a link to a named destination within the same PDF document is

```
.pdfhref L [-D <dest-name>] [-P <prefix-text>] [-A <affixed-text>] \
    [-X] [--] [descriptive text ...]
```

where <*dest-name*> specifies the name of the link destination, as specified using the `pdfhref` "**`M`**" operation; (it may be defined either earlier in the document, to create a backward reference, or later, to create a forward reference).

If any *`descriptive text`* arguments are specified, then they will be inserted into the `groff` output stream, to define the text appearing in the "hot-spot" region of the link; this will be printed in the link colour specified by the string, `PDFHREF.TEXT.COLOUR`, which is described in section 2.5.5.1, "Using Colour to Demarcate Link Regions". If the **`-X`** option is also specified, then the *`descriptive text`* will be augmented, by prefacing it with page and section number indicators, in accordance with the reference formatting rules which are in effect, (see section 2.5.5, "Establishing a Format for References"); such indicators will be included within the active link region, and will also be printed in the link colour.

Note that *either* the **`-D`** <***`dest-name`***> option, *or* the *`descriptive text`* arguments, *but not both*, may be omitted. If the **`-D`** <***`dest-name`***> option is omitted, then the first word of *`descriptive text`*, i.e. all text up to but not including the first space, will be interpreted as the <***`dest-name`***> for the link; this text will also appear in the running text of the document, within the active region of the link. Alternatively, if the **`-D`** <***`dest-name`***> option *is* specified, and *`descriptive text`* is not, then the running text which defines the reference, and its active region, will be derived from the reference description which is specified when the named destination is marked, (see section 2.5.2, "Marking a Reference Destination"), and will be formatted according to the reference formatting rules which are in effect, when the reference is placed, (see section 2.5.5, "Establishing a Format for References"); in this case, it is not necessary to specify the **`-X`** option to activate automatic formatting of the reference — it is implied, by the omission of all *`descriptive text`* arguments.

The **`-P`** <***`prefix-text`***> and **`-A`** <***`affixed-text`***> options may be used to specify additional text which will be placed before and after the linked text respectively, with no intervening space. Such prefixed and affixed text will be printed in the normal text colour, and will not be included within the active region of the link. This feature is mostly useful for creating parenthetical references, or for placing punctuation adjacent to, but not included within, the text which defines the active region of the link.

The operation of the `pdfhref` macro, when used with its "**`L`**" operator to place a link to a named PDF destination, may best be illustrated by an example. However, since the appearance of the link will be influenced by factors established when the named destination is marked, (see section 2.5.2, "Marking a Reference Destination"), and also by the formatting rules in effect when the link is placed, the presentation of a suitable exanple will be deferred, until the formatting mechanism has been explained, (see section 2.5.5, "Establishing a Format for References").

2.5.3.2. References to Destinations in Other PDF Documents

The `pdfhref` macro's "**L**" operator is not restricted to creating reference links within a single PDF document. When the link destination is defined in a different document, then the syntactic form for invoking `pdfhref` is modified, by the addition of options to specify the name and location of the PDF file in which the destination is defined. Thus, the extended `pdfhref` syntactic form becomes

```
.pdfhref L -F <file> [-D <dest-name>] \
    [-DF <dos-file>] [-MF <mac-file>] [-UF <unix-file>] \
    [-WF <win-file>] [-P <prefix-text>] [-A <affixed-text>] \
    [-X] [--] [descriptive text ...]
```

where the **-F** <***file***> option serves *two* purposes: it both indicates to the `pdfhref` macro that the specified reference destination is defined in an external PDF file, and it also specifies the normal path name, which is to be used to locate this file, when a user selects the reference.

In addition to the **-F** <***file***> option, which *must* be specified when referring to a destination in an external PDF file, the **-DF** <***dos-file***>, **-MF** <***mac-file***>, **-UF** <***unix-file***> and **-WF** <***win-file***> options may be used to specify the location of the file containing the reference destination, in a variety of operating system dependent formats. These options assign their arguments to the `/DosFile`, `/MacFile`, `/UnixFile` and `/WinFile` keys of the generated `pdfmark` respectively; thus when any of these options are specified, *in addition to* the **-F** <***file***> option, and the document is read on the appropriate operating systems, then the path names specified by <***dos-file***>, <***mac-file***>, <***unix-file***> and <***win-file***> will be searched, *instead* of the path name specified by <***file***>, for each of the `MS-DOS`®, `Apple`® `Macintosh`®, `Unix`™ and `MS-Windows`® operating systems, respectively; see the "pdfmark Reference Manual", for further details.

Other than the use of these additional options, which specify that the reference destination is in an external PDF file, the behaviour of the `pdfhref` "**L**" operator, with the **-F** <***file***> option, remains identical to its behaviour *without* this option, (see section 2.5.3.1, "References within a Single PDF Document"), with respect to the interpretation of other options, the handling of the ***descriptive text*** arguments, and the formatting of the displayed reference.

Once again, since the appearance of the reference is determined by factors specified in the document reference map, and also by the formatting rules in effect when the reference is placed, the presentation of an example of the placing of a reference to an external destination will be deferred, until the formatting mechanism has been explained, (see section 2.5.5, "Establishing a Format for References").

2.5.4. Linking to Internet Resources

In addition to supporting the creation of cross references to named destinations in PDF documents, the `pdfhref` macro also has the capability to create active links to Internet resources, or indeed to *any* resource which may be specified by a Uniform Resource Identifier, (which is usually abbreviated to the acronym "URI", and sometimes also referred to as a Uniform Resource Locator, or "URL").

Since the mechanism for creating a link to a URI differs somewhat from that for creating PDF references, the `pdfhref` macro is invoked with the "**W**" (for "web-link") operator, rather than the "**L**" operator; nevertheless, the invocation syntax is similar, having the form

```
.pdfhref W [-D <URI>] [-P <prefix-text>] [-A <affixed-text>] \
    [--] descriptive text ...
```

where the optional **-D** <***URI***> modifier specifies the address for the target Internet resource, in any appropriate *Uniform Resource Identifier* format, while the ***descriptive text*** argument specifies the text which is to appear in the "hot-spot" region, and the **-P** <***prefix-text***> and **-A** <***affixed-text***> options have the same effect as in the case of local document links, (see section 2.5.3.1, "References within a Single PDF Document").

Notice that it is not mandatory to include the **-D** <***URI***> in the link specification; if it *is* specified, then it is not necessary for the URI to appear, in the running text of the document — the ***descriptive text*** argument exactly defines the text which will appear within the "hot-spot" region, and this need not include the URI. However, if the **-D** <***URI***> specification is omitted, then the ***descriptive text*** argument *must* be an *exact* representation of the URI, which *will*, therefore, appear as the entire content of the "hot-spot". For example, we could introduce a reference to the groff web site, in which the actual URI is concealed, by using mark up such as:–

```
For example, we could introduce a reference to
.pdfhref W -D http://www.gnu.org/software/groff -A , the groff web site
in which the actual URI is concealed,
```

Alternatively, to refer the reader to the groff web site, making it obvious that the appropriate URI is http://www.gnu.org/software/groff, the requisite mark up might be:–

```
to refer the reader to the groff web site,
making it obvious that the appropriate URI is
.pdfhref W -A , http://www.gnu.org/software/groff
the requisite mark up might be:\(en
```

2.5.5. Establishing a Format for References

There are two principal aspects to be addressed, when defining the format to be used when displaying references. Firstly, it is desirable to provide a visual cue, to indicate that the text describing the reference is imbued with special properties — it is dynamically linked to the reference destination — and secondly, the textual content should describe where the link leads, and ideally, it should also describe the content of the reference destination.

The visual cue, that a text region defines a dynamically linked reference, is most commonly provided by printing the text within the active region in a distinctive colour. This technique will be employed automatically by the `pdfhref` macro — see section 2.5.5.1, “Using Colour to Demarcate Link Regions” — unless the user specifically chooses to adopt, and implement, some alternative strategy.

2.5.5.1. Using Colour to Demarcate Link Regions

Typically, when a PDF document contains *active* references to other locations, either within the same document, or even in other documents, or on the World Wide Web, it is usually desirable to make the regions where these active links are placed stand out from the surrounding text.

2.5.5.2. Specifying Reference Text Explicitly

2.5.5.3. Using Automatically Formatted Reference Text

2.5.5.4. Customising Automatically Formatted Reference Text

It is incumbent on the user, if employing automatic formatting of the displayed reference, (see section 2.5.5, “Establishing a Format for References”), to ensure that an appropriate reference definition is created for the reference destination, and is included in the reference map for the document in which the reference will appear; thus, it may be easiest to *always* use manual formatting for external references.

2.5.6. Problematic Links

Irrespective of whether a `pdfhref` reference is placed using the “**`L`**” operator, or the “**`W`**” operator, there may be occasions when the resulting link does function as expected. A number of scenarios, which are known to be troublesome, are described below.

2.5.6.1. Links with a Page Transition in the Active Region

When a link is placed near the bottom of a page, it is possible that its active region, or “hot-spot”, may extend on to the next page. In this situation, a page trap macro is required to intercept the page transition, and to restart the mapping of the “hot-spot” boundary on the new page.

The `pdfmark` macro package includes a suitable page trap macro, to satisfy this requirement. However, to avoid pre-empting any other requirement the user may have for a page transition trap, this is *not* installed as an active page trap, unless explicitly requested by the user.

To enable proper handling of page transitions, which occur within the active regions of reference links, the user should:–

1. Define a page transition macro, to provide whatever features may be required, when a page transition occurs — e.g. printing footnotes, adding page footers and headers, etc. This macro should end by setting the output position at the correct vertical page offset, where the printing of running text is to restart, following the page transition.
2. Plant a trap to invoke this macro, at the appropriate vertical position marking the end of normal running text on each page.
3. Initialise the `pdfhref` hook into this page transition trap, by invoking

   `pdfhref I -PT <`*`macro-name`*`>`

   where **`<`*`macro-name`*`>`** is the name of the user supplied page trap macro, to ensure that `pdfhref` will correctly restart mapping of active link regions, at the start of each new page.

It may be observed that this initialisation of the `pdfhref` page transition hook is, typically, required only once *before* document formatting begins. Users of document formatting macro packages may reasonably expect that this initialisation should be performed by the macro package itself. Thus, writers of such macro packages which include `pdfmark` bindings, should provide appropriate initialisation, so relieving the end user of this responsibility. The following example, abstracted from the sample `ms` binding package, `spdf.tmac`, illustrates how this may be accomplished:–

```
.\" groff "ms" provides the "pg@bottom" macro, which has already
.\" been installed as a page transition trap.  To ensure proper
.\" mapping of "pdfhref" links which overflow the bottom of any
.\" page, we need to install the "pdfhref" page transition hook,
.\" as an addendum to this macro.
.
.pdfhref I -PT pg@bottom
```

2.6. Annotating a PDF Document using Pop-Up Notes

2.7. Synchronising Output and `pdfmark` Contexts

It has been noted previously, that the `pdfview` macro, (see section 2.2, “Selecting an Initial Document View”), the `pdfinfo` macro, (see section 2.3, “Adding Document Identification Meta-Data”), and the `pdfhref` macro, when used to create a document outline, (see section 2.4, “Creating a Document Outline”), do not immediately write their `pdfmark` output to the PostScript® data stream; instead, they cache their output, in a `groff` diversion, in the case of the `pdfview` and `pdfinfo` macros, or in an ordered collection of strings and numeric registers, in the case of the document outline, until a more appropriate time for copying it out. In the case of `pdfview` and `pdfinfo` “meta-data”, this “more appropriate time” is explicitly chosen by the user; in the case of document outline data, *some* cached data may be implicitly written out as the document outline is compiled, but there will *always* be some remaining data, which must be explicitly flushed out, before the `groff` formatting process is allowed to complete.

To allow the user to choose when cached `pdfmark` data is to be flushed to the output stream, the `pdfmark` macro package provides the `pdfsync` macro, (to synchronise the cache and output states). In its simplest form, it is invoked without arguments, i.e.

`.pdfsync`

This form of invocation ensures that *both* the “meta-data cache”, containing `pdfview` and `pdfinfo` data, *and* the “outline cache”, containing any previously uncommitted document outline data, are flushed; ideally, this should be included in a `groff` “end macro”, to ensure that *both* caches are flushed, before `groff` terminates.

Occasionally, it may be desirable to flush either the “meta-data cache”, without affecting the “outline cache”, or vice-versa, at a user specified time, prior to reaching the end of the document. This may be accomplished, by invoking the `pdfsync` macro with an argument, i.e.

`.pdfsync M`

to flush only the “meta-data cache”, or

```
.pdfsync O
```

to flush only the “outline cache”.

The “meta-data cache” can normally be safely flushed in this manner, at any time *after* output of the first page has started; (it may cause formatting problems, most notably the appearance of unwanted white space, if flushed earlier, or indeed, if flushed immediately after a page transition, but before the output of the content on the new page has commenced). Caution is required, however, when explicitly flushing the “outline cache”, since if the outline is to be subsequently extended, then the first outline entry after flushing *must* be specified at level 1. Nevertheless, such explicit flushing may occasionally be necessary; for example, the `TC` macro in the `spdf.tmac` package, (see section 3.1, “Using `pdfmark` Macros with the `ms` Macro Package”), invokes “`.pdfsync O`” to ensure that the outline for the “body” section of the document is terminated, *before* it commences the formatting of the table of contents section.

3. PDF Document Layout

The pdfmark macros described in the preceding section, (see section 2, “Exploiting PDF Document Features”), provide no inherent document formatting capability of their own. However, they may be used in conjunction with any other groff macro package of the user’s choice, to add such capability.

In preparing this document, the standard ms macro package, supplied as a component of the GNU Troff distribution, has been employed. To facilitate the use of the pdfmark macros with the ms macros, a binding macro package, spdf.tmac, has been created. The use of this binding macro package is described in the following section, (see section 3.1, “Using pdfmark Macros with the ms Macro Package”); it may also serve as an example to users of other standard groff macro packages, as to how the pdfmark macros may be employed with their chosen primary macro package.

3.1. Using pdfmark Macros with the ms Macro Package

The use of the binding macro package, spdf.tmac, allows for the use of the pdfmark macros in conjunction with the ms macros, simply by issuing a groff command of the form

```
groff -Tps -mspdf [-options ...] file(s) ...
```

(Or use the PDF post-processor to avoid using ghostscript, -Tpdf).

When using the spdf.tmac package, the groff input files may be marked up using any of the standard ms macros to specify document formatting, while PDF features may be added, using any of the pdfmark macros described previously, (see section 2, “Exploiting PDF Document Features”). Additionally, spdf.tmac defines a number of convenient extensions to the ms macro set, to better accomodate the use of PDF features within the ms formatting framework, and to address a number of ms document layout issues, which require special handling when producing PDF documents. These additional macros, and the issues they are intended to address, are described below.

3.1.1. ms Section Headings in PDF Documents

Traditionally, ms provides the NH and SH macros, to specify section headings. However, there is no standard mechanism for generating a table of contents entry based on the text of the section heading; neither is there any recognised standard method for establishing a cross reference link to the section.

To address this ms limitation, spdf.tmac defines the XN macro, (see section 3.1.1.1, “The XN Macro”), to be used in conjunction with the NH macro.

3.1.1.1. The XN Macro

4. The PDF Publishing Process

4.1. Resolving Cross References

4.1.1. Creating a Document Reference Map

4.1.2. Deploying a Document Reference Map