## NAME

groff_pdfhref − place hypertext references in PDF files created by groff

## DESCRIPTION

This manual page describes the capabilities, and the usage, of the **pdfhref** macro, which is provided by the *groff-pdfmark* macro package. The **pdfhref** macro offers *ten* distinct capabilities, each of which is identified by passing a single-character "*opcode*", as the first argument, namely:

**.pdfhref O** Create a *document outline* reference to a PDF *bookmark*, which is placed at the current location within the document body.

**.pdfhref M** Place a named reference mark, at the current location within the document body.

**.pdfhref D** Add an explicit entry to a document's *reference dictionary*.

**.pdfhref N** Manage a stack of arbitrarily named *local* "namespaces", which may be used to differentiate similarly named reference marks in *external* document files, so avoiding potential reference name collisions, and facilitating the specification of links to such external reference marks.

**.pdfhref L** Create a dynamic link to a named reference mark, within the same, or another, PDF document.

**.pdfhref W** Create a dynamic link to an internet (or similar) resource.

**.pdfhref F** Specify a macro for formatting the representation of dynamic links.

**.pdfhref K** Customize the set of keywords, which may be interpreted by the formatting macro, to describe dynamic link representations.

**.pdfhref I** Initialize a handler macro, to support processing of **pdfhref** features; (currently only supports initialization of a page trap handler, for correctly mapping the page co-ordinates of dynamic links which span page breaks).

**.pdfhref Z** Identify the page co-ordinates, for the starting and ending positions, alternately, which are required to map the bounds of active regions, to be associated with any dynamic links created by the **.pdfhref L**, and **.pdfhref W** operations.

Each of these capabilities is described, together with a detailed synopsis of the corresponding **pdfhref** macro calling semantics, in the **USAGE** section, which follows.

## USAGE

It is *strongly* recommended that **pdfroff**(1) is used, in preference to **groff**(1) itself, when formatting any document which uses the **pdfhref** macro; this will *automatically* take care of the multiple-pass **groff**(1) processing, which is normally required to correctly resolve **pdfhref** references.

The **pdfhref** macro is made available when the **groff_pdfmark**(7) macro package is loaded; this *may* be achieved by specifying the requirement on the command line:

**pdfroff −mpdfmark** [*option* ...] [*file* ...]
**pdfroff −m pdfmark** [*option* ...] [*file* ...]

However, it is often more convenient to have it loaded implicitly, by some other major document layout macro package, using the request:

**.mso pdfmark.tmac**

The capabilities of the **pdfhref** macro, as summarized in the preceding **DESCRIPTION** section, are documented in detail, below.

### Creating a Document Outline

When **pdfhref** is invoked by a macro call of the form:

**.pdfhref O** [**−T** <*tag*> | **−N** <*refname*>] <*level*> <*descriptive text* ...>

this operation is an *exact* analogue of the **pdfbookmark** macro, as described in **groff_pdfmark**(7); it creates a named reference mark, (either using the optional <*refname*> argument as an explicit name, or using an internally generated, serialized name), at the current output location within the document text, and in-

serts an entry, comprising the specified <*descriptive text* ...>, and referring to this named location, into the document outline *cache*, at the nesting level specified by the mandatory <*level*> argument.

If the **−N** <*refname*> option is specified, (noting that the <*refname*> argument is mandatory, and the space separating it from the **−N** flag is required), the <*refname*> argument is used as an explicit *bookmark* name; thus, the internal generation of a suitable name is not required, and is suppressed.

Conversely, if the **−T** <*tag*> option is specified, (note that the <*tag*> argument is mandatory, and the space separating it from the **−T** flag is required), the internally generated *bookmark* name is qualified, by appending the value of the <*tag*> argument, as a suffix to the generated name. This option is useful when two, or more separately formatted PDF document files are to be combined, to create a single finished document, *without* introducing collisions between internally generated *bookmark* names, since any *untagged* names may be duplicated in the separately formatted parts.

It may be inferred, from the foregoing, that the effects of the **−N** <*refname*>, and the **−T** <*tag*> options are mutually exclusive — the latter is effective *only* when the reference mark name is internally generated, which is not the case when the former is in effect. Although it may seem to be an error, to specify both together, this error will not be caught; the effect of the **−N** <*refname*> option will prevail, and the **−T** <*tag*> option will simply be silently ignored.

After each invocation of the **.pdfhref O** macro, (or the completely analogous **.pdfbookmark** macro), the assigned *bookmark* name, (whether explicitly specified, or internally generated, and, in the latter case, including any <*tag*> suffix which may have been specified), is stored in the **groff**(7) string object named **PDFBOOKMARK.NAME**; the value of this may then be saved, for use in subsequent *backward* references to the marked location, (for example, using the **.pdfhref L** operation, as described below). This particular feature will be of little use for *forward* references, (since the reference mark's name will be unknown before the *bookmark* has been placed), or indeed, for *explicitly* named *bookmarks*, (because the name is already known, *without* any need to inspect the value of **PDFBOOKMARK.NAME**); however, in any case where it may be found to be useful, it is important to remember that the value of **PDFBOOKMARK.NAME** is *volatile* — it will be overwritten by *any* subsequent invocation of **.pdfhref O**, or of **.pdfbookmark** — so it *must* be copied to an alternatively named **groff**(7) string object, of the user's choice, *before* it is overwritten.

It may be noted that document outline entries are, initially, stored in a local cache, rather than writing them *directly* into the document outline itself. This is necessary, because the number of directly nested child entries, at each nesting level, *must* be known at the time when each outline entry is actually written, but it isn't known until the *next sibling* entry is to be cached. Thus, the cache *must* always lead the actually committed document outline, by at least one entry at every unfinished nesting level. Consequently, there will usually be uncommitted content in the outline cache, at the end of input processing; this must be flushed, by invoking the complementary **.pdfsync O** macro, from **groff_pdfmark**(7), (or even, just the generic **.pdfsync** macro), after *all* input has been processed, (typically, from within an end-of-input trap).

### Creating Reference Marks and Links

The **pdfhref** macro supports three distinct operations, for placing reference marks within PDF documents, and defining active links to them:

**.pdfhref D** **−N** <*dest-name*> [<*keyword*> <*value*>] ... [*descriptive text* ...]
**.pdfhref M** [**−N** <*dest-name*>] [**−E**] [**−X**] [**−−**] [*descriptive text* ...]
**.pdfhref L** [**−F** <*file-name*> [**−DF** <*dos-file*>] [**−MF** <*mac-file*>] [**−UF** <*unix-file*>] [**−WF** <*win-file*>]] [**−D** <*dest-name*>] [**−P** <*prefix-text*>] [**−A** <*affix-text*>] [**−X**] [**−−**] [*descriptive text* ...]

The "**.pdfhref M** ..." form of this macro call creates a named reference mark, at the current output position within the running text of the document, and inserts a corresponding entry into the document's reference dictionary; the content of the reference dictionary entry is derived from the macro arguments, as directed by the **PDFHREF.INFO** template, (see section **CONTROL VARIABLES**, below).

On the other hand, the "**.pdfhref D** ..." form of the call simply inserts a record, with its content as specified *directly* by the macro arguments, into the reference dictionary.

Finally, the "**.pdfhref L** ..." form of the macro call creates a link to a named reference mark. This mark would typically be created by a macro call of the "**.pdfhref M** ..." form; it may be placed either before,

or after the link, in *the same* document file, or even in *a different* document file, (in which case, a reference dictionary entry created by the "**.pdfhref D** ..." form of the call may be useful).

In all of these call forms, when an *explicit* "*dest-name*" specification is provided, (in the form of a "**−D** <*dest-name*>", or a "**−N** <*dest-name*>" specification, as appropriate), the specified "*dest-name*" serves *both* as the reference mark name, *and* as the record key, for lookup of the associated reference dictionary entry. In those cases where an explicit "*dest-name*" specification is optional, and is omitted, the first space-delimited word of the "*descriptive text* ..." argument list, (which *must* be specified in any such case), is interpreted as an *implicit* specification of "*dest-name*", and is used in its stead.

In all cases, the "**−−**" flag may be specified, to indicate that all arguments which follow are to be interpreted as "*descriptive text* ...", regardless of whether, or not, they might otherwise be interpreted as macro options.

In normal use, the "**.pdfhref M** ..." form of the **pdfhref** macro call contributes *nothing* to the running text of the document. This normal behaviour may be changed, by specification of the optional "**−E**" flag; when this is specified, in addition to the normal behaviour, the content of the "*descriptive text* ..." arguments will be copied into the running text, *immediately following* the position of the named reference mark, which the call creates.

Also in normal use, when "**.pdfhref M** ..." is executed, it creates a reference dictionary entry *internally*, within **groff**(7) string space *only*; the string name is derived from the reference mark name, and serves as the dictionary lookup key. To make any reference dictionary entry *externally* visible, the "**−X**" flag may be specified; this causes the "**.pdfhref M** ..." macro to "export" the reference mark name, and its associated reference dictionary record content, through the standard error output stream, whence it will be recognized by **pdfroff**(1), and redirected, in the form of a corresponding "**.pdfhref D** ..." macro call, into a (nominally temporary) reference dictionary file; (the **pdfroff**(1) "*−−reference−dictionary*=<*file-name*>" option may be specified, to extend the life of this file beyond the lifetime of the **pdfroff**(1) process which creates it).

Like the "**.pdfhref M** ..." form, the "**.pdfhref D** ..." form of the **pdfhref** macro call causes insertion of a named destination record into the *internal* reference dictionary. As with the "**.pdfhref M** ..." form, the name of the **groff**(7) string variable, in which the dictionary record is stored, and of which the name also serves as the dictionary lookup key, is taken from the "**−D** <*dest-name*>" argument specification, (which is *required*, in any "**.pdfhref D** ..." macro call); unlike the "**.pdfhref M** ..." form, however, the content of the reference dictionary record is *not* controlled by the **PDFHREF.INFO** template; it is, rather, determined *directly*, from the sequence of "<*keyword*> <*value*>" pairs, and "*descriptive text* ..." arguments, as specified in the "**.pdfhref D** ..." call. Furthermore, since the intent of "**.pdfhref D** ..." is, primarily, to *import* reference dictionary entries, which had previously been *exported* by a "**.pdfhref M** ..." call, (often from within a separate document), there is no equivalent to of the "**.pdfhref M −X**" flag, for use with "**.pdfhref D** ...".

Unlike the "**.pdfhref M** ...", and the "**.pdfhref D** ..." operations, the "**.pdfhref L** ..." form of the **pdfhref** macro call does not *create* reference dictionary entries, but it may need to *consult* them; the precise behaviour is determined by the logic incorporated within the following processing sequence:

- If an *explicit* "**−D** <*dest-name*>" argument is specified, then "*dest-name*" identifies the named reference mark to which the link refers; in this case, "*descriptive text* ..." arguments become optional, and may be omitted, or specified, at the user's discretion.
- Conversely, when there is no "**−D** <*dest-name*>" argument specification, then the first of the "*descriptive text* ..." arguments, (at least one of which is *required*, in this case), will be interpreted as an *implicit* specification of "*dest-name*", and thus *this* identifies the named reference mark to which the link refers.
- If there are no "*descriptive text* ..." arguments specified, (which implies that there *must* be a "**−D** <*dest-name*>" specification), then a reference dictionary lookup, keyed on "*dest-name*", is performed, and the content of the associated record is formatted into the active link region, under the control of either a user-specified, or an internal default formatting macro.
- If any "*descriptive text* ..." arguments *are* specified, together with the "**−X**" flag, (which exhibits distinctly *different* behaviour from that of "**.pdfhref M −X** ..."), then a "*dest-name*" reference dictionary lookup is again performed; in this case, however, *only* "<*keyword*> <*value*>" pairs from

the associated dictionary record, followed by the "*descriptive text* ..." arguments from the "**.pdfhref L −X** ..." call, are passed to the link formatting macro.

- Finally, if any "*descriptive text* ..." arguments are specified, but the "**−X**" flag is not, then the reference dictionary is *not* consulted, and the **pdfhref** link text formatting macro is *not* called; the content of the "*descriptive text* ..." arguments is simply written, as is, into the active link region, with normal **troff**(1) formatting applied.

By default, "**.pdfhref L** ..." will assume that all named references are to be resolved within the same document file as the link itself, *unless* the "**−F** <*file-name*>" option is specified; this causes the **pdfhref** macro to emit additional **pdfmark** code, stipulating that the named reference destination may be found in the file named by the "*file-name*" argument.

When the "**−F** <*file-name*>" option is specified, the "*file-name*" specified may not be universally suitable for use on all file systems which it is desired to support. To mitigate this limitation, it is permitted to specify any of the following additional options, together with "**−F** <*file-name*>":

**−DF** <*dos-file*>
: Specify an alternative *file-name*, which is suitable for use on MS-DOS™ file systems.

**−MF** <*mac-file*>
: Specify an alternative *file-name*, which is suitable for use on Apple Macintosh™ file systems.

**−UF** <*unix-file*>
: Specify an alternative *file-name*, which is suitable for use on Unix file systems.

**−WF** <*dos-file*>
: Specify an alternative *file-name*, which is suitable for use on MS-Windows™ file systems.

The two remaining options, which are supported by "**.pdfhref L** ..." macro calls, provide a mechanism for specifying additional running-text content, (typically punctuation), which is to be printed immediately *before*, or *after* the associated link, *without* any intervening white space, but *not* within its active region; the "**−P** <*prefix-text*>" option places "*prefix-text*" as such additional content, *before* the link, while "**−A** <*affix-text*>" places "*affix-text*" in similar fashion, *after* the link.

**Linking to Reference Marks in External Files**

Although the "**.pdfhref L** ..." macro call supports linking to external reference marks, by explicitly adding the "**−F** <*file-name*>" option, (possibly augmented by any one, or more, of its alternative file naming options), the macro also supports an *implicit* mechanism for specification of the external file association, using **pdfhref** "namespaces". Such namespaces are created, and managed, by a macro call conforming to the synopsis:

**.pdfhref N** [[**−F** <*file-name*> [**−DF** <*dos-file*>] [**−MF** <*mac-file*>] [**−UF** <*unix-file*>] [**−WF** <*win-file*>]] [**−D** <*dictionary-file*>] <*namespace-id*>]

Notice that, when invoked in the "**.pdfhref N** ..." form, the **pdfhref** macro may be called with, or without further arguments; however, if *any* such arguments *are* specified, then the "<*namespace-id*>" argument *must* be present.

When called *with* arguments, the specified "*namespace-id*" is pushed on to an internal namespace stack, making it active. If this is the *first* "**.pdfhref N** ..." call which specifies this "*namespace-id*", then a namespace specification record is created, and any "**−F** <*file-name*>" argument which may also be specified, together with any accompanying alternative file name specifications, is recorded within it.

Conversely, when called *without* arguments, the currently active "*namespace-id*", (if any), is popped off the namespace stack, thus deactivating it, and reactivating any previously active namespace.

When called *with* arguments, if a "**−D** <*dictionary-file*>" argument is included amongst those specified, then the content of the specified reference dictionary file is read, by means of a **groff**(7) **.so** request, and any "**.pdfhref D** ..." records which are found within it are processed within the context of the specified "*namespace-id*". Then, after this reference dictionary file has been read, in its entirety, this "*namespace-id*" is popped off the namespace stack, thus deactivating it, and reactivating any previously active namespace.

On the other hand, if called *with* arguments, but *without* specifying a "**−D** <*dictionary-file*>" argument, then the specified "*namespace-id*" remains in effect, after completion of the call, until the user issues a

further "**.pdfhref N**" call, *without* arguments. This allows the user to load a reference dictionary file manually, and/or specify any desired "**.pdfhref D** ..." records manually, before the call to deactivate the namespace is issued.

When any "**.pdfhref D** ..." record is processed, within the context of an active "**.pdfhref N** ..." namespace, and assuming the the "**.pdfhref D** ..." call includes a "**−N** <*ref-name*>" argument, (as is normally the case), then the specified "<*ref-name*>", as recorded in the *internal reference dictionary*, will be qualified by addition of the "<*namespace-id*>" from the namespace-activating "**.pdfhref N** ..." macro call, as a "<*ref-name*>" prefix, causing the "<*ref-name*>" key to be *recorded* as "<*namespace-id*>::<*ref-name*>"; consequently, this becomes the *effective* "<*ref-name*>" by which any "**.pdfhref L** ..." macro call should refer to this *external* reference mark, within the specification of its "**−D** <*ref-name*>" argument.

The effect may be illustrated by consideration of the following adaptation of the syntactic prototype for the "**.pdfhref L** ..." macro call:

**.pdfhref L** [**−D** <*namespace-id*>::<*dest-name*>] [**−P** <*prefix-text*>] [**−A** <*affix-text*>] [**−X**] [**−−**] [*descriptive text ...*]

This formal prototype, for the "**.pdfhref L** ..." macro call, is applicable *exclusively* when specifying a link to a reference mark, with its actual name specified by "<*ref-name*>", when its *internal* reference dictionary entry has been encapsulated within the namespace specified by "<*namespace-id*>"; (the double colon punctuation, inspired by the double colon operator of the C++ programming language, is required). This facility for encapsulation of external file reference data within locally defined, user specified namespaces, may offer certain advantages; most notably:

- The syntax of the "**.pdfhref L** ..." macro call itself is somewhat simplified, particularly with respect to the specification of the "**−F** <*file-name*>" argument, and any associated alternative file name arguments.
- The "<*namespace-id*>" prefix, within the "**−D** <*namespace-id*>::<*ref-name*>" argument, serves as a differentiator for matching "*ref-name*" specifications within distinct files; without any such differentiation, processing of any "**.pdfhref D** ..." record, with an included "**−N** <*ref-name*>" specification which matches any existing "*ref-name*" entry in the internal reference dictionary, would clobber that existing reference dictionary entry.

Notice that, when this modified form of the "**.pdfhref L** ..." macro call is used, then *neither* a "**−F** <*file-name*>" argument, which would normally be required for references to external files, *nor* any of its associated alternative file naming arguments, is specified; rather, these are inferred from their specification as arguments to the "**.pdfhref N** ..." macro call, which defines the namespace in which the corresponding reference dictionary entry has been encapsulated, and whence they are automatically propagated to any "**.pdfhref L** ..." macro call, in which the "**−D** <*ref-name*>" argument is specified in the modified, namespace-qualified "**−D** <*namespace-id*>::<*ref-name*>" format.

**Linking to Internet Resources**

Links to internet, and similar resources, are supported by a single **pdfhref** macro operation; however, this operation may be invoked using either one of two distinct forms of the macro call:

**.pdfhref W** [**−P** <*prefix-text*>] [**−A** <*affix-text*>] [**−−**] <*URI*>
**.pdfhref W** **−D** <*URI*> [**−P** <*prefix-text*>] [**−A** <*affix-text*>] [**−−**] *descriptive text ...*

In the first of these, the uniform resource identifier (URI) takes the place of any "*descriptive text ...*" arguments; it *must* be a verbatim representation of the URI, and will appear as such within the active region of the link, in the published PDF document.

In the second form, the URI is explicitly specified by way of the "**−D** <*URI*>" option. The content of the active region of the link will then be the "*descriptive text ...*" arguments *only*; this will *not* include any representation of the URI, *unless* it is duplicated within these arguments.

In each of these cases, the "**−−**" flag, and each of the "**−P** <*prefix-text*>" and "**−A** <*affix-text*>" options exhibit *identical* behaviour to that when they are used in the "**.pdfhref L** ..." form of the **pdfhref** macro call.

**Link Format Customization**

When the "**.pdfhref L** ..." form of the **pdfhref** macro is called, either *without* specification of any "*descriptive text ...*" arguments, or with the effect of the "**−X**" flag applied to any such arguments, then

the content which is placed within the active region of the resultant link will be determined by calling an auxiliary formatting macro.

Although the *groff-pdfmark* macro package does provide an internal default implementation of a suitable formatting macro, the user may choose to supply an alternative; this may be assigned, as required, by a **pdfhref** macro call of the form:

**.pdfhref F** [<*macro-name*>]

When used in this form, the optional "<*macro-name*>" argument, if specified, should name a user-provided macro, which will perform the auxiliary formatting operation for subsequent "**.pdfhref L** ..." calls; if "<*macro-name*>" is omitted, the internal default macro will be reassigned, as the handler for any subsequent "**.pdfhref L** ..." auxiliary formatting operations.

When "**.pdfhref L** ..." calls an auxilliary formatting macro, regardless of whether it is a user-provided macro, or the internal default macro which is called, the content of the reference dictionary record, which is associated with the specified link destination, is passed as auxiliary formatting macro arguments, while any "*descriptive text* ..." arguments which were specified for the "**.pdfhref L** ..." macro call itself, are passed by assigning them to a string variable named **PDFHREF.DESC**, (which remains undefined, if no "*descriptive text* ..." arguments have been specified). Then, after processing its arguments, together with any additional content passed in **PDFHREF.DESC**, the auxiliary formatting macro ultimately returns content for interpolation into the active region of the generated link, by assigning it to a string variable named **PDFHREF.TEXT**.

There is just one requirement imposed on any user-provided auxiliary formatting macro: it *must* assign a string value to **PDFHREF.TEXT**, for interpolation into the active region of each generated link. On entry, it may rely on its arguments representing the content of the reference dictionary record for the link which is to be generated, and on the value of the **PDFHREF.DESC** string variable having been assigned to represent any "*descriptive text* ..." arguments from the originating "**.pdfhref L** ..." macro call, (or on this string variable being undefined, if there were no such arguments); however, whatever use the formatting macro may then make of this input information is entirely at the discretion of the macro writer, who may wish to consider the following sequence of operations, as implemented for the internal default formatting macro:

- The **PDFHREF.TEXT** string variable is assigned an initial value which is equal to that of the **PDFHREF.PREFIX** template string variable, if that is defined, (as it is, by default); otherwise **PDFHREF.TEXT** is initialized as an empty string.
- The macro arguments are processed in sequence, from left to right. If the first argument is an *exact* match for any registered formatting keyword, (see the **Link Format Template Registration**, section below), then the second argument is interpolated into the formatting template which is associated with that keyword, and the result is appended, with a single intervening space, to the currently accumulated value of **PDFHREF.TEXT**. The macro arguments are then shifted to discard the first two, (thus promoting the third to become the new first argument), and this step is repeated. This cycle continues until all remaiming arguments have been exhausted, or until the first argument, (either on entry, or after promotion), does *not* match any registered formatting keyword; thereafter, the cycle is terminated, and processing proceeds *immediately* to the following step.
- If **PDFHREF.DESC** was *not* defined, on entry to the macro, it is defined now, with the entire sequence of remaining macro arguments, separated by spaces, being assigned as its value; otherwise, **PDFHREF.DESC** retains its original value, as assigned by "**.pdfhref L** ..." before calling the auxiliary formatting macro.
- The deferred expansion of **PDFHREF.DESC** is now appended to the already accumulated value of **PDFHREF.TEXT**, preceded by one intervening space.
- Finally, the accumulated value of **PDFHREF.TEXT** is reinterpreted, reassigning the resultant text back to **PDFHREF.TEXT** itself, but *without* any leading spaces which may have been accumulated; this reassigned value is then returned to the calling "**.pdfhref L** ..." instance.

After calling the default auxiliary formatting macro, (which, as may seen from the foregoing sequence of operations, is fully conformant with the requirements for such a macro), or any user-provided alternative, the "**.pdfhref L** ..." handler interpolates the returned value of **PDFHREF.TEXT** into the active region of the specified link, (bracketted by any "*prefix-text*", and "*affix-text*", which may be specified); it then

deletes the **PDFHREF.DESC** (if it is defined), and **PDFHREF.TEXT** string variables.

**Link Format Template Registration**

When "**.pdfhref L** ..." calls the default *groff-pdfmark* auxiliary formatting macro, to lay out text within any active link region, it interpolates a sequence of "<*keyword*> <*value*>" pairs, as described in the **Link Format Customization** section, above; (a user-provided auxiliary formatting macro *may*, or may not exhibit similar behaviour, at the discretion of the macro writer).

A small, but extensible, vocabulary of keywords, which become subject to such interpolation, is built into the *groff-pdfmark* package; this vocabulary may be extended, by invocation of:

**.pdfhref K** <*keyword*> <*format-name*> [<*keyword*> <*format-name*>] ...

Each specified "<*keyword*>" *must* be valid as a **groff**(7) identifier; it is registered such that it may be recognized by an auxiliary link formatting macro, and is associated with a **groff**(7) string, named as specified by the immediately following "<*format-name*>" argument, (which *must*, therefore, also be valid as a **groff**(7) identifier), as its corresponding formatting template; this template will subsequently be interpolated, as modified by the accompanying "<*value*>" argument, into the active region of a **pdfhref** link, when the associated "<*keyword*>" is encountered within the sequence of arguments which have been passed to a "**.pdfhref L** ..." auxiliary formatting macro.

For each "<*format-name*>" specified, the user *must* ensure that a suitable **groff**(7) string definition is also provided, for use as the requisite formatting template.

The initially defined default "<*keyword*>" vocabulary comprises the following entries:

| Keyword | Format Name | Template |
|---|---|---|
| | **PDFHREF.PREFIX** | *see* |
| *file* | **PDFHREF.FILEREF** | *\\$1* |
| *page* | **PDFHREF.PAGEREF** | *page \\$1* |
| *section* | **PDFHREF.SECTREF** | *section \\$1* |

In each case, the "*\\$1*" field, within the template definition, expands to the content of the "<*value*>" argument, which follows the identified "<*keyword*>", when passed as arguments to, and interpolated by, the default auxiliary formatting macro.

Among the pre-defined formatting templates, enumerated above, it may be noted that the definition of "**PDFHREF.PREFIX**" is anomalous. In fact, this particular template is associated with *neither* any "<*keyword*>", *nor* any "<*value*>" argument; within the default auxiliary formatting macro, it is *automatically* interpolated at the *beginning* of *every* active link.

Any of the formatting template strings, whether defined by default, or by the user, may be redefined by the user, to modify the appearance of formatted active links.

**Features Initialization**

There is one feature of the **pdfhref** macro, which can negatively affect the interpolation of active links, if it is mishandled — specifically, if a page break occurs within the active region of any link, special handling is required, to ensure that the region is correctly mapped.

To handle this scenario correctly, *groff-pdfmark* provides a helper macro, which *must* be called from within a page trap macro, and this *must* itself be invoked as the page break is traversed; this helper macro may be attached to an existing — possibly user-defined — page trap macro, by an initialization macro call of the form:

**.pdfhref I –PT** <*macro-name*>

It is assumed that users will have implemented their own page trap conventions, (or will be using a standard macro package which does so), and *groff-pdfmark* cannot make pre-emptive assumptions about how it should interact with those; therefore, the onus for integration of the **pdfhref** page trap handler into the user's implementation, by issuing the above initialization macro call, is placed on the user; (ideally, if using a standard macro package with *groff-pdfmark* bindings, this initialization should be performed *automatically*, within the bindings for that package).

When calling the *groff-pdfmark* page trap initialization macro, the "<*macro-name*>" argument is expected to refer to an existing macro, which has already been installed as a page trap handler; the initialization call then simply appends a hook to the end of that macro, *without* verification that the macro will subsequently

be invoked by a page trap.

If no macro, with the name specified by "<*macro-name*>" exists, at the time when the initialization macro is called, then a macro of that name will be defined, *and* registered as a trap, at the top of each page; this may not achieve the desired effect.

If no "<*macro-name*>" argument is specified, when the initialization macro is called, (which is technically an error, but it is not considered fatal), then the *groff-pdfmark* page trap hook will, itself, be installed as a free-standing trap, at the top of each page; once again, the effect may not be as intended.

**Mapping of Active Link Regions**

When placing reference links, using either the "**.pdfhref L** ...", or the "**.pdfhref W** ..." form of the **pdfhref** macro, the text, which may be *perceived* to define each link, is not actually significant, nor indeed, is it even incorporated within the *pdfmark* encoding of the link specification. Rather, together with a pointer to the named destination, each link specification comprises a collection of one or more rectangular regions, each of which is defined by page co-ordinates, and corresponding page number.

Each aggregate collection of such linked page regions is combined, in order to encompass all of the text which is associated with the link; notwithstanding that, if this text spans more than a single output line, it is likely that the shape of the aggregate region will be irregular, and thus, it will require more than a single rectangle to cover it, the entire aggregate is mapped out by *just one* pair of invocations of the **pdfhref** macro, with each call within each pair having the form:

**.pdfhref Z** <*page*> <*xpos*> <*ypos*>

Within each pair of "**.pdfhref Z** ..." calls, the first specifies the page number, and page co-ordinates, where the text which is to be encapsulated within the link region *starts*, while the second specifies tha page number, and co-ordinates where it *ends*. From this information, the "**.pdfhref L** ..." and "**.pdfhref W** ..." macros are able to compute the number, and placement of sufficient rectangles to encompass the text, and so to emit the *pdfmark* code to create the active link region.

It may appear that determination of page numbers, and co-ordinates, to map out link regions, and to specify the corresponding sequence of "**.pdfhref Z** ..." calls, would be an onerous task for any document author; fortunately, this is not necessary, because **pdfroff**(1) performs it *automatically*, thus:

- During the initial **pdfroff**(1) document analysis phase of PDF production, the "**.pdfhref L** ..." and "**.pdfhref W** ..." macros insert special marker characters, at the beginning, and at the end, of each run of text which is to be encapsulated within a link; the insertion of these markers produces a sequence of location information, on the *standard error* data stream, whence **pdfroff**(1) captures it, and generates a corresponding sequence of "**.pdfhref Z** ..." macro calls.
- During the subsequent PDF document publication phase, **pdfroff**(1) injects the generated sequence of "**.pdfhref Z** ..." macro calls at the beginning of the document input stream, storeing their data content internally, in **groff**(7) string space, whence subsequent "**.pdfhref L** ..." and "**.pdfhref W** ..." macro calls may reinterpret it, sequentially, and emit *pdfmark* code to create the actual links, while ultimately discarding each such reinterpreted record, as it is processed.

There is an unfortunate side effect of this technique for computation of link locations — the special markers, which are placed around the text associated with each link, although each is of zero width, do, of necessity, incorporate a visible component, and this is unwanted in the finished PDF document. To avoid this, these special markers are *not* inserted around any text for which a pair of "**.pdfhref Z** ..." entries has already been recorded, in **groff**(7) string space; thus, since **pdfroff**(1) *never* injects "**.pdfhref Z** ..." macro calls during its initial document analysis phase, and *always* injects them during the final publication phase, markers are *always* inserted while performing document analysis, but are omitted while producing the final publication, the unwanted visual artefacts are avoided in the finally published document.

## CONTROL VARIABLES

The behaviour of the **pdfhref** macro may be influenced by the values which are assigned to a collection of **groff**(7) string variables, and numeric registers.

The effective string variables are:

**PDFHREF.BORDER**

This string variable specifies, as a minimum, a space separated numeric triplet, optionally followed by further space separated *pdfmark array*, the aggregate effect of which is to define the properties of the border decoration around **pdfhref** active link regions.

Unlike the Adobe® default, which places a one pixel wide solid border around link regions, the default setting for **PDFHREF.BORDER** *disables* border decoration; the Adobe default behaviour may be achieved by assigning a triplet of "`0 0 1`" to **PDFHREF.BORDER**.

**PDFHREF.COLOR**
**PDFHREF.COLOUR**

Each of this (normally aliased) pair of string variables specifies a space separated triplet of floating point values, each in the range *0.0* to *1.0*, representing the colour, in the RGB colour-space, which is to be used for *decoration* of, (but *not* for the text within), **pdfhref** active link regions on the output page.

The American English spelling of **PDFHREF.COLOR**, and its World English counterpart, **PDFHREF.COLOUR**, are interpreted synonymously; however, if both are defined, *and* their values are *distinct*, precedence will be given to **PDFHREF.COLOUR**, at each single point of use, after which the two will be resynchronized, by binding them as aliases for the triplet value specified by **PDFHREF.COLOR**.

A practical reason for the specification of distinct values for **PDFHREF.COLOUR**, and **PDFHREF.COLOR**, might be to introduce a single-use change in the effective colour, which is to be applied to a particular link region. To achieve this, it is first necessary to use the **groff**(7) **rm** request, to delete any existing specification of **PDFHREF.COLOUR**, and thus break the alias with **PDFHREF.COLOR**. A new triplet may then be specified for **PDFHREF.COLOUR** itself, *before* the link is inserted into the document output stream; after the link has been formatted, the original alias with **PDFHREF.COLOR** is reinstated.

**PDFHREF.TEXT.COLOR**
**PDFHREF.TEXT.COLOUR**

Another pair of normally aliased string variables, these specify the text colour which will be used to represent active **pdfhref** links within published documents, but unlike the specifications for **PDFHREF.COLOUR** and **PDFHREF.COLOR**, these are expressed in terms of **groff**(7) colour names, (with their default being an internally defined name, which is mapped to the colour corresponding to **PDFHREF.COLOUR**).

Once again, the American and World English spellings are considered to be synonymous, with precedence being given to **PDFHREF.TEXT.COLOUR**, if the two definitions are distinct; however, in this case, the distinction is preserved until *either one* of the two distinct definitions is deleted.

If *both* definitions are deleted, the default equivalence with **PDFHREF.COLOUR** will be reinstated, when the next active link is formatted.

**PDFHREF.DESC**
**PDFHREF.TEXT**

Reserved for internal use *only*, this pair of string variables provides a channel for communication between the "**.pdfhref L** ..." handler macro, and an auxiliary formatting macro, for layout of text within the active regions of **pdfhref** links. Neither of these string variables is visible to the user, outside the scope of a user-defined auxiliary formatting macro, within which they serve the purposes described in section **Link Format Customization**, above.

**PDFHREF.HEIGHT**
This is a string variable representation of the height of a **pdfhref** active link region, where the content of the region is entirely accommodated on a single line of output text. By default, it is set equal to the nominal line spacing of the current font, (i.e. "*1.0v*"), and changing it is *not* recommended.

**PDFHREF.INFO**
This string variable defines a template, which is applied to establish the content of any **pdfhref** reference dictionary entry, which may be stored when processing a **.pdfhref M** macro call. Its default value is "*page \\n% \\$**", with the expansion of its "*\\$**" argument representing any "*descriptive text* ..." arguments, from the **.pdfhref M** macro call.

**PDFHREF.PREFIX**
This string variable specifies text, which will *always* be inserted into a **pdfhref** active link region, by the default link text formatting macro, *before* formatting any further content which is retrieved from the **pdfhref** reference dictionary; its default value is the single English word, "*see*", separated from its following content by a single space.

**PDFHREF.FILEREF**
This string variable defines a formatting template, which specifies how an external file name reference should be represented, within the content of the active region of a **pdfhref** link, when a "*file <file-name>*" specification is encountered while interpolating a reference dictionary entry. It has a default value of "*\\$1*", which results in just the "*<file-name>*" element of the reference dictionary entry being represented within the link.

**PDFHREF.PAGEREF**
This string variable serves as a template, specifying how a page number reference should be formatted, when it is encountered within a reference dictionary entry, and is to be incorporated into the text which is to be included within the active region of a **pdfhref** link. Its default value is "*page \\$1*", which simply duplicates any introductory "*page \n%*" combination, which may have been incorporated into the reference dictionary, as a result of substitution into the **PDFHREF.INFO** template.

**PDFHREF.SECTREF**
Also serving as a template, this string variable specifies how a section number reference should be formatted, when it is encountered within a reference dictionary entry, and is to be incorporated into the text which is to be included within the active region of a **pdfhref** link. Its default value is "*section \\$1*", which duplicates any introductory "*section *(SN*" combination, when this was specified in the **PDFHREF.INFO** template, for inclusion in the reference dictionary entry.

**PDFHREF.VIEW**
**PDFBOOKMARK.VIEW**
This distinct pair of string variables specify the view properties, for presentation of of the destinition when either a **pdfhref** active link, (defined by a **.pdfhref L** macro call), or a PDF outline reference, (defined by either a **.pdfhref O**, or a **.pdfbookmark** macro call), is clicked, respectively. It must be assigned a value which conforms to the array argument requirements of the */View* link annotation *pdfmark* operator, (excluding the array's enclosing brackets, but including any **grops**(1) "*u*" transformation operators, which may be necessary to convert from the *groff* page co-ordinate system to the PDF viewport mapping model); the default value, for both, is "*/FitH \\n[PDFHREF.Y] u*".

The numeric register variables are:

**PDFHREF.LEADING**
The value of this numeric register specifies the displacement of the top of any active **pdfhref** link region, *above* the topmost extent of the natural bounding box for the glyphs which are contained within it. This may be considered to represent an *adjustment* to the natural leading, which is associated with the contained glyphs, and which may already be more than the desired leading within each link region; consequently, a *negative* value may be appropriate, as reflected in the default setting, which is "*-1p*".

**PDFHREF.VIEW.LEADING**

This numeric register specifies, in **groff**(7) basic units, the vertical distance below the top of the viewport window, at which each reference mark will be placed, when it is selected to be brought into view.

Note that **PDFHREF.VIEW.LEADING** does *not* represent *true* leading, in the typographic sense; any text, which has been typeset *above* the reference mark, and which is located within **PDFHREF.VIEW.LEADING** basic units of the reference mark, *will* remain visible within the viewport.

**PDFHREF.Y**

The value of this numeric register is internally computed, when any **pdfhref** link destination reference mark is defined; its value represents the vertical distance, in **groff**(7) basic units, from the top of the output page to the location of the reference mark.

While it may be normal to refer to the value of **PDFHREF.Y**, within any redefinition of **PDFHREF.VIEW**, or of **PDFBOOKMARK.VIEW**; it is *always* recomputed *immediately before* either of these view specifications is evaluated, and it is deleted *immediately* thereafter, so it is not visible, and thus cannot serve any useful purpose, in any other context.

With the exception of those which are designated as *reserved*, or otherwise, as not having any useful effect in a user-visible scope, any of these strings, or numeric registers, may be redefined by the user, to achieve a change in **pdfhref** macro behaviour.

## FILES

*/usr/local/share/groff/site-tmac/pdfmark.tmac*

Implements the **pdfhref** macro, (among others).

## CAVEATS AND BUGS

Although the Adobe® PDF specification makes provision for linking to references by page number and co-ordinates, this mechanism is *not* supported by the **pdfhref** macro; support is provided for linking to named reference marks *only*.

When any reference mark name, which is to be used as a link destination, is implicitly derived from the initial part of "*reference text* ..." arguments, that part of those arguments *must* represent a *unique* and *valid* **groff**(7) identifier; this limitation will often be inconvenient, in which case, use of an explicitly named reference mark is recommended.

Similarly, all "<*keyword*>" and "<*format-name*>" arguments, used in the registration of link formatting templates, *must* be specified as valid **groff**(7) identifiers.

Interpolation of either a "**.pdfhref L** ..." or a "**.pdfhref W** ..." macro call, which is *followed* by a *non-breaking* **groff**(7) "**.ne**" request, and which results in relocation of the output line in which the link text appears, may disassociate the link text from its mapped active region; to avoid such anomalous behaviour, ensure that the non-breaking "**.ne**" request is placed *before* calling any macro which then has the effect — possibly indirectly — of interpolating the affected link.

Failure to initialize the *groff-pdfmark* page trap hook, by attachment to an existing page trap macro, may result in erratic behaviour.

## AUTHORS

The **pdfhref** macro is provided by the supplementary **groff_pdfmark**(7) macro package, which was written by Keith Marshall <keith@address.hidden>; formerly distributed with *GNU roff*, this is now independently maintained at, and distributed from Keith's *groff-pdfmark* project hosting web-site <https://savannah.nongnu.org/projects/groff-pdfmark/>, whence the latest version may *always* be obtained.

## SEE ALSO

**groff**(1), **grops**(1), **pdfroff**(1), **troff**(1), **groff**(7), **groff_pdfmark**(7)

More comprehensive documentation, on the use of the **pdfhref** macro may be found, in PDF format, in the reference guide "*Portable Document Format Publishing with GNU Troff*", which has also been written by Keith Marshall; the most recently published version of this guide may be read online, by following the appropriate document reference link on the *groff-pdfmark* project hosting web-site <https://savannah.nongnu.org/projects/groff-pdfmark/>, whence a copy may also be downloaded.