The l3pdftools module
temporary collection of pdf related commands
LaTeX PDF management bundle

The LaTeX Project*

Version 0.97a, released 2026-04-21

1 l3pdftools documentation

This module collects a number of candidate commands for the l3pdf module

---

`\pdf_purify:nN` `\pdf_purify:nN {⟨content⟩} {⟨tl var⟩}`

New: 2026-02-18

This command calls `\text_purify:n` for ⟨*content*⟩ and stores the result in ⟨*tl var*⟩. It performs first various setups to make the conversion suitable for a PDF context, most importantly for the bookmarks. Beside other it executes the hyperref command `\pdfstringdefPreHook`, that means that it knows the exceptions added by `\pdfstringdefDisableCommands`. It also sets `\Hy@pdfstringtrue`, this means that if the content uses `\texorpdfstring` the pdf branch is used.

At the begin the command calls the hook `pdf/purify/setup` which can be used to add more definitions and exceptions.

While the implementation is quite different the command has a similar purpose as `\pdfstringdef`. The most notable difference from a user viewpoint is that it doesn't directly encode the result as a PDF string, typically a call of `\pdf_purify:nN` should be followed with a call of `\pdf_string_from_unicode:nnN`.

`\pdf_name_from_unicode_e:n` ⋆ `\pdf_name_from_unicode_e:n {⟨content⟩}`
`\pdf_name_from_unicode_e:V` ⋆

New: 2021-02-14

This converts ⟨*content*⟩ to a format suitable for a PDF Name. The output depends on the backend: For almost all backends it will first expand the content with `\text_expand:n` and then escape it in the way needed in a PDF Name with `\str_convert_pdfname:e`, and at last prepend a slash before. Typically such names use only ascii, but non-ascii is supported, but should be utf8 encoded. For example
`\pdf_name_from_unicode_e:n {A~B\c_percent_str C\c_hash_str D€}}`
will output `/A#20B#25C#23D#E2#82#AC`.

With dvips it will expand the content with `\text_expand:n` and then wrap it in a `cvn` operation ("convert to name"). So the example above will output `(A B%C#D€) cvn` to the postscript. The content should not contain unbalanced parentheses with dvips.

*E-mail: latex-team@latex-project.org

\pdf_string_from_unicode:nnN \pdf_string_from_unicode:nnN {⟨*format*⟩} {⟨*content*⟩} ⟨*tl var*⟩
\pdf_string_from_unicode:nVN

New: 2020-07-04

This converts ⟨*content*⟩ following the rules defined by ⟨*format*⟩ and stores the result in ⟨*tlvar*⟩. The assignment is done locally. Non-ascii input should be utf8 encoded. Currently the following formats exist:

utf8/string-raw this converts with `\str_set_convert:Nnnn` into utf8/string.

utf8/string this converts into utf8/string and adds parentheses around the result.

utf8/URI-raw this converts with `\str_set_convert:Nnnn` into utf8/url and then replaces reserved and digits back from the percent encoding. Parentheses are escaped.

utf8/URI this converts into utf8/URI and adds parentheses around the result.

utf16/string-raw this converts with `\str_set_convert:Nnnn` into utf16/string.

utf16/string this converts into utf16/string and adds parentheses around the result.

utf16/hex-raw this converts into utf16/hex

utf16/hex this converts into utf16/hex and adds bracket around the result.

1.1 BDC operator / Properties resource

Entries to the /Properties dictionary in the page resources can be added with dvips only through side-effects: if a BDC-mark is created dvips/ghostscript will automatically create the necessary objects and names. To get a sensible abstraction the code does for some of the following command the same for the other backends if the core management code has been activated. This means that the behaviour of the command is different then. The `\pdf_bdcobject:..` should only be used if the management is active.

\pdf_bdc:nn
\pdf_bdc:ee

Updated: 2024-10-23

\pdf_bdc:nn {⟨*tag*⟩} {⟨*dictionary content*⟩}

This command adds a BDC marked content operator to the current page stream. ⟨*tag*⟩ is the tag of this operator (without the leading slash), ⟨*dictionary content*⟩ is the content of the second argument. With the exception of the dvips backend, the dictionary content is added inline in the stream. Such an inline BDC is typically better for ActualText additions as some PDF reader ignore entries given in properties.

\pdf_bdc_shipout:ee `\pdf_bdc_shipout:ee {⟨tag⟩} {⟨dictionary content⟩}`

New: 2023-08-18

This command adds a BDC marked content operator to the current page stream. ⟨*tag*⟩ is the tag of this operator (without the leading slash), ⟨*dictionary content*⟩ is the content of the second argument.

Differently to `\pdf_bdc:ee` the arguments are not expanded when the command is *used*, but only at *shipout*. This requires new engines which allow to use the keyword `shipout` with the primitive `\special` and `\pdfliteral`. Also similar to `\pdf_bdc:ee` the content of ⟨*dictionary content*⟩ is added inline in the stream with most engines (not on the dvips + ps2pdf route). This means that this command can also be used if such an inline dictionary is preferred.

The command requires current engines and is not defined if an too old engine is detected!

\pdf_bdcobject:nn `\pdf_bdcobject:nn {⟨tag⟩} {⟨object name⟩}`

New: 2020-07-03

This command adds a BDC marked content operator to the current page stream. ⟨*tag*⟩ is the tag of this operator (without the leading slash), ⟨*object name*⟩ is a the name of an dictionary object reserved with `\pdf_object_new:n` and filled with `\pdf_object_write:nnn` with the properties of the BDC. Reusing a predefined object can save space but the command works correctly only if the resources management has been activated and should be used only if this can be ensured.

\pdf_bdcobject:n `\pdf_bdcobject:n {⟨tag⟩}`

Updated: 2020-07-03

This command adds a BDC marked content operator to the current page stream. ⟨*tag*⟩ is the tag of this operator (without the leading slash). As object this commands uses the last anonymous dictionary object created with `\pdf_object_unnamed_write:nn`. It lies in the responsibility of the user that the last object is the wanted one. Like with `\pdf_bdcobject:nn` the command works correctly only if the resources management has been activated and should be used only if this can be ensured.

\pdf_bmc:n `\pdf_bmc:n {⟨tag⟩}`

New: 2019-10-17

This command created a BMC marked content operator. The argument is the tag without the leading slash. It can be e.g. used for simple artifact markers.

\pdf_emc: `\pdf_emc:`

New: 2019-06-30

This command closes the BDC marked content operator opened with `\pdf_bdc:nn`. It should be on the same page as the bdc-command.

```
\pdf_object_new:n     {module/objA}
\pdf_object_write:nnn {module/objA}{dict}{/Type/Artifact}
\pdf_bdc:nn {Span}{module/objA}
text
\pdf_emc:
```

2 l3pdftools implementation

1 ⟨*header⟩

```
\ProvidesExplPackage{l3pdftools}{2026-04-21}{0.97a}
  {candidate commands for l3pdf---LaTeX PDF management bundle}
⟨/header⟩

⟨@@=pdf⟩
⟨*package⟩
```

2.1 Conversions and export functions

\pdf_name_from_unicode_e:n
\pdf_name_from_unicode_e:V

```
\cs_generate_variant:Nn \str_convert_pdfname:n { e }

\cs_new:Npn \pdf_name_from_unicode_e:n #1
  {
    \__kernel_pdf_name_from_unicode_e:n { #1 }
  }

\cs_generate_variant:Nn \pdf_name_from_unicode_e:n {V}
```

(*End of definition for* `\pdf_name_from_unicode_e:n`. *This function is documented on page* 1.)

The convert command must use a different value the source encoding depending on the engines. Until the PR in str-convert is active we add the alias here too

```
\bool_lazy_any:nTF
  {
    \sys_if_engine_luatex_p:
    \sys_if_engine_xetex_p:
  }
  {
    \prop_gput:Nnn \g__str_alias_prop { default } {  }
  }
  {
    \prop_gput:Nnn \g__str_alias_prop { default } { utf8 }
  }
```

\pdf_string_from_unicode:nnN

```
\msg_new:nnn {pdfmanagement}{ unknown-convert}
 {
   Unknown~string~conversion~method~'#1'!
 }
\cs_new:Npn \pdf_string_from_unicode:nnN #1 #2 #3
  {
    \cs_if_exist_use:cF { __pdf_string_from_unicode_#1:nN }
      {
        \msg_error:nnn { pdf } { unknown-convert } {#1}
        \use_none:nn
      }
    { #2 } #3
  }

\cs_generate_variant:Nn \pdf_string_from_unicode:nnN {nVN}
```

(*End of definition for* `\pdf_string_from_unicode:nnN`. *This function is documented on page 2.*)

Most converter are simply wrapper around the str-convert commands and so use the same names, with the addition raw if no delimiters are added. The exception is the one for url's: it reverts most of the percent encodings and escapes the parentheses. That's why its name is URI instead of url. The current code is probably quite slow and will need a replacement.

```
__pdf_string_from_unicode_utf8/string-raw:nN
__pdf_string_from_unicode_utf8/string:nN
__pdf_string_from_unicode_utf8/URI-raw:nN
__pdf_string_from_unicode_utf8/URI:nN
__pdf_string_from_unicode_utf16/string-raw:nN
__pdf_string_from_unicode_utf16/string:nN
__pdf_string_from_unicode_utf16/hex-raw:nN
__pdf_string_from_unicode_utf16/hex:nN
```

```
%% TODO Names need a review when it is clear which converters
%% are actually needed
%% string conversions and printing
%% we assume here that the text purify step has been done. The input is
%% a list of (utf8) chars.
%% str convert, not expandable.
%  filespec (attachment view)  tests:
%  utf8:  gr\303\274\303\237e.txt
%    %doesn't work, umlaut wrong,
%  utf8 with BOM \357\273\277gr\303\274\303\237e.txt
%    %doesn't work, umlaut wrong, bom visible
%  utf16 with BE: (FEFF)
%   \376\377\000g\000r\000\374\000\337\000e\000.\000t\000x\000t %works
%                xetex converts to <feff0067007200fc00df0065002e007400780074>
%  utf16 with BE / HEX: <FEFF0067007200FC00DF0065002E007400780074> works

% bookmarks: as pdfoutline uses () currently only utf16 with BE is usable.
% check if one can use HEX too when directly writing the object
% ==========
% uri: utf16BE/string seems not to work, hex neither
%      utf8/string works but not on macos,
%      so a specific utf8/url variant is needed
% ==========
% "input" is utf8 for pdftex, empty (native) for unicode engine
% commands to output literal strings  (...)

\cs_new_protected:cpn { __pdf_string_from_unicode_utf8/string-raw:nN }  #1 #2
  {
     \str_set_convert:Nnnn #2
        { #1 }
        { default }
        {utf8/string}
  }

\cs_new_protected:cpn { __pdf_string_from_unicode_utf8/string:nN } #1 #2
  {
    \use:c { __pdf_string_from_unicode_utf8/string-raw:nN } { #1 } #2
    \str_put_left:Nn #2  {(}
    \str_put_right:Nn #2 {)}
  }
% special url command:
\cs_new_protected:cpx { __pdf_string_from_unicode_utf8/URI-raw:nN }  #1 #2
  {
     \exp_not:N \str_set_convert:Nnnn #2
```

```
      { #1 }
      { default }
      {utf8/url}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 3A} {:}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 2F} {/}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 23} {\c_hash_str}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 5B} {[}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 5D} {]}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 40} {\c_atsign_str}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 21} {!}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 24} {\c_dollar_str}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 26} {\c_ampersand_str}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 27} {'}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 2A} {*}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 2B} {+}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 2C} {,}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 3B} {;}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 3D} {=}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 3F} {?}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 30} {0}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 31} {1}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 32} {2}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 33} {3}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 34} {4}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 35} {5}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 36} {6}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 37} {7}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 38} {8}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 39} {9}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 28} {\c_backslash_str(}
    \exp_not:N \str_replace_all:Nnn #2 {\c_percent_str 29} {\c_backslash_str)}
  }

\cs_new_protected:cpn { __pdf_string_from_unicode_utf8/URI:nN } #1 #2
  {
     \use:c { __pdf_string_from_unicode_utf8/URI-raw:nN } {#1} #2
     \str_put_left:Nn #2  {(}
     \str_put_right:Nn #2 {)}
  }
% with utf16 with BE marker
\cs_new_protected:cpn { __pdf_string_from_unicode_utf16/string-raw:nN }  #1 #2
  {
    \str_set_convert:Nnnn #2
      { #1 }
      { default }
      {utf16/string}
  }

\cs_new_protected:cpn { __pdf_string_from_unicode_utf16/string:nN } #1 #2
  {
    \use:c { __pdf_string_from_unicode_utf16/string-raw:nN } {#1} #2
    \str_put_left:Nn #2  {(}
    \str_put_right:Nn #2 {)}
  }
```

```
\cs_new_protected:cpn { __pdf_string_from_unicode_utf16/hex-raw:nN } #1 #2
  {
     \str_set_convert:Nnnn #2
       { #1 }
       { default }
       {utf16/hex}
  }

\cs_new_protected:cpn { __pdf_string_from_unicode_utf16/hex:nN } #1 #2
  {
    \use:c { __pdf_string_from_unicode_utf16/hex-raw:nN } {#1} #2
    \str_put_left:Nn #2  {<}
    \str_put_right:Nn #2 {>}
  }

```

(*End of definition for* `__pdf_string_from_unicode_utf8/string-raw:nN` *and others.*)

2.2 A \pdfstringdef replacement

The following definitions are intended to replace the `\pdfstringdef` command of hyperref. Like `\pdfstringdef` they take an input which can contain various LaTeX-commands, sets up a number of sensible replacements and then outputs into a variable something that can be used in a PDF string.

2.2.1 TeX or PDF?

We need a "TeX or PDF" command, we use for now the conditional and definition of hyperref but this should perhaps be changed.

```
\newif\ifHy@pdfstring
\def\texorpdfstring{%
   \ifHy@pdfstring
     \expandafter\@secondoftwo
   \else
     \expandafter\@firstoftwo
   \fi
}
```

2.2.2 The main conversion command

2.2.3 Converting text

The title text of the bookmark and other strings must be converted to strings suitable for a PDF. With hyperref one can use `\pdfstringdef` but we also provide a native version. This follows the implementation in the generic hyperref driver, the hyperref driver should in the end use this implementation too.

The conversion has to do two things: purifying the input and then converting it into a PDF string. `\pdfstringdef` does both together but we split up the task. This allows to post process the purified text, it also allows to use more than one conversion command.

`\l__pdf_purify_tmpa_tl` Temporary variables
`\g__pdf_purify_tmpa_str`

```
\tl_new:N\l__pdf_purify_tmpa_tl
\str_new:N\g__pdf_purify_tmpa_str
```

(*End of definition for* `\l__pdf_purify_tmpa_tl` *and* `\g__pdf_purify_tmpa_str`.)

A socket which should contain the setup definitions. A different plug should at the end call the hook `pdf/purify/setup` (if user settings should be used too).

```
\socket_new:nn{pdf/purify/setup}{0}
\socket_new_plug:nnn { pdf/purify/setup } { kernel}
 {
   \__pdf_purify_equivalents:
   \hook_use:n {pdf/purify/setup}
 }
\socket_assign_plug:nn { pdf/purify/setup } { kernel}
```

The hook inside the setup socket. By default it contains the hyperref hook:

```
\hook_new:n{pdf/purify/setup}
\providecommand\pdfstringdefPreHook{}
\hook_gput_code:nnn{pdf/purify/setup}{pdf}{\pdfstringdefPreHook}
```

A socket which should contain the code to expand/purify the content. The arguments are the content and a string variable for the result.

```
\socket_new:nn {pdf/purify} {2}
\socket_new_plug:nnn {pdf/purify} {kernel}
 {
```

Note: currently `\text_expand:n` does not remove a `\protect` command and does not expand a following token even if it has a declared replacement. But in PDF strings we want to expand as much as possible, and e.g. should be able to remove/format the `\numberline` command in a bookmark.

```
   \str_set:Ne #2 {\text_purify:n { #1 } }
 }
\socket_assign_plug:nn{pdf/purify}{kernel}
```

A hook for final cleanup, currently it contains nothing. It gets as argument a string variable and so can adapt that.

```
\hook_new_with_args:nn {pdf/purify/after}{1}
```

`\pdf_purify:nN` `\__pdf_purify:nN`

`\pdf_purify:nN` runs all the purifying steps but does not convert to a PDF string. It is meant to produce input for commands doing the conversion later.

```
\cs_new_protected:Npn \__pdf_purify:nN #1 #2 %#1 content #2 str-var
 {
   \group_begin:
   \socket_use:n {pdf/purify/setup}
   \socket_use:nnn {pdf/purify} {#1} {#2}
   \hook_use:nnw {pdf/purify/after} {1} {#2}
   \str_gset_eq:NN \g__pdf_purify_tmpa_str #2
   \group_end:
   \str_set_eq:NN #2 \g__pdf_purify_tmpa_str
 }
\cs_set_eq:NN \pdf_purify:nN\__pdf_purify:nN
```

(*End of definition for* `\pdf_purify:nN` *and* `\__pdf_purify:nN`. *This function is documented on page* 1.)

2.2.4 Setup: replacement definition for pdf strings

Similar to `\pdfstringdef` we need to declare various equivalents

```
\cs_new_protected:Npn \__pdf_purify_equivalents:
 {
   \Hy@pdfstringtrue
   \text_declare_expand_equivalent:Nn\MakeUppercase{\__pdf_purify_makeuppercase:n}
   \text_declare_expand_equivalent:Nn\MakeLowercase{\__pdf_purify_makelowercase:n}
   \cs_set:Npn\dots{…} %U+2026
   \cs_set:Npn\ldots{…}
   \cs_set:Npn\P{¶}
   \cs_set:Npn\TeX{TeX}
   \cs_set:Npn\LaTeX{LaTeX}
   \cs_set:Npn\LaTeXe{LaTeX2 }
   \cs_set:Npn\SliTeX{Sli\TeX}%
   \cs_set:Npn\MF{Metafont}%
   \cs_set:Npn\MP{Metapost}%
   \cs_set_eq:NN\phantom\use_none:n
   \cs_set_eq:NN\vphantom\use_none:n
   \cs_set_eq:NN\hphantom\use_none:n
   \cs_set_eq:NN\mathversion\use_none:n
   % \ding ?
   % \Cube ?
   % \begin ?
   % \end ?
```

Font commands.

```
   \cs_set_eq:NN\emph\use_i:n
   \cs_set_eq:NN\em\prg_do_nothing:
   \cs_set_eq:NN\textnormal\use_i:n
   \cs_set_eq:NN\textrm\use_i:n
   \cs_set_eq:NN\textsf\use_i:n
   \cs_set_eq:NN\texttt\use_i:n
   \cs_set_eq:NN\textbf\use_i:n
   \cs_set_eq:NN\textmd\use_i:n
   \cs_set_eq:NN\textit\use_i:n
   \cs_set_eq:NN\textsc\use_i:n
   \cs_set_eq:NN\textsl\use_i:n
   \cs_set_eq:NN\textup\use_i:n
   \cs_set_eq:NN\normalfont\prg_do_nothing:
   \cs_set_eq:NN\rmfamily\prg_do_nothing:
   \cs_set_eq:NN\sffamily\prg_do_nothing:
   \cs_set_eq:NN\ttfamily\prg_do_nothing:
   \cs_set_eq:NN\bfseries\prg_do_nothing:
   \cs_set_eq:NN\mdseries\prg_do_nothing:
   \cs_set_eq:NN\itshape\prg_do_nothing:
   \cs_set_eq:NN\scshape\prg_do_nothing:
   \cs_set_eq:NN\slshape\prg_do_nothing:
   \cs_set_eq:NN\upshape\prg_do_nothing:
   \cs_set_eq:NN\Huge\prg_do_nothing:
   \cs_set_eq:NN\LARGE\prg_do_nothing:
   \cs_set_eq:NN\Large\prg_do_nothing:
   \cs_set_eq:NN\footnotesize\prg_do_nothing:
```

```
    \cs_set_eq:NN\huge\prg_do_nothing:
    \cs_set_eq:NN\large\prg_do_nothing:
    \cs_set_eq:NN\normalsize\prg_do_nothing:
    \cs_set_eq:NN\scriptsize\prg_do_nothing:
    \cs_set_eq:NN\small\prg_do_nothing:
    \cs_set_eq:NN\tiny\prg_do_nothing:
```

Spaces

```
    \cs_set_eq:NN\quad\space
    \cs_set_eq:NN\qquad\space
    \cs_set_eq:NN\hspace\__pdf_purify_hspace:w
```

Colors. This handles only a few commands. But pdfstringdef ignores color completely, so probably not a practical problem.

```
    \cs_set_eq:NN\textcolor\use_ii:nn
    \cs_set_eq:NN\color \@gobble@om
    \cs_set_eq:NN\label \@gobble@om
    \cs_set_eq:NN\index \@gobble@som
    \cs_set_eq:NN\glossary \@gobble@om
    \cs_set_eq:NN\ref\__pdf_purify_ref:w
    \cs_set_eq:NN\pageref\__pdf_purify_pageref:w
    \cs_set_eq:NN\nameref\__pdf_purify_nameref:w
    \cs_set_eq:NN\autoref\HyPsd@autoref
    \let\@mkboth\@gobbletwo
    % what to do with all the \textxxx commands?
    \text_declare_expand_equivalent:Nn\textparagraph{¶}
    \text_declare_expand_equivalent:Nn\textdollar{\$}
 }
```

Replacements for reference commands

```
\cs_new:Npn\__pdf_purify_nameref:n #1
  {\cs_if_exist:cTF {r@#1}{\tl_item:cn{r@#1}{3}}{0}}
\DeclareExpandableDocumentCommand\__pdf_purify_ref:w{sm}{\@kernel@ref@exp{#2}}
\DeclareExpandableDocumentCommand\__pdf_purify_pageref:w{sm}{\@kernel@pageref@exp{#2}}
\DeclareExpandableDocumentCommand\__pdf_purify_nameref:w{sm}{\__pdf_purify_nameref:n{#2}}
%
```

Replacements for uppercase and lowercase command

```
\cs_new:Npn \__pdf_purify_makeuppercase:n #1
  {\use:e
    {\exp_args:Ne
       \text_uppercase:nn{\BCPdata { casing }}{#1}
    }
  }
\cs_new:Npn \__pdf_purify_makelowercase:n #1
  {\use:e
    {\exp_args:Ne
       \text_lowercase:nn{\BCPdata { casing }}{#1}
    }
  }
```

Replacement for hspace:

```
\NewExpandableDocumentCommand\__pdf_purify_hspace:w{sm}
  {
    \dim_compare:nNnT{#2}>{0pt}{\space}
  }
```

2.3 BDC operator commands

```
\pdf_bdc:nn
\pdf_bdc:ee
\pdf_bdc_property:nn
\pdf_bdc_shipout:ee
\pdf_bdcobject:nn
\pdf_bdcobject:n
\pdf_bmc:n
\pdf_emc:
\cs_new_protected:Npn  \pdf_bdc:nn #1 #2 { \__pdf_backend_bdc:nn { #1 }{ #2 } }
\cs_generate_variant:Nn \pdf_bdc:nn {ee}

\cs_new_protected:Npn  \pdf_bdc_property:nn #1 #2
  { \__pdf_backend_bdc_contobj:nn { #1 }{ #2 } }
\cs_new_protected:Npn  \pdf_bdc_shipout:ee #1 #2
  {
    \__pdf_backend_bdc_shipout:ee { #1 }{ #2 }
    \cs_gset_eq:NN \pdf_bdc_shipout:ee \__pdf_backend_bdc_shipout:ee
  }
\cs_new_protected:Npn  \pdf_bdcobject:nn #1 #2 { \__pdf_backend_bdcobject:nn { #1 }{ #2 } }
\cs_new_protected:Npn  \pdf_bdcobject:n #1     { \__pdf_backend_bdcobject:n  { #1 } }
\cs_new_protected:Npn  \pdf_bmc:n #1     { \__pdf_backend_bmc:n  { #1 } }
\cs_new_protected:Npn  \pdf_emc:         { \__pdf_backend_emc: }
```

(*End of definition for* `\pdf_bdc:nn` *and others. These functions are documented on page* 2.)

```
⟨/package⟩
```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Q

R

S

T

U

V