Texinfo modules documentation
# Table of Contents

1 Texinfo::Commands ......................................... 1  
1.1 Texinfo::Commands NAME .................................. 1  
1.2 Texinfo::Commands SYNOPSIS ............................... 1  
1.3 Texinfo::Commands NOTES .................................. 1  
1.4 Texinfo::Commands DESCRIPTION ............................ 1  
1.5 @-COMMAND INFORMATION ................................... 1  
1.6 @-COMMAND CLASSES ...................................... 1  
1.7 Texinfo::Commands SEE ALSO ............................... 5  
1.8 Texinfo::Commands AUTHOR ................................. 5  
1.9 Texinfo::Commands COPYRIGHT AND LICENSE ............... 5  

2 Texinfo::Common .......................................... 6  
2.1 Texinfo::Common NAME .................................... 6  
2.2 Texinfo::Common SYNOPSIS ................................ 6  
2.3 Texinfo::Common NOTES .................................... 6  
2.4 Texinfo::Common DESCRIPTION .............................. 6  
2.5 MISC INFORMATION ........................................ 6  
2.6 @-COMMAND INFORMATION .................................. 6  
2.7 Texinfo::Common METHODS .................................. 7  
2.8 Texinfo::Common SEE ALSO .................................. 11  
2.9 Texinfo::Common AUTHOR ................................... 11  
2.10 Texinfo::Common COPYRIGHT AND LICENSE ................. 11  

3 Texinfo::Parser ........................................... 12  
3.1 Texinfo::Parser NAME ..................................... 12  
3.2 Texinfo::Parser SYNOPSIS .................................. 12  
3.3 Texinfo::Parser NOTES .................................... 12  
3.4 Texinfo::Parser DESCRIPTION ............................... 12  
3.5 Texinfo::Parser METHODS .................................. 13  
3.5.1 Initialization .......................................... 13  
3.5.2 Parsing Texinfo text .................................... 14  
3.5.3 Getting information on the document .................... 14  
3.6 TEXINFO TREE ........................................... 17  
3.6.1 Element keys .......................................... 17  
3.6.2 Element types .......................................... 18  
3.6.2.1 Types for command elements .......................... 18  
3.6.2.2 Types for text elements ............................... 19  
3.6.2.3 Tree container elements ............................... 20  
3.6.2.4 Types of container elements ......................... 20  
3.6.3 Information available in the info key ................. 24  
3.6.4 Information available in the extra key ................. 24  
3.6.4.1 Extra keys available for more than one @-command ... 24
3.6.4.2 Extra keys specific of certain @-commands or containers . 25
3.7 Texinfo::Parser SEE ALSO .............................................. 29
3.8 Texinfo::Parser AUTHOR .................................................. 29
3.9 Texinfo::Parser COPYRIGHT AND LICENSE ......................... 29

4 Texinfo::Structuring ...................................................... 30
  4.1 Texinfo::Structuring NAME ............................................ 30
  4.2 Texinfo::Structuring SYNOPSIS .................................... 30
  4.3 Texinfo::Structuring NOTES ........................................ 31
  4.4 Texinfo::Structuring DESCRIPTION ................................. 31
  4.5 Texinfo::Structuring METHODS ..................................... 31
  4.6 Texinfo::Structuring SEE ALSO .................................... 36
  4.7 Texinfo::Structuring AUTHOR ....................................... 36
  4.8 Texinfo::Structuring COPYRIGHT AND LICENSE .................... 37

5 Texinfo::Report .......................................................... 38
  5.1 Texinfo::Report NAME ................................................ 38
  5.2 Texinfo::Report SYNOPSIS .......................................... 38
  5.3 Texinfo::Report NOTES ............................................... 38
  5.4 Texinfo::Report DESCRIPTION ..................................... 38
  5.5 Texinfo::Report METHODS .......................................... 38
  5.6 Texinfo::Report AUTHOR ............................................ 40
  5.7 Texinfo::Report COPYRIGHT AND LICENSE ......................... 40

6 Texinfo::Translations .................................................. 41
  6.1 Texinfo::Translations NAME ........................................ 41
  6.2 Texinfo::Translations SYNOPSIS ................................... 41
  6.3 Texinfo::Translations NOTES ........................................ 41
  6.4 Texinfo::Translations DESCRIPTION ............................... 41
  6.5 Texinfo::Translations METHODS ................................... 41
  6.6 Texinfo::Translations AUTHOR ..................................... 42
  6.7 Texinfo::Translations COPYRIGHT AND LICENSE ................... 42

7 Texinfo::Transformations ................................................. 43
  7.1 Texinfo::Transformations NAME ..................................... 43
  7.2 Texinfo::Transformations NOTES .................................... 43
  7.3 Texinfo::Transformations DESCRIPTION ............................. 43
  7.4 Texinfo::Transformations METHODS ................................ 43
  7.5 Texinfo::Transformations SEE ALSO ................................ 44
  7.6 Texinfo::Transformations AUTHOR .................................. 44
  7.7 Texinfo::Transformations COPYRIGHT AND LICENSE ............... 44
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Texinfo::Convert::Texinfo</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Texinfo::Convert::Texinfo</td>
<td>46</td>
</tr>
<tr>
<td>8.1</td>
<td>Texinfo::Convert::Texinfo NAME</td>
<td>46</td>
</tr>
<tr>
<td>8.2</td>
<td>Texinfo::Convert::Texinfo SYNOPSIS</td>
<td>46</td>
</tr>
<tr>
<td>8.3</td>
<td>Texinfo::Convert::Texinfo NOTES</td>
<td>46</td>
</tr>
<tr>
<td>8.4</td>
<td>Texinfo::Convert::Texinfo DESCRIPTION</td>
<td>46</td>
</tr>
<tr>
<td>8.5</td>
<td>Texinfo::Convert::Texinfo METHODS</td>
<td>46</td>
</tr>
<tr>
<td>8.6</td>
<td>Texinfo::Convert::Texinfo AUTHOR</td>
<td>46</td>
</tr>
<tr>
<td>8.7</td>
<td>Texinfo::Convert::Texinfo COPYRIGHT AND LICENSE</td>
<td>46</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Texinfo::Convert::Utils</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Texinfo::Convert::Utils</td>
<td>47</td>
</tr>
<tr>
<td>9.1</td>
<td>Texinfo::Convert::Utils NAME</td>
<td>47</td>
</tr>
<tr>
<td>9.2</td>
<td>Texinfo::Convert::Utils SYNOPSIS</td>
<td>47</td>
</tr>
<tr>
<td>9.3</td>
<td>Texinfo::Convert::Utils NOTES</td>
<td>47</td>
</tr>
<tr>
<td>9.4</td>
<td>Texinfo::Convert::Utils DESCRIPTION</td>
<td>47</td>
</tr>
<tr>
<td>9.5</td>
<td>Texinfo::Convert::Utils METHODS</td>
<td>47</td>
</tr>
<tr>
<td>9.6</td>
<td>Texinfo::Convert::Utils SEE ALSO</td>
<td>49</td>
</tr>
<tr>
<td>9.7</td>
<td>Texinfo::Convert::Utils AUTHOR</td>
<td>49</td>
</tr>
<tr>
<td>9.8</td>
<td>Texinfo::Convert::Utils COPYRIGHT AND LICENSE</td>
<td>49</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Texinfo::Convert::Unicode</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Texinfo::Convert::Unicode</td>
<td>50</td>
</tr>
<tr>
<td>10.1</td>
<td>Texinfo::Convert::Unicode NAME</td>
<td>50</td>
</tr>
<tr>
<td>10.2</td>
<td>Texinfo::Convert::Unicode SYNOPSIS</td>
<td>50</td>
</tr>
<tr>
<td>10.3</td>
<td>Texinfo::Convert::Unicode NOTES</td>
<td>50</td>
</tr>
<tr>
<td>10.4</td>
<td>Texinfo::Convert::Unicode DESCRIPTION</td>
<td>50</td>
</tr>
<tr>
<td>10.5</td>
<td>Texinfo::Convert::Unicode METHODS</td>
<td>50</td>
</tr>
<tr>
<td>10.6</td>
<td>Texinfo::Convert::Unicode AUTHOR</td>
<td>51</td>
</tr>
<tr>
<td>10.7</td>
<td>Texinfo::Convert::Unicode COPYRIGHT AND LICENSE</td>
<td>51</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Texinfo::Convert::NodeNameNormalization</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Texinfo::Convert::NodeNameNormalization</td>
<td>53</td>
</tr>
<tr>
<td>11.1</td>
<td>Texinfo::Convert::NodeNameNormalization NAME</td>
<td>53</td>
</tr>
<tr>
<td>11.2</td>
<td>Texinfo::Convert::NodeNameNormalization SYNOPSIS</td>
<td>53</td>
</tr>
<tr>
<td>11.3</td>
<td>Texinfo::Convert::NodeNameNormalization NOTES</td>
<td>53</td>
</tr>
<tr>
<td>11.4</td>
<td>Texinfo::Convert::NodeNameNormalization DESCRIPTION</td>
<td>53</td>
</tr>
<tr>
<td>11.5</td>
<td>Texinfo::Convert::NodeNameNormalization METHODS</td>
<td>53</td>
</tr>
<tr>
<td>11.6</td>
<td>Texinfo::Convert::NodeNameNormalization AUTHOR</td>
<td>54</td>
</tr>
<tr>
<td>11.7</td>
<td>Texinfo::Convert::NodeNameNormalization COPYRIGHT AND LICENSE</td>
<td>54</td>
</tr>
</tbody>
</table>
12 Texinfo::Convert::Text .................................. 55
  12.1 Texinfo::Convert::Text NAME .................................. 55
  12.2 Texinfo::Convert::Text SYNOPSIS ................................. 55
  12.3 Texinfo::Convert::Text NOTES .................................. 55
  12.4 Texinfo::Convert::Text DESCRIPTION .............................. 55
  12.5 Texinfo::Convert::Text METHODS ................................ 55
  12.6 Texinfo::Convert::Text AUTHOR ................................. 56
  12.7 Texinfo::Convert::Text COPYRIGHT AND LICENSE .............. 57

13 Texinfo::Convert::Converter ................................. 58
  13.1 Texinfo::Convert::Converter NAME ................................ 58
  13.2 Texinfo::Convert::Converter SYNOPSIS ............................ 58
  13.3 Texinfo::Convert::Converter NOTES ................................ 58
  13.4 Texinfo::Convert::Converter DESCRIPTION .......................... 58
  13.5 Texinfo::Convert::Converter METHODS .............................. 59
    13.5.1 Initialization .............................................. 59
    13.5.2 Getting and setting customization variables ................ 60
    13.5.3 Conversion to XML ......................................... 60
    13.5.4 Helper methods ............................................ 61
  13.6 Texinfo::Convert::Converter SEE ALSO .......................... 64
  13.7 Texinfo::Convert::Converter AUTHOR ............................ 64
  13.8 Texinfo::Convert::Converter COPYRIGHT AND LICENSE ........ 64

14 Texinfo::Convert::Info ........................................ 65
  14.1 Texinfo::Convert::Info NAME .................................... 65
  14.2 Texinfo::Convert::Info SYNOPSIS ................................ 65
  14.3 Texinfo::Convert::Info NOTES ................................... 65
  14.4 Texinfo::Convert::Info DESCRIPTION .............................. 65
  14.5 Texinfo::Convert::Info METHODS ................................ 65
  14.6 Texinfo::Convert::Info AUTHOR .................................. 66
  14.7 Texinfo::Convert::Info COPYRIGHT AND LICENSE .............. 66

15 Texinfo::Convert::HTML ....................................... 67
  15.1 Texinfo::Convert::HTML NAME ................................... 67
  15.2 Texinfo::Convert::HTML SYNOPSIS ................................ 67
  15.3 Texinfo::Convert::HTML NOTES ................................... 67
  15.4 Texinfo::Convert::HTML DESCRIPTION ............................. 67
  15.5 Texinfo::Convert::HTML METHODS ................................ 67
  15.6 Texinfo::Convert::HTML AUTHOR .................................. 68
  15.7 Texinfo::Convert::HTML COPYRIGHT AND LICENSE .............. 68
1 Texinfo::Commands

1.1 Texinfo::Commands NAME
Texinfo::Commands - Classification of commands

1.2 Texinfo::Commands SYNOPSIS

use Texinfo::Commands;
if ($Texinfo::Commands::accent_commands{$a_command}) {
    print STDERR "$a_command is an accent command\n";
}

1.3 Texinfo::Commands NOTES
The Texinfo Perl module main purpose is to be used in texi2any to convert Texinfo to other formats. There is no promise of API stability.

1.4 Texinfo::Commands DESCRIPTION
Texinfo::Commands holds a few hashes with information on @-commands and hashes classifying Texinfo @-commands.

1.5 @-COMMAND INFORMATION
Hashes are defined as our variables, and are therefore available outside of the module.

%index
    Hash describing the default Texinfo indices. The format of this hash is described in [Texinfo::Parser::indices_information], page 15.

1.6 @-COMMAND CLASSES
Hashes are defined as our variables, and are therefore available outside of the module.
The key of the hashes are @-command names without the @. The following hashes are available:

%accent_commands
    Accent @-commands taking an argument, like @' or @ringaccent, including @dotless and @tieaccent.
%block_commands
    Commands delimiting a block with a closing @end. The values are:
    conditional
        @if* commands;
    def
        Definition commands like @deffn;
    float
        @float;
format_raw
    raw output format commands such as @html or @info;

item_container
    commands with @item containing any content, @itemize and @enumerate;

item_line
    commands like @table in which the @item argument is on its line;

menu
    menu @-commands, @menu, @detailmenu and @dirent;

math
    Math block commands, like @displaymath.

multitable
    @multitable;

other
    The remaining block commands.

preformatted
    Commands whose content should not be filled, like @example or @display.

quotation
    Commands like @quotation.

raw
    @-commands that have no expansion of @-commands in their bodies (@macro, @verbatim and @ignore);

region
    Commands delimiting a region of the document out of the main processing: @titlepage, @copying, @documentdescription.

%blockitem_commands
    Block commands containing @item with possible content before an @item, like @itemize, @table or @multitable.

%brace_code_commands
    Brace commands that have their argument in code style, like @code.

%brace_commands
    The commands that take braces. Value is noarg for brace commands without argument such as @AA, @TeX, or @equiv. Other values include accent, arguments, context and other values.

%close_paragraph_commands
    Commands that stop a paragraph. Root commands are not specified here, but they also close paragraphs.
%commands\_args\_number

Set to the number of arguments separated by commas that may appear in braces or on the @-command line. That means 0 or unset for most block commands, including @example which has an unlimited (variadic) number of arguments, 1 for @quotation, 2 for @float, 1 for most brace commands, 2 for @email and @abbr, 5 for @image and @ref.

Values are not necessarily set for all the commands, as commands are also classified by type of command, some type of commands implying a number of arguments, and the number of arguments may not be set if it corresponds to the default (0 for block commands, 1 for other commands that take arguments).

%contain\_basic\_inline\_commands

Commands containing simple text only, much like paragraph text, but without @ref, @footnote, @titlefont, @anchor nor @verb.

%contain\_plain\_text\_commands

Commands accepting only plain text with accent, symbol and glyph commands.

%def\_commands

Definition commands.

%default\_index\_commands

Index entry commands corresponding to default indices. For example @cindex.

%explained\_commands

@-commands whose second argument explain first argument and further @-command call without first argument, as @abbr and @acronym.

%formattable\_line\_commands

Line commands which may be formatted as text, but that require constructing some replacement text, for example @printindex, @need or @verbatiminclude. @contents and @shortcontents are not in this hash, since they are in a corresponding situation only when the tables of contents are formatted where the commands are.

%formatted\_nobrace\_commands

Commands not taking brace formatted as text or with text in the main document body, corresponding to symbol commands such as @@ or@: and commands such as @item. @-commands appearing only in headers are not in this hash, but in in %in\_heading\_spec\_commands.

%formatted\_line\_commands

Line commands which arguments may be formatted as text, such as @center, @author, @item, @node, @chapter and other. Index commands may be formatted as text too, but they may be added with @def*index, therefore they are not in that hash. Also, in general, they are not formatted as text where they appear, only when an index is printed.

%heading\_spec\_commands

@-commands used to specify custom headings, like @everyheading.
Chapter 1: Texinfo::Commands

%in_heading_spec_commands
Special \@-commands appearing in custom headings, such as \@thischapter, \@thistitle or \@1.

%in_index_commands
\@-commands only valid in index entries, such as \@sortas or \@subentry.

%inline_conditional_commands
%inline_format_commands
Inline conditional commands, like \@inlineifclear, and inline format commands like \@inlineraw and \@inlinefmt.

%letter_no_arg_commands
\@-commands with braces but no argument corresponding to letters, like \@AA{} or \@SS{} or \@o{}.

%math_commands
\@-commands which contains math, like \@math or \@displaymath.

%line_commands
Commands that do not take braces, take arguments on the command line and are not block commands either, like \@node, \@chapter, \@cindex, \@defnxs, \@end, \@footnotestyle, \@set, \@settitle, \@itemx, \@definfoenclose, \@comment and many others.
Note that \@item is in \%line_commands for its role in \@table and similar \@-commands.

%no_paragraph_commands
Commands that do not start a paragraph.

%nobrace_commands
Command that do not take braces, do not have argument on their line and are not block commands either. The value is \textit{symbol} for single character non-alphabetical \@-commands such as \@{, \@ or \@:. Other commands in that hash include \@indent, \@tab or \@thissection.
Note that \@item is in \%nobrace_commands for its role in \@multitable, \@itemize and \@enumerate.

%non_formatted_block_commands
Block commands not formatted as text, such as \@ignore or \@macro.

%preamble_commands
\@-commands that do not stop the preamble.

%preformatted_commands
%preformatted_code_commands
%preformatted_commands is for commands whose content should not be filled, like \@example or \@display. If the command is meant for code, it is also in %preformatted_code_commands, like \@example.

%ref_commands
Cross reference \@-command referencing nodes, like \@xref or \@link.
%root_commands
   Commands that are at the root of a Texinfo document, namely @node and sectioning commands, except heading commands like @heading.

%sectioning_heading_commands
   All the sectioning and heading commands.

%variadic_commands
   Commands with unlimited arguments, like @example.

1.7 Texinfo::Commands SEE ALSO
Section 3.1 [Texinfo::Parser], page 12.

1.8 Texinfo::Commands AUTHOR
Patrice Dumas, <pertusus@free.fr>

1.9 Texinfo::Commands COPYRIGHT AND LICENSE
Copyright 2010- Free Software Foundation, Inc. See the source file for all copyright years.

   This library is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.
2 Texinfo::Common

2.1 Texinfo::Common NAME
Texinfo::Common - Texinfo modules common data and miscellaneous methods

2.2 Texinfo::Common SYNOPSIS
use Texinfo::Common;

my @commands_to_collect = ('math');
my $collected_commands = Texinfo::Common::collect_commands_in_tree($document_root, @commands_to_collect);

2.3 Texinfo::Common NOTES
The Texinfo Perl module main purpose is to be used in texi2any to convert Texinfo to other formats. There is no promise of API stability.

2.4 Texinfo::Common DESCRIPTION
Texinfo::Common holds hashes with miscellaneous information and some hashes with information on Texinfo @-commands, as well as miscellaneous methods.

2.5 MISC INFORMATION
Hashes are defined as our variables, and are therefore available outside of the module.
  TODO: undocumented %null_device_file %default_parser_customization_values
%document_settable_multiple_at_commands %document_settable_unique_at_commands
%default_converter_command_line_options %default_main_program_customization_options
%default_converter_customization @variable_string_settables %document_settable_at_commands
%def_map %commandstructuring_level %level_to_structuring_command %encoding_name_conversion_map
%texinfo_output_formats
  Cannonical output formats that have associated conditionals. In practice corresponds to format_raw %block_commands plus info and plaintext.

2.6 @-COMMAND INFORMATION
Hashes are defined as our variables, and are therefore available outside of the module.
  The key of the hashes are @-command names without the @. The following hashes are available:
%all_commands
  All the @-commands.
Chapter 2: Texinfo::Common

%def_aliases
%def_no_var_arg_commands

%def_aliases associates an aliased command to the original command, for example defun is associated to deffn.

%def_no_var_arg_commands associates a definition command name with a true value if the argument on the definition command line can contain non-metasyntactic variables. For instance, it is true for deftypevr but false for defun, since @defun argument is supposed to contain metasynntactic variables only.

%nobrace_symbol_text

Values are ASCII representation of single character non-alphabetical commands without brace such as * or ::. The value may be an empty string.

%non_formatted_brace_commands

Brace commands that are not immediately replaced with text, such as anchor, caption, errmsg and others.

%small_block_associated_command

Associate small command like smallexample to the regular command example.

2.7 Texinfo::Common METHODS

Two methods are exported in the default case for Texinfo modules messages translation in the Uniforum gettext framework, __ and __p.

The Texinfo tree and Texinfo tree elements used in argument of some functions are documented in Section 3.6 [Texinfo::Parser TEXINFO TREE], page 17. When customization information is needed, an object that defines set_conf and/or get_conf is expected, for example a converter inheriting from Texinfo::Convert::Converter, see Section 13.5.2 [Texinfo::Convert::Converter Getting and setting customization variables], page 60.

$translated_string = __($msgid)
$translated_string = __p($msgctxt, $msgid)

Returns the $msgid string translated in the Texinfo messages text domain. __p can be used instead of __ to pass a $msgctxt context string to provide translators with information on the string context when the string is short or if the translation could depend on the context. __ corresponds to the gettext function and __p to the pgettext function.

It is not advised to use those functions in user-defined code. It is not practical either, as the translatable strings marked by __ or __p need to be collected and added to the Texinfo messages domain. This facility could only be used in user-defined code with translatable strings already present in the domain anyway. In fact, these functions are documented mainly because they are automatically exported.

collect_commands_in_tree($tree, $commands_list)

Returns a hash reference with keys @-commands names specified in the $commands_list array reference and values arrays of tree elements corresponding to those @-command found in $tree by traversing the tree.

collect_commands_list_in_tree($tree, $commands_list)

Return a list reference containing the tree elements corresponding to the @-commands names specified in the $commands_list found in $tree by traversing the tree. The order of the @-commands should be kept.

$encoding_name = element_associated_processing_encoding($element)

Returns the encoding name that can be used for decoding derived from the encoding that was set where $element appeared.

$result = element_is_inline($element, $check_current)

Return true if the element passed in argument is in running text context. If the optional $check_current argument is set, check the element itself, in addition to the parent context.

($encoded_file_name, $encoding) = encode_file_name($file_name, $input_encoding)

Encode the $file_name text string to a binary string $encoded_file_name based on $input_encoding. Also returns the $encoding name actually used which may have undergone some normalization. This function is mostly a wrapper around Section “Encode::encode” in Encode which avoids calling the module if not needed. Do nothing if $input_encoding is undef.

$text = enumerate_item_representation($specification, $number)

This function returns the number or letter corresponding to item number $number for an @enumerate specification $specification, appearing on an @enumerate line. For example

   enumerate_item_representation('c', 3)

is e.

$command = find_parent_root_command($object, $tree_element)

Find the parent root command (sectioning command or node) of a tree element. The $object argument is optional, its global_commands field is used to continue through @insertcopying if in a @copying.

$entry_content_element = index_content_element($element, $prefer_reference_element)

Return a Texinfo tree element corresponding to the content of the index entry associated to $element. If $prefer_reference_element is set, prefer an untranslated element. If the element is an index command like @cindex or an @ftable @item, the content element is the argument of the command. If the element is a definition line, the index entry element is based on the name and class.

$result = is_content_empty($tree, $do_not_ignore_indexEntries)

Return true if the $tree has content that could be formatted. $do_notignore_indexEntries is optional. If set, index entries are considered to be formatted.
$file = locate_include_file($customization_information, file_path)

Locate $file_path. If $file_path is an absolute path or has . or .. in the path
directories it is checked that the path exists and is a file. Otherwise, the file
name in $file_path is located in include directories also used to find texinfo files
included in Texinfo documents. $file_path should be a binary string. undef
is returned if the file was not found, otherwise the file found is returned as a
binary string.

($index_entry, $index_info) = lookup_index_entry($index_entry_info,
indices_information)

Returns an $index_entry hash based on the $index_entry_info and
indices_information. Also returns the $index_info hash with information on
the index associated to the index entry. $index_entry_info should be an array
reference with an index name as first element and the index entry number in
that index (1-based) as second element. In general, the $index_entry_info is an
extra index_entry], page 25, associated to an element.
The $index_entry hash is described in [Texinfo::Parser index
entries], page 16. The $index_info hash is described in L[Texinfo::Parser::indices_ information], page 15.

move_index_entries_after_items_in_tree($tree)

In @enumerate and @itemize from the tree, move index entries appearing just
before @item after the @item. Comment lines between index entries are moved
too.

relate_index_entries_to_table_items_in_tree($tree)

In tables, relate index entries preceding and following an entry with said item.
Reference one of them in the entry’s entry_associated_element.

$normalized_name = normalize_top_node_name($node_string)

Normalize the node name string given in argument, by normalizing Top node
case.

protect_colon_in_tree($tree)

protect_node_after_label_in_tree($tree)

Protect colon with protect_colon_in_tree and characters that are special in
node names after a label in menu entries (tab dot and comma) with protect_node_after_label_in_tree. The protection is achieved by putting protected
characters in @asis{}.

protect_comma_in_tree($tree)

Protect comma characters, replacing , with @comma{} in tree.

$contents_result = protect_first_parenthesis($contents)

Return a contents array reference with first parenthesis in the contents array
reference protected. If $contents is undef a fatal error with a backtrace will be
emitted.

$level = section_level($section)

Return numbered level of the tree sectioning element $section, as modified by
raise/lowersections.
$element = set_global_document_command($customization_information,  
$global_commands_information, $cmdname, $command_location)
Set the Texinfo customization variable corresponding to $cmdname in  
$customization_information. The $global_commands_information should  
contain information about global commands in a Texinfo document, typically  
obtained from a parser [$parser->global_commands_information()], page 15.  
$command_location specifies where in the document the value should be taken  
from, for commands that may appear more than once. The possibilities are:
last
Set to the last value for the command.
preamble
Set sequentially to the values in the Texinfo preamble.
preamble_or_first
Set to the first value of the command if the first command is not  
in the Texinfo preamble, else set as with preamble, sequentially to  
the values in the Texinfo preamble.
The $element returned is the last element that was used to set the customization  
value, or undef if no customization value was found.
Notice that the only effect of this function is to set a customization variable  
value, no @-command side effects are run, no associated customization variables  
are set.
$status = set_informative_command_value($customization_information, $element)
Set the Texinfo customization option corresponding to the tree element  
$element. The command associated to the tree element should be a command  
that sets some information, such as @documentlanguage, @contents or  
@footnotestyle for example. Return true if the command argument was  
found and the customization variable was set.
set_output_encodings($customization_information, $parser_information)
If not already set, set OUTPUT_ENCODING_NAME based on input file encoding.  
Also set OUTPUT_PERL_ENCODING accordingly which is used to output in the  
correct encoding. In general, OUTPUT_PERL_ENCODING should not be set directly  
by user-defined code such that it corresponds to OUTPUT_ENCODING_NAME.
$splitted_contents = split_custom_heading_command_contents($contents)
Split the $contents array reference at @| in at max three parts. Return  
an array reference containing the split parts. The $contents array  
reference is supposed to be $element->{'args'}->[0]->{'contents'}  
of %Texinfo::Commands::heading_spec_commands commands such as  
@everyheading.
trim_spaces_comment_from_content($contents)
Remove empty spaces after commands or braces at begin and spaces and com-  
ments at end from a content array, modifying it.
$status = valid_customization_option($name)
Return true if the $name is a known customization option.
$status = \text{valid\_tree\_transformation}(\$name)\n
Return true if the \$name is a known tree transformation name that may be passed with TREE\_TRANSFORMATIONS to modify a texinfo tree.

2.8 Texinfo::Common SEE ALSO

Section 3.1 [Texinfo::Parser], page 12, Section 13.1 [Texinfo::Convert::Converter], page 58, and Section 5.1 [Texinfo::Report], page 38.

2.9 Texinfo::Common AUTHOR

Patrice Dumas, <pertusus@free.fr>

2.10 Texinfo::Common COPYRIGHT AND LICENSE

Copyright 2010- Free Software Foundation, Inc. See the source file for all copyright years.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.
3 Texinfo::Parser

3.1 Texinfo::Parser NAME
Texinfo::Parser - Parse Texinfo code into a Perl tree

3.2 Texinfo::Parser SYNOPSIS

use Texinfo::Parser;
my $parser = Texinfo::Parser::parser();
my $tree = $parser->parse_texi_file("somefile.texi");
# a Texinfo::Report object in which the errors and warnings
# encountered while parsing are registered.
my $registrar = $parser->registered_errors();
my ($errors, $errors_count) = $registrar->errors();
foreach my $error_message (@$errors) {
    warn $error_message->{'error_line'};
}

my $indices_information = $parser->indices_information();
my $float_types_arrays = $parser->floats_information();
my $internal_references_array
    = $parser->internal_references_information();
# $labels_information is an hash reference on normalized node/float/anchor names.
my ($labels_information, $targets_list, $nodes_list) = $parser->labels_information();
# A hash reference, keys are @-command names, value is an
# array reference holding all the corresponding @-commands.
my $global_commands_information = $parser->global_commands_information();
# a hash reference on document information (encodings,
# input file name, dircategory and direntry list, for example).
my $global_information = $parser->global_information();

3.3 Texinfo::Parser NOTES
The Texinfo Perl module main purpose is to be used in texi2any to convert Texinfo to
other formats. There is no promise of API stability.

3.4 Texinfo::Parser DESCRIPTION
Texinfo::Parser will parse Texinfo text into a Perl tree. In one pass it expands user-
defined @-commands, conditionals (@ifset, @ifinfo...) and @value and constructs the
tree. Some extra information is gathered while doing the tree: for example, the @quotation
associated to an @author command, the number of columns in a multitable, or the node
associated with a section.
3.5 Texinfo::Parser METHODS

No method is exported in the default case. The module allows both an object-oriented syntax, or traditional function, with the parser as an opaque data structure given as an argument to every function.

3.5.1 Initialization

The following method is used to construct a new Texinfo::Parser object:

```perl
$parser = Texinfo::Parser::parser($options);
```

This method creates a new parser. The options may be provided as a hash reference. Most of those options correspond to Texinfo customization options described in the Texinfo manual.

- **CPP_LINE_DIRECTIVES**
  Handle cpp like synchronization lines if set. Set in the default case.

- **EXPANDED_FORMATS**
  An array reference of the output formats for which \texttt{if FORMAT} conditional blocks should be expanded. Default is empty.

- **FORMAT_MENU**
  Possible values are \texttt{nomenu}, \texttt{menu} and \texttt{sectiontoc}. Only report menu-related errors for \texttt{menu}.

- **INCLUDE_DIRECTORIES**
  An array reference of directories in which \texttt{include} files should be searched for. Default contains the working directory, ..

- **IGNORE_SPACE_AFTER_BRACED_COMMAND_NAME**
  If set, spaces after an @-command name that take braces are ignored. Default on.

- **MAX_MACRO_CALL_NESTING**
  Maximal number of nested user-defined macro calls. Default is 100000.

- **documentlanguage**
  A string corresponding to a document language set by \texttt{documentlanguage}. It overrides the document \texttt{documentlanguage} information, if present.

- **registrar**
  Section 5.1 [Texinfo::Report], page 38, object reused by the parser to register errors.

- **values**
  A hash reference. Keys are names, values are the corresponding values. Same as values set by \texttt{set}. 

3.5.2 Parsing Texinfo text

Different methods may be called to parse some Texinfo code: `parse_texi_line` for a line, `parse_texi_piece` for a fragment of Texinfo, `parse_texi_text` for a string corresponding to a full document and `parse_texi_file` for a file.

For all those functions, if the `$parser` argument is undef, a new parser object is generated to parse the line. Otherwise the parser given as an argument is used to parse into a tree.

When `parse_texi_line` is used, the resulting tree is rooted at a `root_line` type container. Otherwise, the resulting tree should be rooted at a `document_root` type container.

```perl
$tree = parse_texi_line($parser, $text, $first_line_number)
```

This function is used to parse a short fragment of Texinfo code.

- `$text` is the string containing the texinfo line.
- `$first_line_number` is the line number of the line, if undef, it will be set to 1.

```perl
$tree = parse_texi_piece($parser, $text, $first_line_number)
```

This function is used to parse Texinfo fragments.

- `$text` is the string containing the texinfo text.
- `$first_line_number` is the line number of the first text line, if undef, it will be set to 1.

```perl
$tree = parse_texi_text($parser, $text, $first_line_number)
```

This function is used to parse a text as a whole document.

- `$text` is the string containing the texinfo text.
- `$first_line_number` is the line number of the first text line, if undef, it will be set to 1.

```perl
$tree = parse_texi_file($parser, $file_name)
```

The file with name `$file_name` is considered to be a Texinfo file and is parsed into a tree. `$file_name` should be a binary string.

`undef` is returned if the file couldn’t be read.

The errors collected during the tree parsing are registered in a Section 5.1 [Texinfo::Report], page 38, object. This object is available with `registered_errors`. The errors registered in the `Texinfo::Report` object are available through the `errors` method. This method is described in [Texinfo::Report::errors], page 39.

```perl
$registrar = registered_errors($parser)
```

`$registrar` is a Section 5.1 [Texinfo::Report], page 38, object in which the errors and warnings encountered while parsing are registered. If a `registrar` is passed to the parser initialization options, it is reused, otherwise a new one is created.

3.5.3 Getting information on the document

After parsing some information about the Texinfo code that was processed is available from the parser.

Some global information is available through `global_information`:

```perl
$info = global_information($parser)
```

The `$info` returned is a hash reference. The possible keys are

- `dircategory` and `direntry`

An array of successive `@dircategory` and `@direntry` as they appear in the document.
input_encoding_name
input_perl_encoding

input_encoding_name string is the encoding name used for the Texinfo code. input_perl_encoding string is a corresponding Perl encoding name.

input_file_name
input_directory

The name of the main Texinfo input file and the associated directory. Binary strings. In texi2any, they should come from the command line (and can be decoded with the encoding in the customization variable COMMAND_LINE_ENCODING).

Some command lists are available, such that it is possible to go through the corresponding tree elements without walking the tree. They are available through global_commands_information:

$commands = global_commands_information($parser)

$commands is an hash reference. The keys are @-command names. The associated values are array references containing all the corresponding tree elements.

All the @-commands that have an associated label (so can be the target of cross references) – @node, @anchor and @float with label – have a normalized name associated, constructed as described in the HTML Xref node in the Texinfo documentation. Those normalized labels and the association with @-commands is available through labels_information:

$labels_information, $targets_list, $nodes_list = labels_information($parser)

$labels_information is a hash reference whose keys are normalized labels, and the associated value is the corresponding @-command. $targets_list is a list of labels @-command. Using $labels_information is preferred. $nodes_list is a list of all the nodes appearing in the document.

Information on @float is also available, grouped by type of floats, each type corresponding to potential @listoffloats. This information is available through the method floats_information.

$float_types = floats_information($parser)

$float_types is a hash reference whose keys are normalized float types (the first float argument, or the @listoffloats argument). The normalization is the same as for the first step of node names normalization. The value is the list of float tree elements appearing in the texinfo document.

Internal references, that is, @-commands that refer to node, anchors or floats within the document are also available:

$internal_references_array = internal_references_information($parser)

The function returns a list of cross-reference commands referring to the same document.

Information about defined indices, merged indices and index entries is also available through the indices_information method.

$indices_information = $parser->indices_information()

$indices_information is a hash reference. The keys are
in_code
   1 if the index entries should be formatted as code, 0 in the opposite case.

name
   The index name.

prefix
   An array reference of prefix associated to the index.

merged_in
   In case the index is merged to another index, this key holds the name of the index the index is merged into. It takes into account indirectly merged indices.

contained_indices
   An hash reference holding names of indices that are merged into the index, including itself. It also contains indirectly merged indices. This key is removed if the index is itself later merged to another index.

index_entries
   An array reference containing index entry structures for index entries associated with the index. The index entry could be associated to @-commands like @cindex, or @item in @vtable, or definition commands entries like @deffn. The keys of the index entry structures are

   index_name
      The index name associated to the command. Not modified if the corresponding index is merged in another index (with @synindex, for example).

   entry_element
      The element in the parsed tree associated with the @-command holding the index entry.

   entry_number
      The number of the index entry.

The following shows the references corresponding to the default indexes cp and fn, the fn index having its entries formatted as code and the indices corresponding to the following texinfo

   @defindex some
   @defcodeindex code

   $index_names = {'cp' => {'name' => 'cp', 'in_code' => 0},
                  'fn' => {'name' => 'fn', 'in_code' => 1},
                  'some' => {'in_code' => 0},
                  'code' => {'in_code' => 1}};

If name is not set, it is set to the index name.
3.6 TEXINFO TREE

A Texinfo tree element (called element because node is overloaded in the Texinfo world) is an
hash reference. There are three main categories of tree element. Tree elements associated
with an @-command have a cmdname key holding the @-command name. Tree elements
corresponding to text fragments have a text key holding the corresponding text. Finally,
the last category is other elements, which in most cases have a type key holding their
name. Text fragments and @-command elements may also have an associated type when
such information is needed.

The children of an @-command or of other container element are in the array referred
to with the args key or with the contents key. The args key is for arguments of @-
commands, either in braces or on the rest of the line after the command, depending on
the type of command. The contents key array holds the contents of the texinfo code
appearing within a block @-command, within a container, or within a @node or sectioning
@-command.

Another important key for the elements is the extra key which is associated to a hash
reference and holds all kinds of information that is gathered during the parsing and may
help with the conversion.

You can see examples of the tree structure by running makeinfo like this:

```
makeinfo -c DUMP_TREE=1 -c TEXINFO_OUTPUT_FORMAT=parse document.texi
```

For a simpler, more regular representation of the tree structure, you can do:

```
makeinfo -c TEXINFO_OUTPUT_FORMAT=debugtree document.texi
```

3.6.1 Element keys

cmdname

The command name of @-command elements.

text

The text fragment of text elements.

type

The type of element considered, in general a container. Frequent types en-
countered are paragraph for a paragraph container, brace CommandLineArg for the
container holding the brace @-commands contents, line_Arg and block line_Arg
contain the arguments appearing on the line of @-commands. Text fragments
may have a type to give an information of the kind of text fragment, for example
spaces before paragraph is associated to spaces appearing before a paragraph
beginning. Most @-commands elements do not have a type associated.

args

Arguments in braces or on @-command line. An array reference.

contents

The TexInfo appearing in the element. For block commands, other containers,
@node and sectioning commands. An array reference.

parent

The parent element.
source_info
An hash reference corresponding to information on the location of the element in the Texinfo input manual. It should mainly be available for @-command elements, and only for @-commands that are considered to be complex enough that the location in the document is needed, for example to prepare an error message.

The keys of the line number hash references are

line_nr
The line number of the @-command.

file_name
The file name where @-command appeared.

macro
The user macro name the @-command is expanded from.

info
A hash reference holding any other information that cannot be obtained otherwise from the tree. See Section 3.6.3 [Information available in the info key], page 24.

extra
A hash reference holding information that could also be obtained from the tree, but is directly associated to the element to simplify downstream code. See Section 3.6.4 [Information available in the extra key], page 24.

3.6.2 Element types
3.6.2.1 Types for command elements
Some types can be associated with @-commands (in addition to cmdname), although usually there will be no type at all. The following are the possible values of type for tree elements for @-commands.

command_as_argument
This is the type of a command given in argument of @itemize, @table, @vtable or @ftable. For example in

@itemize @bullet
@item item
@end itemize

the element corresponding with bullet has the following keys:

'cmdname' => 'bullet'
'type' => 'command_as_argument'

The parent @-command has an entry in extra for the command_as_argument element:

'cmdname' => 'itemize'
'extra' => {'command_as_argument' => $command_element_as_argument}
def_line
This type may be associated with a definition command with a x form, like @defunx, @defvrx. For the form without x, the associated def_line is the first contents element. It is described in more details below.

definfoenclose_command
This type is set for an @-command that is redefined by @definfoenclose. The beginning is in {'extra'}->{'begin'} and the end in {'extra'}->{'end'}.

index_entry_command
This is the type of index entry command like @cindex, and, more importantly user-defined index entry commands. So for example if there is:
@defindex foo
...

@fooindex index entry
the @fooindex @-command element will have the index_entry_command type.

3.6.2.2 Types for text elements
The text elements may have the following types (or may have no type at all):

after_menu_description_line
space_at_end_menu_node
Space after a node in the menu entry, when there is no description, and space appearing after the description line.

empty_line
An empty line (possibly containing whitespace characters only).

ignorable_spaces_after_command
spaces appearing after an @-command without braces that does not take takes argument on the line, but which is followed by ignorable spaces, such as @item in @itemize or @multitable, or @noindent.

spaces_after_close_brace
Spaces appearing after a closing brace, for some rare commands for which this space should be ignorable (like @caption or @sortas).

spaces_before_paragraph
Space appearing before a paragraph beginning.

raw
Text in an environment where it should be kept as is (in @verbatim, @verb, @macro body).

rawline_arg
Used for the arguments to some special line commands whose arguments aren't subject to the usual macro expansion. For example @set, @clickstyle, @unmacro, @comment. The argument is associated to the text key.

spaces_at_end
Space within an index @-command before an @-command interrupting the index command.
text_after_end
Text appearing after @bye.

text_before_beginning
Text appearing before real content, including the \input texinfo.tex.

untranslated
English text added by the parser that may need to be translated during conversion. Happens for @def* @-commands aliases that leads to prepending text such as ‘Function’.

3.6.2.3 Tree container elements
Some types of element are containers of portions of the tree, either for the whole tree, or for contents appearing before @node and sectioning commands.

before_node_section
Content before nodes and sectioning commands at the beginning of document_root.

document_root
root_line
root_line is the type of the root tree when parsing Texinfo line fragments using parse_texi_line. document_root is the document root otherwise.
document_root first content should be before_node_section, then nodes and sections @-commands elements, @bye element and postamble_after_end.

postamble_after_end
This container holds everything appearing after @bye.
preamble_before_beginning
This container holds everything appearing before the first content, including the \input texinfo.tex line and following blank lines.
preamble_before_setfilename
This container holds everything that appears before @setfilename.
preamble_before_content
This container holds everything appearing before the first formatted content, corresponding to the preamble in the Texinfo documentation.

3.6.2.4 Types of container elements
The other types of element are containers with other elements appearing in their contents. The paragraph container holds normal text from the Texinfo manual outside of any @-commands, and within @-commands with blocks of text (@footnote, @itemize @item, @quotation for example). The preformatted container holds the content appearing in @-commands like @example and the rawpreformatted container holds the content appearing in format commands such as @html. The other containers are more specific.

The types of container element are the following:

balanced_braces
Special type containing balanced braces content (braces included) in the context where they are valid, and where balanced braces need to be collected to know
when a top-level brace command is closed. In `@math`, in raw output format brace commands and within brace `@-commands` in raw output format block commands.

`before_item`  
A container for content before the first `@item` of block `@-commands` with items (`@table`, `@multitable`, `@enumerate`...).

`brace_command_arg`  
`brace_command_context`  
`line_arg`  
`block_line_arg`  
`following_arg`  
Those containers occur within the `args` array of `@-commands` taking an argument. `brace_command_arg` is used for the arguments to commands taking arguments surrounded by braces (and in some cases separated by commas). `brace_command_context` is used for `@-commands` with braces that start a new context (`@footnote`, `@caption`, `@math`).

`line_arg` is used for commands that take the texinfo code on the rest of the line as their argument, such as `@settitle`, `@node`, `@section`. `block_line_arg` is similar but is used for commands that start a new block (which is to be ended with `@end`).

`following_arg` is used for the accent `@-commands` argument that did not use braces but instead followed the `@-command`, possibly after a space, as

```
@~n  
@ringaccent A
```

For example

```
@code{in code}
```

leads to

```
{"cmdname" => "code",
 "args" => [{"type" => "brace_command_arg",
             "contents" => [{"text" => "in code"}]}]
```

As an exception, `@value` flag argument is directly in the `args` array reference, not in a `brace_command_arg` container. Note that only `@value` commands that are not expanded because there is no corresponding value set are present as elements in the tree.

`bracketed_arg`  
Bracketed argument. On definition command and on `@multitable` line.

`bracketed_linemacro_arg`  
Argument of a user defined linemacro call in bracket. It holds directly the argument text (which does not contain the braces) and does not contain other elements. It should not appear directly in the tree as the user defined linemacro call is replaced by the linemacro body.

`def_aggregate`  
Contains several elements that together are a single unit on a `@def*` line.
def_line

def_item

inter_def_item

The \texttt{def_line} type is either associated with a container within a definition command, or is the type of a definition command with a \texttt{x} form, like \texttt{@deffnx}, or \texttt{@defline}. It holds the definition line arguments. The container with type \texttt{def_item} holds the definition text content. Content appearing before a definition command with a \texttt{x} form is in an \texttt{inter_def_item} container.

macro_call

rmacro_call

linemacro_call

Container holding the arguments of a user defined macro, linemacro or rmacro. It should not appear directly in the tree as the user defined call is expanded. The name of the macro, rmacro or linemacro is the the info \texttt{command_name} value.

macro_name

macro_arg

Taken from \texttt{@macro} definition and put in the \texttt{args} key array of the macro, \texttt{macro_name} is the type of the text fragment corresponding to the macro name, \texttt{macro_arg} is the type of the text fragments corresponding to macro formal arguments.

menu_comment

The \texttt{menu_comment} container holds what is between menu entries in menus. For example, in:

\begin{verbatim}
@menu
 Menu title

 * entry::

   Between entries
 * other::
@end menu
\end{verbatim}

Both

Menu title

and

Between entries

will be in a \texttt{menu_comment}. 
menu_entry
menu_entry_leading_text
menu_entry_name
menu_entry_separator
menu_entry_node
menu_entry_description

A menu_entry holds a full menu entry, like

* node:: description.

The different elements of the menu entry are in the menu_entry contents array reference.

menu_entry_leading_text holds the star and following spaces. menu_entry_name is the menu entry name (if present), menu_entry_node corresponds to the node in the menu entry, menu_entry_separator holds the text after the node and before the description, in most cases :: . Lastly, menu_entry_description is for the description.

multitable_head
multitable_body
row

In @multitable, a multitable_head container contains all the rows with @headitem, while multitable_body contains the rows associated with @item. A row container contains the @item and @tab forming a row.

paragraph

A paragraph. The contents of a paragraph (like other container elements for Texinfo content) are elements representing the contents of the paragraph in the order they occur, such as text elements without a cmdname or type, or @-command elements for commands appearing in the paragraph.

preformatted

Texinfo code within a format that is not filled. Happens within some block commands like @example, but also in menu (in menu descriptions, menu comments...).

rawpreformatted

Texinfo code within raw output format block commands such as @tex or @html.

table_entry
table_term
table_definition
inter_item

Those containers appear in @table, @ftable and @vtable. A table_entry container contains an entire row of the table. It contains a table_term container, which holds all the @item and @itemx lines. This is followed by a table_definition container, which holds the content that is to go into the second column of the table.

If there is any content before an @itemx (normally only comments, empty lines or maybe index entries are allowed), it will be in a container with type inter_item at the same level of @item and @itemx, in a table_term.
3.6.3 Information available in the info key

arg_line
The string correspond to the line after the @-command for @-commands that have special arguments on their line, and for @macro line.

command_name
The name of the user defined macro, rmacro or linemacro called associated with the element holding the arguments of the user defined command call.

delimiter
@verb delimiter is in delimiter.

spaces_after_argument
A reference to an element containing the spaces after @-command arguments before a comma, a closing brace or at end of line, for some @-commands and bracketed content type with opening brace, and line commands and block command lines taking Texinfo as argument and comma delimited arguments. Depending on the @-command, the spaces_after_argument is associated with the @-command element, or with each argument element.

spaces_after_cmd_before_arg
For accent commands with spaces following the @-command, like:

@ringaccent A
@^ u

there is a spaces_after_cmd_before_arg key linking to an element containing the spaces appearing after the command in text.

Space between a brace @-command name and its opening brace also ends up in spaces_after_cmd_before_arg. It is not recommended to leave space between an @-command name and its opening brace.

spaces_before_argument
A reference to an element containing the spaces following the opening brace of some @-commands with braces and bracketed content type, spaces following @-commands for line commands and block command taking Texinfo as argument, and spaces following comma delimited arguments. For context brace commands, line commands and block commands, spaces_before_argument is associated with the @-command element, for other brace commands and for spaces after comma, it is associated with each argument element.

3.6.4 Information available in the extra key

3.6.4.1 Extra keys available for more than one @-command

element_node
The node element in the parsed tree containing the element. Set for @-commands elements that have an associated index entry and for @nodedescription.
element_region
The region command (@copying, @titlepage) containing the element, if it is in such an environment. Set for @-commands elements that have an associated index entry and for @anchor.

index_entry
The index entry information is associated to @-commands that have an associated index entry. The associated information should not be directly accessed, instead [Texinfo::Common::lookup_index_entry], page 9, should be called on the extra index_entry value. The $indices_information is the information on a Texinfo manual indices obtained from [Texinfo::Parser::indices_information], page 15. The index entry information hash returned by Texinfo::Common::lookup_index_entry is described in [index_entries], page 16.
Currently, the index_entry value is an array reference with an index name as first element and the index entry number in that index (1-based) as second element.

index_ignore_chars
A string containing the characters flagged as ignored in key sorting in the document by setting flags such as txiindexbackslashignore. Set, if not empty, for @-commands elements that have an associated index entry.

misc_args
An array holding strings, the arguments of @-commands taking simple textual arguments as arguments, like @everyheadingmarks, @frenchspacing, @alias, @synindex, @columnfractions.

missing_argument
Set for some @-commands with line arguments and a missing argument.

text_arg
The string correspond to the line after the @-command for @-commands that have an argument interpreted as simple text, like @setfilename, @end or @documentencoding.

### 3.6.4.2 Extra keys specific of certain @-commands or containers

@abbr
@acronym
The first argument normalized is in normalized.

@anchor

@float
@-commands that are targets for cross-references have a normalized key for the normalized label, built as specified in the Texinfo documentation in the HTML Xref node. There is also a node_content key for an array holding the corresponding content.
@author

If in a @titlepage, the titlepage is in titlepage, if in @quotation or @smallquotation, the corresponding tree element is in quotation.

The author tree element is in the authors array of the @titlepage or the @quotation or @smallquotation it is associated with.

@click

In clickstyle there is the current clickstyle command.

definition command

def, command holds the command name, without x if it is an x form of a definition command. original def cmdname is the original def command.

If it is an x form, it has not_after_command set if not appearing after the definition command without x.

def_line

For each element in a def_line, the key def role holds a string describing the meaning of the element. It is one of category, name, class, type, arg, typearg, spaces or delimiter, depending on the definition.

The def index element is a Texinfo tree element corresponding to the index entry associated to the definition line, based on the name and class. If needed this element is based on translated strings. In that case, if @documentlanguage is defined where the def_line is located, documentlanguage holds the documentlanguage value. def index ref element is similar, but not translated, and only set if there could have been a translation.

The omit_def name space key value is set and true if the Texinfo variable txidefnamenspace was set for the def_line, signaling that the space between function definition name and arguments should be omitted.

@definfoenclose defined commands

begin holds the string beginning the @definfoenclose, end holds the string ending the @definfoenclose.

@documentencoding

The argument, normalized is in input_encoding_name.

@enumerate

The enumerate_specification extra key contains the enumerate argument.

@float

@listoffloats

If @float has a first argument, and for @listoffloats argument there is a float_type key with the normalized float type.

caption and shortcaption hold the corresponding tree elements associated to a @float. The @caption or @shortcaption have the float tree element stored in float.
index entry @-command

@subentry

If an index entry @-command, such as @cindex, or a @subentry contains a @sortas command, sortas holds the @sortas command content formatted as plain text.

subentry links to the next level @subentry element.

Index entry @-command (but not @subentry) can also have seeentry and seealso keys that link to the corresponding @-commands elements.

@inlinefmt
@linenew
@inlinefmtifelse
@inlineifclear
@inlineifset

The first argument is in format. If an argument has been determined as being expanded by the Parser, the index of this argument is in expand_index. Index numbering begins at 0, but the first argument is always the format or flag name, so, if set, it should be 1 or 2 for @inlinefmtifelse, and 1 for other commands.

@item in @enumerate or @itemize

The item_number extra key holds the number of this item.

@item and @tab in @multitable

The cell_number index key holds the index of the column of the cell.

@itemize
@table
@vtable
@ftable

The command_as_argument extra key points to the @-command on as argument on the @-command line.

If the command in argument for @table, @vtable or @ftable is @kbd and the context and @kbdinputstyle is such that @kbd should be formatted as code, the command_as_argument_kbd_code extra key is set to 1.

@kbd
code is set depending on the context and @kbdinputstyle.

@macro

invalid_syntax is set if there was an error on the @macro line. info key hash arg_line holds the line after @macro.

menu_entry_node

Extra keys with information about the node entry label same as those appearing in the @node line_arg explicit directions arguments extra hash labels information.
@multitable
The key max_columns holds the maximal number of columns. If there is a @columnfractions as argument, then the columnfractions key is associated with the element for the @columnfractions command.

@node
Explicit directions labels information is in the line_arg arguments extra node direction @node arguments. They consist in a hash with the node_content key for an array holding the corresponding content, a manual_content key if there is an associated external manual name, and a normalized key for the normalized label, built as specified in the HTML Xref Texinfo documentation node.

An associated_section key holds the tree element of the sectioning command that follows the node. An node_preceding_part key holds the tree element of the @part that precedes the node, if there is no sectioning command between the @part and the node. A node_description key holds the first @nodedescription associated to the node.

A node containing a menu have a menus key which refers to an array of references to menu elements occurring in the node.

The first node containing a @printindex @-command has the isindex key set.

paragraph
The indent or noindent key value is set if the corresponding @-commands are associated with that paragraph.

@part
The next sectioning command tree element is in part_associated_section. The following node tree element is in part_following_node if there is no sectioning command between the @part and the node.

@ref
@xref
@pxref
@inforef
@infolist
The node argument brace_command_arg holds information on the label, like the one appearing in the @node line_arg explicit directions arguments extra hash labels information.

row
The row_number index key holds the index of the row in the @multitable.

sectioning command
The node preceding the command is in associated_node. The part preceding the command is in associated_part. If the level of the document was modified by @raisects or @lowersects, the differential level is in sections_level.

untranslated
documentlanguage holds the @documentlanguage value. If there is a translation context, it should be in translation_context.
3.7 Texinfo::Parser SEE ALSO

3.8 Texinfo::Parser AUTHOR
Patrice Dumas, <pertusus@free.fr>

3.9 Texinfo::Parser COPYRIGHT AND LICENSE
Copyright 2010- Free Software Foundation, Inc. See the source file for all copyright years.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.
4 Texinfo::Structuring

4.1 Texinfo::Structuring NAME
Texinfo::Structuring - information on Texinfo::Parser tree

4.2 Texinfo::Structuring SYNOPSIS

use Texinfo::Structuring qw(sectioning_structure nodes_tree number_floats
associate_internal_references split_by_node split_by_section split_pages
merge_indices sort_indices elements_directions elements_file_directions);

# $tree is a Texinfo document tree. $parser is a Texinfo::Parser object.
# $config is an object implementing the get_conf() method.
my $registrar = $parser->registered_errors();
my $sections_root = sectioning_structure ($registrar, $config, $tree);
my ($labels, $targets_list, $nodes_list) = $parser->labels_information();
my $parser_information = $parser->global_information();
my $global_commands = $parser->global_commands_information();
set_menus_node_directions($registrar, $config, $parser_information,
    $global_commands, $nodes_list, $labels);
my $top_node = nodes_tree($registrar, $config, $parser_information,
    $nodes_list, $labels);
complete_node_tree_with_menus($registrar, $config, $parser_information,
    $nodes_list, $labels);

my $refs = $parser->internal_references_information();
check_nodes_are_referenced($registrar, $config, $nodes_list, $top_node,
    $labels, $refs);
associate_internal_references($registrar, $parser, $parser_information,
    $labels, $refs);

number_floats($parser->floats_information());
my $tree_units;
if ($split_at_nodes) {
    $tree_units = split_by_node($tree);
} else {
    $tree_units = split_by_section($tree);
}
split_pages($tree_units, $split);
elements_directions($config, $labels, $tree_units);
elements_file_directions($tree_units);

my $indices_information = $parser->indices_information();
my $merged_index_entries = merge_indices($indices_information);
my $index_entries_sorted;
if ($sort_by_letter) {
    $index_entries_sorted = sort_indices($registrar, $config,
        $merged_index_entries, $indices_information,
'by_letter');

} else {
    $index_entries_sorted = sort_indices($registrar, $config,
        $merged_index_entries,
        $indices_information);
}

4.3 Texinfo::Structuring NOTES
The Texinfo Perl module main purpose is to be used in texi2any to convert Texinfo to other formats. There is no promise of API stability.

4.4 Texinfo::Structuring DESCRIPTION
Texinfo::Structuring first allows to collect information on a Texinfo tree. In most case, it also requires information from a parser object to do that job. Thanks to sectioning_structure the hierarchy of sectioning commands is determined. The directions implied by menus are determined with set_menus_node_directions. The node tree is analysed with nodes_tree. Nodes directions are completed with menu directions with complete_node_tree_with_menus. Floats get their standard numbering with number_floats and internal references are matched up with nodes, floats or anchors with associate_internal_references.

The following methods depend on the output format, so are usually called from converters.

It is also possible to associate top-level contents of the tree, which consist in nodes and sectioning commands with tree unit elements that group together a node and the next sectioning element. With split_by_node nodes are considered to be the main sectioning elements, while with split_by_section the sectioning command elements are the main elements. The first mode is typical of Info format, while the second corresponds to a traditional book. The elements may be further split in pages, which are not pages as in book pages, but more like web pages, and hold series of tree unit elements.

The elements may have directions to other elements prepared by elements_directions. elements_file_directions should also set direction related to files, provided files are associated with elements by the user.

merge_indices may be used to merge indices, which may be sorted with sort_indices.

4.5 Texinfo::Structuring METHODS
No method is exported in the default case.

Most methods takes a Section 5.1 [Texinfo::Report], page 38, $registrar as argument for error reporting. Most also require Texinfo customization variables information, which means an object implementing the get_conf method, in practice the main program configuration or a converter (Section 13.5.2 [Texinfo::Convert::Converter Getting and setting customization variables], page 60). Other common input arguments such as parser information, labels or refs are obtained from a parser, see Section 3.1 [Texinfo::Parser], page 12.
associate_internal_references($registrar, $customization_information, $parser_information, $labels, $refs)

Verify that internal references (@ref and similar without fourth of fifth argument and menu entries) have an associated node, anchor or float. Set the normalized key in the extra hash menu_entry_node hash for menu entries and in the first argument extra hash for internal references @ref and similar @-commands. Register errors in $registrar.

check_nodes_are_referenced($registrar, $customization_information, $nodes_list, $top_node, $labels, $refs)

Check that all the nodes are referenced (in menu, @*ref or node direction). Register errors in $registrar.

Should be called after complete_node_tree_with_menus in order to have the autogenerated menus available.

complete_node_tree_with_menus($registrar, $customization_information, $nodes_list, $top_node)

Complete nodes directions with menu directions. Check consistency of menus, sectionning and nodes direction structures. Register errors in $registrar.

elements_directions($customization_information, $labels, $tree_units)

Directions are set up for the tree unit elements in the array reference $tree_units given in argument. The corresponding hash is in {'structure'}->{'directions'} and keys correspond to directions while values are elements.

The following directions are set up:

This

The element itself.

Forward

Element next.

Back

Previous element.

NodeForward

Following node element in reading order. It is the next node, or the first in menu or the next of the up node.

NodeBack

Preceding node element.

NodeUp

NodeNext

NodePrev

The up, next and previous node elements.

Up

Next
The up, next and previous section elements.

For top level elements, the previous top level element. For other elements the up top level element. For example, for a chapter element it is the previous chapter, for a subsection element it is the chapter element that contains the subsection.

The next top level section element.

In the directions reference described above for `elements_directions`, sets the `PrevFile` and `NextFile` directions to the elements in previous and following files. It also sets `FirstInFile*` directions for all the elements by using the directions of the first element in file. So, for example, `FirstInFileNodeNext` is the next node of the first element in the file of each element.

The API for association of pages/elements to files is not defined yet.

```perl
@nodes_list = get_node_node_childs_from_sectioning($node)
```

`$node` is a node tree element. Find the node `$node` children based on the sectioning structure. For the node associated with `@top` sectioning command, the sections associated with parts are considered.

```perl
$entry_key = index_entry_sort_string($main_entry, $entry_tree_element, $sortas, $options)
```

Return a string suitable as a sort string, for index entries. The index entry processed is `$entry_tree_element`, and can be a `@subentry`. `$main_entry` is the main index entry tree element that can be used to gather information. `$sortas` can be given to override the sort string (typically obtained from `@sortas`). The `$options` are options used for Texinfo to text conversion for the generation of the sort string, typically obtained from `[setup_index_entry_keys_formatting]`, page 35.

```perl
$merged_entries = merge_indices($indices_information)
```

Using information returned by `[Texinfo::Parser::indices_information]`, page 15, a structure holding all the index entries by index name is returned, with all the entries of merged indices merged with those of the indice merged into.

The `$merged_entries` returned is a hash reference whose keys are the index names and values arrays of index entry structures described in details in `[Texinfo::Parser index_entries]`, page 16.

```perl
$new_block = new_block_command($content, $parent, $command_name)
```

Returns the texinfo tree corresponding to a block command named `$command_name` with contents `$content` and parent in tree `$parent`.

```perl
$new_menu = new_complete_node_menu($node, $use_sections)
```

Returns a texinfo tree menu for node `$node`, pointing to the children of the node obtained with the sectioning structure. If `$use_sections` is set, use section names for the menu entry names.
$detailmenu = new master_menu($translations, $labels, $menus)

Returns a detailmenu tree element formatted as a master node. $translations, if defined, should be a Section 6.1 [Texinfo::Translations], page 41, object and should also hold customization information. $menus is an array reference containing the regular menus of the Top node.

$entry = new node_menu_entry($node, $use_sections)

Returns the texinfo tree corresponding to a single menu entry pointing to $node. If $use_sections is set, use the section name for the menu entry name. Returns undef if the node argument is missing.

$top_node = nodes_tree($registrar, $customization_information, $parser_information, $nodes_list, $labels)

Goes through nodes and set directions. Returns the top node. Register errors in $registrar.

This functions sets, in the structure node element hash:

node_up
node_prev
node_next

Up, next and previous directions for the node.

number_floats($float_information)

Number the floats as described in the Texinfo manual. Sets the number key in the structure hash of the float tree elements.

$command_name = section_level_adjusted_command_name($element)

Return the sectioning command name corresponding to the sectioning element $element, adjusted in order to take into account raised and lowered sections, when needed.

$sections_root, $sections_list = sectioning_structure($registrar, $customization_information, $tree)

This function goes through the tree and gather information on the document structure for sectioning commands. It returns $sections_root the root of the sectioning commands tree and a reference on the sections elements list. Errors are registered in $registrar.

It sets section elements structure hash values:

section_level

The level in the sectioning tree hierarchy. 0 is for @top or @part, 1 for @chapter, @appendix... This level is corrected by @raisesections and @lowersections.

section_number

The sectioning element number.

section_childs

An array holding sectioning elements children of the element.

section_up
section_prev
section_next
The up, previous and next sectioning elements.
toplevel_next
toplevel_prev
toplevel_up
The next and previous and up sectioning elements of toplevel sectioning elements (like @top, @chapter, @appendix), not taking into account @part elements.

set_menus_node_directions($registrar, $customization_information, $parser_information, $global_commands, $nodes_list, $labels);
Goes through menu and set directions. Register errors in $registrar.
This functions sets, in the structure node element hash reference:
  menu_child
The first child in the menu of the node.
  menu_up
  menu_next
  menu_prev
Up, next and previous directions as set in menus.

$option = setup_index_entry_keys_formatting($customization_information)
Return options for conversion of Texinfo to text relevant for index keys sorting.

(&$index_entries_sorted, $index_entries_sort_strings) = sort_indices($registrar,
$customization_information, $merged_index_entries, $indices_information, $sort_by_letter)
If $sort_by_letter is set, sort by letter, otherwise sort all entries together. In both cases, a hash reference with index names as keys $index_entries_sorted is returned.
When sorting by letter, an array reference of letter hash references is associated with each index name. Each letter hash reference has two keys, a letter key with the letter, and an entries key with an array reference of sorted index entries beginning with the letter.
When simply sorting, the array of the sorted index entries is associated with the index name.

$index_entries_sort_strings is a hash reference associating the index entries with the strings that were used to sort them.
Register errors in $registrar.

$tree_units = split_by_node($tree)
Returns a reference array of tree units where a node is associated to the following sectioning commands. Sectioning commands without nodes are also with the previous node, while nodes without sectioning commands are alone in their tree units.
Tree units are regular tree elements with type unit, the associated nodes and sectioning tree elements are in the array associated with the contents key. The
associated elements have a associated_unit key set in the structure hash that
points to the associated tree unit.

Tree units have directions in the structure hash reference, namely unit_next
and unit_prev pointing to the previous and the next tree unit.

In the extra hash reference, tree units have:

unit_command
The node command associated with the element.

$tree_units = split_by_section($tree)
Similarly with split_by_node, returns an array of tree units. This time, lone
nodes are associated with the previous sections and lone sections makes up a
tree unit.

The structure and extra hash keys set are the same, except that
unit_command is the sectioning command associated with the element.

$pages = split_pages($tree_units, $split)
The tree units from the array reference argument have an extra first_in_page
value set in the structure hash reference to the first tree unit in the group,
and based on the value of $split. The possible values for $split are

chapter
The tree units are split at chapter or other toplevel sectioning tree
units.

node
Each element has its own page.

section
The tree units are split at sectioning commands below chapter.

value evaluating to false
No splitting, only one page is returned, holding all the tree units.

warn_non_empty_parts($registrar, $customization_information, $global_commands)
Register a warning in $registrar for each @part that is not empty in
$global_commands information (typically obtained by calling global_commands_information() on a parser).

4.6 Texinfo::Structuring SEE ALSO
Texinfo manual (http://www.gnu.org/s/texinfo/manual/texinfo/), Section 3.1 [Tex-
info::Parser], page 12.

4.7 Texinfo::Structuring AUTHOR
Patrice Dumas, <pertusus@free.fr>
4.8 Texinfo::Structuring COPYRIGHT AND LICENSE

Copyright 2010- Free Software Foundation, Inc. See the source file for all copyright years.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.
5 Texinfo::Report

5.1 Texinfo::Report NAME
Texinfo::Report - Error storing for Texinfo modules

5.2 Texinfo::Report SYNOPSIS
use Texinfo::Report;

my $registrar = Texinfo::Report::new();

if ($warning_happened) {
    $registrar->line_warn($converter, sprintf("\@%s is wrongly used", $current->{'cmdname'}), $current->{'source_info'});
}

my ($errors, $errors_count) = $registrar->errors();
foreach my $error_message (@$errors) {
    warn $error_message->{'error_line'};
}

5.3 Texinfo::Report NOTES
The Texinfo Perl module main purpose is to be used in texi2any to convert Texinfo to other formats. There is no promise of API stability.

5.4 Texinfo::Report DESCRIPTION
The Texinfo::Report module helps with error handling. It is used by the Texinfo modules Section 3.1 [Texinfo::Parser], page 12, and Section 13.1 [Texinfo::Convert::Converter], page 58. To use this module, either create a new Texinfo::Report object or initialize another object such as to be able to call Texinfo::Report methods. In any case, Texinfo::Report::new() is called to setup the module.

Besides the new method, errors is used for reporting errors, and the other methods to store errors (and warnings).

5.5 Texinfo::Report METHODS
No method is exported in the default case.

The new method initializes Texinfo::Report related fields. The errors collected are available through the errors method, the other methods allow registering errors and warnings.

my $registrar = Texinfo::Report::new()
$converter->Texinfo::Report::new()

If called without argument, a Texinfo::Report object is initialized and returned. This is how the module is used in the Texinfo Parsers, as a separate object.
If called on a `$converter`, the `$converter` is initialized itself such as to be able to call `Texinfo::Report` methods. It is how it is used in the Converters.

```perl
($error_warnings_list, $error_count) = errors($registrar)
```

This function returns as `$error_count` the count of errors since calling `new`. The `$error_warnings_list` is an array of hash references one for each error, warning or error line continuation. Each of these has the following keys:

- **type**
  - May be `warning`, or `error`.

- **text**
  - The text of the error.

- **error_line**
  - The text of the error formatted with the file name, line number and macro name, as needed.

- **line_nr**
  - The line number of the error or warning.

- **file_name**
  - The file name where the error or warning occurs.

- **macro**
  - The user macro name that is expanded at the location of the error or warning.

```perl
$registrar->line_warn($text, $configuration_information, $error_location_info, $continuation, $silent)
$registrar->line_error($text, $configuration_information, $error_location_info, $continuation, $silent)
```

Register a warning or an error. The `$text` is the text of the error or warning. The `$configuration_information` object gives some information that can modify the messages or their delivery. The optional `$error_location_info` holds the information on the error or warning location. The `$error_location_info` reference on hash may be obtained from Texinfo elements `source_info` keys. It may also be setup to point to a file name, using the `file_name` key and to a line number, using the `line_nr` key. The `file_name` key value should be a binary string.

The `$continuation` optional arguments, if true, conveys that the line is a continuation line of a message.

The `$silent` optional arguments, if true, suppresses the output of a message that is output immediately if debugging is set.

The `source_info` key of Texinfo tree elements is described in more details in [Texinfo::Parser source_info], page 18.

```perl
$registrar->document_warn($configuration_information, $text, $continuation)
$registrar->document_error($configuration_information, $text, $continuation)
```

Register a document-wide error or warning. `$text` is the error or warning message. The `$configuration_information` object gives some information that can
modify the messages or their delivery. The $continuation$ optional arguments, if true, conveys that the line is a continuation line of a message.

5.6 Texinfo::Report AUTHOR
Patrice Dumas, <pertusus@free.fr>

5.7 Texinfo::Report COPYRIGHT AND LICENSE
Copyright 2010- Free Software Foundation, Inc. See the source file for all copyright years.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.
6 Texinfo::Translations

6.1 Texinfo::Translations NAME
Texinfo::Translations - Translations of output documents strings for Texinfo modules

6.2 Texinfo::Translations SYNOPSIS

    @ISA = qw(Texinfo::Translations);

    my $tree_translated = $converter->gdt('See {reference} in @cite{{book}}',
        {'reference' => $tree_reference,
         'book' => {'text' => $book_name}});

6.3 Texinfo::Translations NOTES
The Texinfo Perl module main purpose is to be used in texi2any to convert Texinfo to other formats. There is no promise of API stability.

6.4 Texinfo::Translations DESCRIPTION
The Texinfo::Translations module helps with translations in output documents.

Translation of error messages uses another interface, which is the classical gettext based perl interface. It is not described as it is described in details elsewhere, some elements are in [Texinfo::Common __ and __p], page 7.

6.5 Texinfo::Translations METHODS
No method is exported.

The gdt and pgdt methods are used to translate strings to be output in converted documents, and returns, in general, a Texinfo tree.

The replace_convert_substrings method is called by gdt to substitute replaced substrings in a translated string and convert to a Texinfo tree. It may be especially useful when overriding or reimplementing gdt.

    $tree = $object->gdt($string, $replaced_substrings, $translation_context, $mode, $lang)

The $string is a string to be translated. In the default case, the function returns a Texinfo tree, as the string is interpreted as Texinfo code after translation. $replaced_substrings is an optional hash reference specifying some substitution to be done after the translation. The key of the $replaced_substrings hash reference identifies what is to be substituted, and the value is some string, texinfo tree or array content that is substituted in the resulting texinfo tree. In the string to be translated word in brace matching keys of $replaced_substrings are replaced.

The $object is typically a converter, but can be any object that implements get_conf, or undefined (undef). If not undefined, the information in the $object is used to determine the encoding, the document language and get some customization information.
The $\textit{translation\_context}$ is optional. If not \texttt{undef} this is a translation context string for $\textit{string}$. It is the first argument of \texttt{pgettext} in the C API of Gettext. $\texttt{lang}$ is optional. If set, it overrides the document language.

For example, in the following call, the string \texttt{See \{reference\} in @cite\{\{book\}\}} is translated, then parsed as a Texinfo string, with \{\textit{reference}\} substituted by $\texttt{\$tree\_reference}$ in the resulting tree, and \{\textit{book}\} replaced by the associated texinfo tree text element:

\begin{verbatim}
$\texttt{tree} = \texttt{\$converter->gdt('See \{reference\} in @cite\{\{book\}\}',
                                           \{ 'reference' => \texttt{\$tree\_reference},
                                           'book' => \{ 'text' => \texttt{\$book\_name}\}\});}
\end{verbatim}

\texttt{gdt} uses a gettext-like infrastructure to retrieve the translated strings, using the \texttt{texinfo\_document} domain.

$\texttt{mode}$ is an optional string which may modify how the function behaves. The possible values are:

\begin{itemize}
  \item \texttt{translated\_text}
  \begin{itemize}
    \item In that case the string is not considered to be Texinfo, a plain string that is returned after translation and substitution. The substitutions may only be strings in that case.
  \end{itemize}
  \item \texttt{pgdt}($\texttt{\$translation\_context}$, $\texttt{\$string}$, $\texttt{\$replaced\_substrings}$, $\texttt{\$mode}$, $\texttt{\$lang}$)
  \begin{itemize}
    \item Same to \texttt{gdt} except that the $\texttt{\$translation\_context}$ is not optional. Calls \texttt{gdt}.
    \item This function is useful to mark strings with a translation context for translation.
    \item This function is similar to \texttt{pgettext} in the Gettext C API.
  \end{itemize}
  \item \texttt{replace\_convert\_substrings}($\texttt{\$translated\_string}$, $\texttt{\$replaced\_substrings}$, $\texttt{\$mode}$)
  \begin{itemize}
    \item $\texttt{\$translated\_string}$ is a string already translated. $\texttt{\$replaced\_substrings}$ is an optional hash reference specifying some substitution to be done. $\texttt{\$mode}$ is an optional string which may modify how the function behaves, and in particular whether the translated string should be converted to a Texinfo tree. $\texttt{\$object}$ is typically a converter, but can be any object that implements \texttt{get\_conf}, or undefined (\texttt{undef}). If not undefined, the information in the $\texttt{\$object}$ is used to get some customization information.
    \item The function performs the substitutions of substrings in the translated string and converts to a Texinfo tree if needed. It is called from \texttt{gdt} after the retrieval of the translated string.
  \end{itemize}
\end{itemize}

\section{Texinfo::Translations AUTHOR}

Patrice Dumas, \texttt{pertusus@free.fr}

\section{Texinfo::Translations COPYRIGHT AND LICENSE}

Copyright 2010- Free Software Foundation, Inc. See the source file for all copyright years.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.
7 Texinfo::Transformations

7.1 Texinfo::Transformations NAME

Texinfo::Transformations - transformations of Texinfo Perl tree

7.2 Texinfo::Transformations NOTES

The Texinfo Perl module main purpose is to be used in texi2any to convert Texinfo to other formats. There is no promise of API stability.

7.3 Texinfo::Transformations DESCRIPTION

Includes miscellaneous methods set_menus_to_simple_menu and menu_to_simple_menu to change the menu texinfo tree, as well as insert_nodes_for_sectioning_commands that adds nodes for sectioning commands without nodes and complete_tree_nodes_menus and complete_tree_nodes_missing_menu that completes the node menus based on the sectioning tree.

7.4 Texinfo::Transformations METHODS

No method is exported in the default case.

```
complete_tree_nodes_menus($tree, $add_section_names_in_entries)
    Add menu entries or whole menus for nodes associated with sections, based on the sectioning tree. If the optional $add_section_names_in_entries argument is set, a menu entry name is added using the section name. This function should be called after [sectioning_structure], page 34.

complete_tree_nodes_missing_menu($tree, $use_section_names_in_entries)
    Add whole menus for nodes associated with sections and without menu, based on the sectioning tree. If the optional $add_section_names_in_entries argument is set, a menu entry name is added using the section name. This function should be called after [sectioning_structure], page 34.

($root_content, $added_sections) = fill_gaps_in_sectioning($tree)
    This function adds empty @unnumbered and similar commands in a tree to fill gaps in sectioning. This may be used, for example, when converting from a format that can handle gaps in sectioning. $tree is the tree root. An array reference is returned, containing the root contents with added sectioning commands, as well as an array reference containing the added sectioning commands.
    If the sectioning commands are lowered or raised (with @raisessections, @lowersection) the tree may be modified with @raisessections or @lowersection added to some tree elements.

($root_content, $added_nodes) = insert_nodes_for_sectioning_commands($tree, $nodes_list, $targets_list, $labels)
    Insert nodes for sectioning commands without node in $tree. Add nodes to the labels used as targets for references $labels and $targets_list and to $nodes_list.
```
An array reference is returned, containing the root contents with added nodes, as well as an array reference containing the added nodes.

```cpp
menu_to_simple_menu($menu)
set_menus_to_simple_menu($nodes_list)
```

`menu_to_simple_menu` transforms the tree of a menu tree element. `set_menus_to_simple_menu` calls `menu_to_simple_menu` for all the menus of the nodes in `$nodes_list`.

A simple menu has no `menu_comment`, `menu_entry` or `menu_entry_description` container anymore, their content are merged directly in the menu in `preformatted` container.

```cpp
protect_hashchar_at_line_beginning($registrar, $customization_information, $tree)
```

Protect hash (#) character at the beginning of line such that they would not be considered as lines to be processed by the CPP processor. The `$registrar` and `$customization_information` arguments may be undef. If defined, the `$registrar` argument should be a Section 5.1 [Texinfo::Report], page 38, object in which the errors and warnings encountered while parsing are registered. If defined, `$customization_information` should give access to customization through `get_conf`. If both `$registrar` and `$customization_information` are defined they are used for error reporting in case an hash character could not be protected because it appeared in a raw environment.

```cpp
$modified_tree = reference_to_arg_in_tree($tree)
```

Modify `$tree` by converting reference @-commands to simple text using one of the arguments. This transformation can be used, for example, to remove reference @-command from constructed node names trees, as node names cannot contain reference @-command while there could be some in the tree used in input for the node name tree.

```cpp
regenerate_master_menu($translations, $labels)
```

Regenerate the Top node master menu, replacing the first detailmenu in Top node menus or appending at the end of the Top node menu. `$translations`, if defined, should be a Section 6.1 [Texinfo::Translations], page 41, object and should also hold customization information.

### 7.5 Texinfo::Transformations SEE ALSO


### 7.6 Texinfo::Transformations AUTHOR

Patrice Dumas, <pertusus@free.fr>

### 7.7 Texinfo::Transformations COPYRIGHT AND LICENSE

Copyright 2010- Free Software Foundation, Inc. See the source file for all copyright years.
This library is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.
8 Texinfo::Convert::Texinfo

8.1 Texinfo::Convert::Texinfo NAME
Texinfo::Convert::Texinfo - Convert a Texinfo tree to Texinfo code

8.2 Texinfo::Convert::Texinfo SYNOPSIS
use Texinfo::Convert::Texinfo qw(convert_to_texinfo);

my $texinfo_text = convert_to_texinfo($tree);

8.3 Texinfo::Convert::Texinfo NOTES
The Texinfo Perl module main purpose is to be used in texi2any to convert Texinfo to
other formats. There is no promise of API stability.

8.4 Texinfo::Convert::Texinfo DESCRIPTION
Texinfo::Convert::Texinfo converts a Texinfo tree (described in Section 3.1
[Texinfo::Parser], page 12) to Texinfo code. If the Texinfo tree results from parsing some
Texinfo document, The converted Texinfo code should be exactly the same as the initial
document, except that user defined @-macros and @value are expanded, and some invalid
code is discarded.

8.5 Texinfo::Convert::Texinfo METHODS

$textinfo_text = convert_to_texinfo($tree)
Converts the Texinfo tree $tree to Texinfo code.

8.6 Texinfo::Convert::Texinfo AUTHOR
Patrice Dumas, <pertusus@free.fr>

8.7 Texinfo::Convert::Texinfo COPYRIGHT AND LICENSE
Copyright 2010- Free Software Foundation, Inc. See the source file for all copyright years.

This library is free software; you can redistribute it and/or modify it under the terms
of the GNU General Public License as published by the Free Software Foundation; either
version 3 of the License, or (at your option) any later version.
9 Texinfo::Convert::Utils

9.1 Texinfo::Convert::Utils NAME
Texinfo::Convert::Utils - miscellaneous functions usable in all converters

9.2 Texinfo::Convert::Utils SYNOPSIS
use Texinfo::Convert::Utils;

my $today_tree = Texinfo::Convert::Utils::expand_today($converter);
my $verbatiminclude_tree
  = Texinfo::Convert::Utils::expand_verbatiminclude(undef, $converter, $verbatiminclude);

9.3 Texinfo::Convert::Utils NOTES
The Texinfo Perl module main purpose is to be used in texi2any to convert Texinfo to
other formats. There is no promise of API stability.

9.4 Texinfo::Convert::Utils DESCRIPTION
miscellaneous methods that may be useful for backends converting texinfo trees. This
module contains the methods that can be used in converters which do not inherit from
Section 13.1 [Texinfo::Convert::Converter], page 58.

9.5 Texinfo::Convert::Utils METHODS
No method is exported in the default case.

Most methods takes a $converter as argument, in some cases optionally, to get
some information, see Section 13.5.2 [Texinfo::Convert::Converter Getting and setting
customization variables], page 60, and use methods for error reporting, see Section 13.1
[Texinfo::Convert::Converter], page 58, and Section 5.1 [Texinfo::Report], page 38, and for
strings translations, see Section 6.1 [Texinfo::Translations], page 41.

Even when the caller does not inherit from Section 13.1 [Texinfo::Convert::Converter],
page 58, it could implement the required interfaces and could also have a converter available
in some cases, to call the functions which require a converter.

$result = add_heading_number($converter, $heading_element, $heading_text,
$do_number)

The $converter argument may be undef. $heading_element is a heading com-
mand tree element. $heading_text is the already formatted heading text. if the
$do_number optional argument is defined and false, no number is used and the
text is returned as is. This function returns the heading with a number and
the appendix part if needed. If $converter is not defined, the resulting string
won’t be translated.
($category, $class, $type, $name, $arguments) = definition_arguments_content($element)
$element should be a @def* Texinfo tree element. The $category, $class, $type, $name are elements corresponding to the definition @-command line. Texinfo elements on the @-command line corresponding to arguments in the function definition are returned in the $arguments array reference. Arguments correspond to text following the other elements on the @-command line. If there is no argument, $arguments will be undef.

$tree = definition_category_tree($converter, $def_line)
The $converter argument may be undef. $def_line is a def_line texinfo tree container. This function returns a texinfo tree corresponding to the category of the $def_line taking the class into account, if there is one. If $converter is not defined, the resulting string won’t be translated.

($encoded_name, $encoding) = $converter->encoded_input_file_name($converter, $character_string_name, $input_file_encoding)
($encoded_name, $encoding) = $converter->encoded_output_file_name($converter, $character_string_name)
Encode $character_string_name in the same way as other file names are encoded in converters, based on customization variables, and possibly on the input file encoding. Return the encoded name and the encoding used to encode the name. The encoded_input_file_name and encoded_output_file_name functions use different customization variables to determine the encoding. The $converter argument is not optional and is used both to access to customization variables and to access to parser information.
The <$input_file_encoding> argument is optional. If set, it is used for the input file encoding. It is useful if there is more precise information on the input file encoding where the file name appeared.

$tree = expand_today($converter)
Expand today’s date, as a texinfo tree with translations. The $converter argument is not optional and is used both to retrieve customization information and to translate strings.

$tree = expand_verbatiminclude($registrar, $customization_information, $verbatiminclude)
The $registrar argument may be undef. The $customization_information argument is required and is used to retrieve customization information Section 13.5.2 [Texinfo::Convert::Converter Getting and setting customization variables], page 60. $verbatiminclude is a @verbatiminclude tree element. This function returns a @verbatim tree elements after finding the included file and reading it. If $registrar is not defined, error messages are not registered.

($@contents, $@accent_commands) = find_innermost_accent_contents($element)
$element should be an accent command Texinfo tree element. Returns an array reference containing the innermost accent @-command contents, normally a text element with one or two letter, and an array reference containing the accent commands nested in $element (including $element).
$heading_element = find_root_command_next_heading_command($element,
$expanded_format_raw, $do_not_ignore_contents, $do_not_ignore_index_entries)

Return an heading element found in the $element contents if it appears before 
contents that could be formatted. $expanded_format_raw is a hash reference 
with raw output formats (html, docbook, xml...) as keys, associated value 
should be set for expanded raw output formats. $do_not_ignore_contents is op-
tional. If set, @contents and @shortcontents are considered to be formatted. 
$do_not_ignore_index_entries is optional. If set, index entries are considered to 
be formatted.

Only heading elements corresponding to @heading, @subheading and similar @-
commands that are not associated to nodes in general are found, not sectioning 
commands.

9.6 Texinfo::Convert::Utils SEE ALSO
Section 13.1 [Texinfo::Convert::Converter], page 58, and Section 6.1 [Texinfo::Translations], 
page 41.

9.7 Texinfo::Convert::Utils AUTHOR
Patrice Dumas, <pertusus@free.fr>

9.8 Texinfo::Convert::Utils COPYRIGHT AND LICENSE
Copyright 2010- Free Software Foundation, Inc. See the source file for all copyright years.

This library is free software; you can redistribute it and/or modify it under the terms 
of the GNU General Public License as published by the Free Software Foundation; either 
version 3 of the License, or (at your option) any later version.
10 Texinfo::Convert::Unicode

10.1 Texinfo::Convert::Unicode NAME
Texinfo::Convert::Unicode - Representation as Unicode characters

10.2 Texinfo::Convert::Unicode SYNOPSIS
use Texinfo::Convert::Unicode qw(unicode_accent encoded_accents
  unicode_text);
use Texinfo::Convert::Text qw(convert_to_text);

my ($innermost_contents, $stack)
  = Texinfo::Convert::Utils::find_innermost_accent_contents($accent);

my $formatted_accents = encoded_accents ($converter,
  convert_to_text($innermost_contents), $stack, $encoding,
  \&Texinfo::Text::ascii_accent_fallback);

my $accent_text = unicode_accent('e', $accent_command);

10.3 Texinfo::Convert::Unicode NOTES
The Texinfo Perl module main purpose is to be used in texi2any to convert Texinfo to
other formats. There is no promise of API stability.

10.4 Texinfo::Convert::Unicode DESCRIPTION
Texinfo::Convert::Unicode provides methods dealing with Unicode representation and
conversion of Unicode code points, to be used in converters.

When an encoding supported in Texinfo is given as argument of a method of the module,
the accented letters or characters returned by the method should only be represented by
Unicode code points if it is known that Perl should manage to convert the Unicode code
points to encoded characters in the encoding character set. Note that the actual conversion
is done by Perl, not by the module.

10.5 Texinfo::Convert::Unicode METHODS
$result = brace_no_arg_command($command_name, $encoding)
Return the Unicode representation of a command with brace and no argument
$command_name (like @bullet{}, @aa{} or @guilsinglleft{}), or undef if
the Unicode representation cannot be converted to encoding $encoding.

$possible_conversion = check_unicode_point_conversion($arg, $output_debug)
Check that it is possible to output actual UTF-8 binary bytes corresponding to
the Unicode code point string $arg (such as 201D). Perl gives a warning and
will not output UTF-8 for Unicode non-characters such as U+10FFFF. If the
optional $output_debug argument is set, a debugging output warning is emitted
if the test of the conversion failed. Returns 1 if the conversion is possible and can be attempted, 0 otherwise.

$\text{result} = \text{encoded\_accents}(\text{converter}, \text{text}, \text{stack}, \text{encoding}, \text{format\_accent}, \text{set\_case})$

$\text{encoding}$ is the encoding the accented characters should be encoded to. If $\text{encoding}$ not set, $\text{result}$ is set to $\text{undef}$. Nested accents and their content are passed with $\text{text}$ and $\text{stack}$. $\text{text}$ is the text appearing within nested accent commands. $\text{stack}$ is an array reference holding the nested accents texinfo tree elements. In general, $\text{text}$ is the formatted contents and $\text{stack}$ the stack returned by [Texinfo::Convert::Utils::find\_innermost\_accent\_contents], page 48.

The function tries to convert as much as possible the accents to $\text{encoding}$ starting from the innermost accent.

$\text{format\_accent}$ is a function reference that is used to format the accent commands if there is no encoded character available at some point of the conversion of the $\text{stack}$. $\text{converter}$ is a converter object optionally used by $\text{format\_accent}$. It may be $\text{undef}$ if there is no need of converter object in $\text{format\_accent}$.

If $\text{set\_case}$ is positive, the result is upper-cased, while if it is negative, the result is lower-cased.

$\text{width} = \text{string\_width}(\text{string})$

Return the string width, taking into account the fact that some characters have a zero width (like composing accents) while some have a width of 2 (most chinese characters, for example).

$\text{result} = \text{unicode\_accent}(\text{text}, \text{accent\_command})$

$\text{text}$ is the text appearing within an accent command. $\text{accent\_command}$ should be a Texinfo tree element corresponding to an accent command taking an argument. The function returns the Unicode representation of the accented character.

$\text{is\_decoded} = \text{unicode\_point\_decoded\_in\_encoding}(\text{encoding}, \text{unicode\_point})$

Return true if the $\text{unicode\_point}$ will be encoded in the encoding $\text{encoding}$. The $\text{unicode\_point}$ should be specified as a four letter string describing an hexadecimal number with letters in upper case (such as 201D). Tables are used to determine if the $\text{unicode\_point}$ will be encoded, when the encoding does not cover the whole Unicode range.

If the encoding is not supported in Texinfo, the result will always be false.

$\text{result} = \text{unicode\_text}(\text{text}, \text{in\_code})$

Return $\text{text}$ with dashes and quotes corresponding, for example to --- or ', represented as Unicode code points. If $\text{in\_code}$ is set, the text is considered to be in code style.

10.6 Texinfo::Convert::Unicode AUTHOR

Patrice Dumas, <pertusus@free.fr>

10.7 Texinfo::Convert::Unicode COPYRIGHT AND LICENSE

Copyright 2010- Free Software Foundation, Inc. See the source file for all copyright years.
This library is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.
11 Texinfo::Convert::NodeNameNormalization

11.1 Texinfo::Convert::NodeNameNormalization NAME
Texinfo::Convert::NodeNameNormalization - Normalize and transliterate Texinfo trees

11.2 Texinfo::Convert::NodeNameNormalization SYNOPSIS

```perl
use Texinfo::Convert::NodeNameNormalization qw(normalize_node
  normalize_transliterate_texinfo);

my $normalized = normalize_node({'contents' => $node_contents});

my $file_name = normalize_transliterate_texinfo({'contents'
  => $section_contents});
```

11.3 Texinfo::Convert::NodeNameNormalization NOTES
The Texinfo Perl module main purpose is to be used in texi2any to convert Texinfo to other formats. There is no promise of API stability.

11.4 Texinfo::Convert::NodeNameNormalization DESCRIPTION
Texinfo::Convert::NodeNameNormalization allows to normalize node names, with `normalize_node` following the specification described in the Texinfo manual HTML Xref node. This is useful whenever one want a unique identifier for Texinfo content, which is only composed of letter, digits, - and _. In Section 3.1 [Texinfo::Parser], page 12, `normalize_node` is used for @node, @float and @anchor names normalization, but also @float types and @acronym and @abbr first argument.

It is also possible to transliterate non-ASCII letters, instead of mangling them, with `normalize_transliterate_texinfo`, losing the uniqueness feature of normalized node names.

Another method, `transliterate_protect_file_name` transliterates non-ASCII letters and protect characters that should not appear on file names.

11.5 Texinfo::Convert::NodeNameNormalization METHODS

```perl
$partially_normalized = convert_to_normalized($tree)
    The Texinfo $tree is returned as a string, with @-commands and spaces nor-
    malized as described in the Texinfo manual HTML Xref node. ASCII 7-bit
    characters other than spaces and non-ASCII characters are left as is in the
    resulting string.

$normalized = normalize_node($tree)
    The Texinfo $tree is returned as a string, normalized as described in the Texinfo
    manual HTML Xref node.
```
The result will be poor for Texinfo trees which are not @-command arguments (on an @-command line or in braces), for instance if the tree contains @node or block commands.

$transliterated = normalize_transliterate_texinfo($tree, $no_unidecode)

The Texinfo $tree is returned as a string, with non-ASCII letters transliterated as ASCII, but otherwise similar with normalize_node output. If the optional $no_unidecode argument is set, Text::Unidecode is not used for characters whose transliteration is not built-in.

$file_name = transliterate_protect_file_name($string, $no_unidecode)

The string $string is returned with non-ASCII letters transliterated as ASCII, and ASCII characters not safe in file names protected as in node normalization. If the optional $no_unidecode argument is set, Text::Unidecode is not used for characters whose transliteration is not built-in.

11.6 Texinfo::Convert::NodeNameNormalization AUTHOR
Patrice Dumas, <pertusus@free.fr>

11.7 Texinfo::Convert::NodeNameNormalization COPYRIGHT AND LICENSE
Copyright 2010- Free Software Foundation, Inc. See the source file for all copyright years.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.
12 Texinfo::Convert::Text

12.1 Texinfo::Convert::Text NAME
Texinfo::Convert::Text - Convert Texinfo tree to simple text

12.2 Texinfo::Convert::Text SYNOPSIS
use Texinfo::Convert::Text qw(convert_to_text ascii_accent text_accents);

my $result = convert_to_text($tree);
my $result_encoded = convert_to_text($tree,
    {'enabled_encoding' => 'utf-8'});
my $result_converter = convert_to_text($tree,
    {'converter' => $converter});

my $result_accent_text = ascii_accent('e', $accent_command);
my $accents_text = text_accents($accents, 'utf-8');

12.3 Texinfo::Convert::Text NOTES
The Texinfo Perl module main purpose is to be used in texi2any to convert Texinfo to other formats. There is no promise of API stability.

12.4 Texinfo::Convert::Text DESCRIPTION
Texinfo::Convert::Text is a simple backend that converts a Texinfo tree to simple text. It is used in converters, especially for file names. The converter is very simple, and, in the default case, cannot handle output strings translation or error handling.

12.5 Texinfo::Convert::Text METHODS
$result = convert_to_text($tree, $options)
Convert a Texinfo tree to simple text. $options is a hash reference of options. The converter is very simple, and has almost no internal state besides the options. It cannot handle as is output strings translation or error storing.
If the converter option is set, some additional features may be available for the conversion of some @-commands, like output strings translation or error reporting.
The following options may be set:
enabled_encoding
If set, the value is considered to be the encoding name texinfo accented letters should be converted to. This option being set corresponds to the --enable-encoding option, or the ENABLE_ENCODING customization variable for Info and Plaintext and for some conversion to text in other formats. For file names in HTML and LaTeX,
and for DocBook or Texinfo XML, this variable should in general be set unless the output encoding is US-ASCII.

\textsc{sc}

If set, the text is upper-cased.

\texttt{code}

If set the text is in code style. (mostly --, ---, '' and ``` are kept as is).

\texttt{NUMBER\_SECTIONS}

If set, sections are numbered when output.

\texttt{sort\_string}

A somehow internal option to convert to text more suitable for alphabetical sorting rather than presentation.

\texttt{converter}

If this converter object is passed to the function, some features of this object may be used during conversion. Mostly error reporting and strings translation, as the converter object is also supposed to be a Section 5.1 [Texinfo::Report], page 38, objet. See also Section 13.1 [Texinfo::Convert::Converter], page 58.

\texttt{expanded\_formats\_hash}

A reference on a hash. The keys should be format names (like \texttt{html}, \texttt{tex}), and if the corresponding value is set, the format is expanded.

$\texttt{result\_accent\_text} = \texttt{ascii\_accent($text, $accent\_command)}$

$\texttt{text}$ is the text appearing within an accent command. $\texttt{accent\_command}$ should be a Texinfo tree element corresponding to an accent command taking an argument. The function returns a transliteration of the accented character.

$\texttt{result\_accent\_text} = \texttt{ascii\_accent\_fallback($converter, $text, $accent\_command)}$

Same as \texttt{ascii\_accent} but with an additional first argument converter, which is ignored, but needed if this function is to be in argument of functions that need a fallback for accents conversion.

$\texttt{accents\_text} = \texttt{text\_accents($accents, $encoding, $set\_case)}$

$\texttt{accents}$ is an accent command that may contain other nested accent commands. The function will format the whole stack of nested accent commands and the innermost text. If $\texttt{encoding}$ is set, the formatted text is converted to this encoding as much as possible instead of being converted as simple ASCII. If $\texttt{set\_case}$ is positive, the result is meant to be upper-cased, if it is negative, the result is to be lower-cased.

12.6 Texinfo::Convert::Text AUTHOR

Patrice Dumas, <pertusus@free.fr>
12.7 Texinfo::Convert::Text COPYRIGHT AND LICENSE

Copyright 2010- Free Software Foundation, Inc. See the source file for all copyright years.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.
13 Texinfo::Convert::Converter

13.1 Texinfo::Convert::Converter NAME
Texinfo::Convert::Converter - Parent class for Texinfo tree converters

13.2 Texinfo::Convert::Converter SYNOPSIS

```perl
package Texinfo::Convert::MyConverter;

use Texinfo::Convert::Converter;
@ISA = qw(Texinfo::Convert::Converter);

sub converter_defaults ($$) {
    return %myconverter_defaults;
}
sub converter_initialize($) {
    my $self = shift;
    $self->{'document_context'} = [{
}
    sub convert($$) {
        ...
    }
    sub convert_tree($$) {
        ...
    }
    sub output($$) {
        ...
    }
}

# end of Texinfo::Convert::MyConverter

my $converter = Texinfo::Convert::MyConverter->converter(
    {'parser' => $parser});
$converter->output($texinfo_tree);
```

13.3 Texinfo::Convert::Converter NOTES
The Texinfo Perl module main purpose is to be used in texi2any to convert Texinfo to other formats. There is no promise of API stability.

13.4 Texinfo::Convert::Converter DESCRIPTION
Texinfo::Convert::Converter is a super class that can be used to simplify converters initialization. The class also provide some useful methods.
In turn, the converter should define some methods. Two are optional, `converter_defaults`, `converter_initialize` and used for initialization, to give information to Texinfo::Convert::Converter.

The `convert_tree` method is mandatory and should convert portions of Texinfo tree. The `output` method is used by converters as entry point for conversion to a file with headers and so on. Although it is is not called from other modules, it should in general be implemented by converters. `output` is called from `texi2any`. `convert` is not required, but customarily used by converters as entry point for a conversion of a whole Texinfo tree without the headers done when outputting to a file.

Existing backends may be used as examples that implement those methods. Texinfo::Convert::Texinfo together with Texinfo::Convert::PlainTexinfo, as well as Texinfo::Convert::TextContent are trivial examples. Texinfo::Convert::Text is less trivial, although still simple, while Texinfo::Convert::DocBook is a real converter that is also not too complex.

The documentation of Section 2.1 [Texinfo::Common], page 6, Section 10.1 [Texinfo::Convert::Unicode], page 50, and Section 5.1 [Texinfo::Report], page 38, describes modules or additional function that may be useful for backends, while the parsed Texinfo tree is described in Section 3.1 [Texinfo::Parser], page 12.

### 13.5 Texinfo::Convert::Converter METHODS

#### 13.5.1 Initialization

A module subclassing Texinfo::Convert::Converter is created by calling the `converter` method that should be inherited from Texinfo::Convert::Converter.

```perl
$converter = MyConverter->converter($options)
```

The `$options` hash reference holds options for the converter. In this option hash reference a Section 3.1 [parser object], page 12, may be associated with the `parser` key. The other options are Texinfo customization options and a few other options that can be passed to the converter. Most of the customization options are described in the Texinfo manual. Those customization options, when appropriate, override the document content. **TODO what about the other options (all are used in converters; 'structuring' is available in HTML)**

The `$options` hash reference holds options for the converter.

```perl
$converter->get_info()?
```

The parser should not be available directly anymore after getting the associated information. **TODO document this associated information ('parser_info', 'indices_information', 'floats'..., most available in HTML converter, either through $converter->get_info() or label_command())**

The `converter` function returns a converter object (a blessed hash reference) after checking the options and performing some initializations, especially when a parser is given among the options. The converter is also initialized as a Section 5.1 [Texinfo::Report], page 38.

To help with these initializations, the modules subclassing Texinfo::Convert::Converter can define two methods:

```perl
%defaults = $converter->converter_defaults($options)
```

The module can provide a defaults hash for converter customization options. The `$options` hash reference holds options for the converter.
This method is called at the end of the `Texinfo::Convert::Converter` converter initialization.

### 13.5.2 Getting and setting customization variables

`Texinfo::Convert::Converter` implements a simple interface to set and retrieve Texinfo customization variables. Helper functions from diverse Texinfo modules needing customization information expect an object implementing `get_conf` and/or `set_conf`. The converter itself can therefore be used in such cases.

```perl
$converter->force_conf($variable_name, $variable_value)
```

Set the Texinfo customization option `$variable_name` to `$variable_value`. This should rarely be used, but the purpose of this method is to be able to revert a customization that is always wrong for a given output format, like the splitting for example.

```perl
$converter->get_conf($variable_name)
```

Returns the value of the Texinfo customization variable `$variable_name`.

```perl
$status = $converter->set_conf($variable_name, $variable_value)
```

Set the Texinfo customization option `$variable_name` to `$variable_value` if not set as a converter option. Returns false if the customization options was not set.

### 13.5.3 Conversion to XML

Some `Texinfo::Convert::Converter` methods target conversion to XML. Most methods take a `$converter` as argument to get some information and use methods for error reporting.

```perl
$formatted_text = $converter->xml_format_text_with_numeric_entities($text)
```

Replace quotation marks and hyphens used to represent dash in Texinfo text with numeric XML entities.

```perl
$protected_text = $converter->xml_protect_text($text)
```

Protect special XML characters (&, <, >, ") of $text.

```perl
$comment = $converter->xml_comment($text)
```

Returns an XML comment for $text.

```perl
$result = xmlAccent($text, $accent_command, $in_upper_case, $use_numeric_entities)
```

$rtext is the text appearing within an accent command. `$accent_command` should be a Texinfo tree element corresponding to an accent command taking an argument. `$in_upper_case` is optional, and, if set, the text is put in upper case. The function returns the accented letter as XML named entity if possible, falling back to numeric entities if there is no named entity and to an ASCII transliteration as last resort. `$use_numeric_entities` is optional. If set, numerical entities are used instead of named entities if possible.

```perl
$result = $converter->xml_accents($accent_command, $in_upper_case)
```

$`accent_command` is an accent command, which may have other accent commands nested. If `$in_upper_case` is set, the result should be upper cased. The function returns the accents formatted as XML.
Chapter 13: Texinfo::Convert::Converter

$result = xml_numeric_entity_accent($accent_command_name, $text)

$accent_command_name is the name of an accent command. $text is the text appearing within the accent command. Returns the accented letter as XML numeric entity, or undef if there is no such entity.

13.5.4 Helper methods

The module provides methods that may be useful for converter. Most methods take a $converter as argument to get some information and use methods for error reporting, see Section 5.1 [Texinfo::Report], page 38. Also to translate strings, see Section 6.1 [Texinfo::Translations], page 41. For useful methods that need a converter optionally and can be used in converters that do not inherit from Texinfo::Convert::Converter, see Section 9.1 [Texinfo::Convert::Utils], page 47.

$result = $converter->convert_accents($accent_command, \&format_accents,
$output_encoded_characters, $in_upper_case)

$accent_command is an accent command, which may have other accent commands nested. The function returns the accents formatted either as encoded letters if $output_encoded_characters is set, or formatted using \&format_accents. If $in_upper_case is set, the result should be uppercased.

$result = $converter->convert_document_sections($root, $file_handler)

This method splits the $root Texinfo tree at sections and calls convert_tree on the elements. If the optional $file_handler is given in argument, the result are output in $file_handler, otherwise the resulting string is returned.

$succeeded = $converter->create_destination_directory($destination_directory_path,
$destination_directory_name)

Create destination directory $destination_directory_path. $destination_directory_path should be a binary string, while $destination_directory_name should be a character string, that can be used in error messages. $succeeded is true if the creation was successful or unneeded, false otherwise.

($output_file, $destination_directory, $output_filename, $document_name, $input_basefile) = $converter->determine_files_and_directory($output_format)

Determine output file and directory, as well as names related to files. The result depends on the presence of @setfilename, on the Texinfo input file name, and on customization options such as OUTPUT, SUBDIR or SPLIT, as described in the Texinfo manual. $output_format is optional. If it is not set the current output format, if defined, is used instead. If not an empty string, _$output_format is prepended to the default directory name.

$output_file is mainly relevant when not split and should be used as the output file name. In general, if not split and $output_file is an empty string, it means
that text should be returned by the converter instead of being written to an output file. This is used in the test suite. $destination_directory is either the directory $output_file is in, or if split, the directory where the files should be created. $output_filename is, in general, the file name portion of $output_file (without directory) but can also be set based on $setfilename, in particular when $output_file is an empty string. $document_name is $output_filename without extension. $input_basefile is based on the input texinfo file name, with the file name portion only (without directory).

The strings returned are text strings.

($encoded_name, $encoding) = $converter->encoded_input_file_name($character_string_name, $input_file_encoding)
($encoded_name, $encoding) = $converter->encoded_output_file_name($character_string_name)

Encode $character_string_name in the same way as other file names are encoded in the converter, based on customization variables, and possibly on the input file encoding. Return the encoded name and the encoding used to encode the name. The encoded_input_file_name and encoded_output_file_name functions use different customization variables to determine the encoding.

The <$input_file_encoding> argument is optional. If set, it is used for the input file encoding. It is useful if there is more precise information on the input file encoding where the file name appeared.

Note that encoded_output_file_name is a wrapper around the function with the same name in [Texinfo::Convert::Utils::encoded_output_file_name], page 48, and encoded_input_file_name is a wrapper around the function with the same name in [Texinfo::Convert::Utils::encoded_input_file_name], page 48.

($caption, $prepended) = $converter->float_name_caption($float)

$float is a texinfo tree @float element. This function returns the caption element that should be used for the float formatting and the $prepended texinfo tree combining the type and label of the float.

$tree = $converter->float_type_number($float)

$float is a texinfo tree @float element. This function returns the type and number of the float as a texinfo tree with translations.

$end_line = $converter->format_comment_or_return_end_line($element)

Format comment at end of line or return the end of line associated with the element. In many cases, converters ignore comments and output is better formatted with new lines added independently of the presence of newline or comment in the initial Texinfo line, so most converters are better off not using this method.

$filename = sub $converter->node_information_filename($normalized, $node_contents)

Returns the normalized file name corresponding to the $normalized node name and to the $node_contents node name contents.
($normalized_name, $filename) =
$converter->normalized_sectioning_command_filename($element)
  Returns a normalized name $normalized_name corresponding to a section-
ing command tree element $element, expanding the command argument using transli-
teration and characters protection. Also returns $filename the corre-
sponding filename based on $normalized_name taking into account additional
constraint on file names and adding a file extension.

$converter->present_bug_message($message, $element)
  Show a bug message using $message text. Use information on $element tree
element if given in argument.

$converter->set_global_document_commands($commands_location, $selected.commands)
  Set the Texinfo customization options for @-commands. $selected.commands is
  an optional array reference containing the @-commands set, if not given all the
global informative @-commands are set. $commands_location specifies where
in the document the value should be taken from. The possibilities are:

  before
  Set to the values before document conversion, from defaults and
  command-line.

  last
  Set to the last value for the command.

  preamble
  Set sequentially to the values in the Texinfo preamble.

  preamble_or_first
  Set to the first value of the command if the first command is not
  in the Texinfo preamble, else set as with preamble, sequentially to
  the values in the Texinfo preamble.

  Notice that the only effect of this function is to set a customization variable
  value, no @-command side effects are run, no associated customization variables
  are set.

  For more information on the function used to set the value for each of the
  command, see [Texinfo::Common set_global_document_command], page 10.

$table_item_tree = $converter->table_item_content_tree($element, $contents)
  $element should be an @item or @itemx tree element. $contents should be
  corresponding texinfo tree contents. Returns a tree in which the @-command
  in argument of @*table of the $element has been applied to $contents.

$result = $converter->top_node_filename($document_name)
  Returns a file name for the Top node file using either TOP_FILE customization
  value, or EXTENSION customization value and $document_name.

  Finally, there is:

$result = $converter->output_internal_links()
  At this level, the method just returns undef. It is used in the HTML output,
  following the --internal-links option of texi2any specification.
13.6 Texinfo::Convert::Converter SEE ALSO
Section 2.1 [Texinfo::Common], page 6, Section 10.1 [Texinfo::Convert::Unicode], page 50, Section 5.1 [Texinfo::Report], page 38, Section 6.1 [Texinfo::Translations], page 41, Section 9.1 [Texinfo::Convert::Utils], page 47, and Section 3.1 [Texinfo::Parser], page 12.

13.7 Texinfo::Convert::Converter AUTHOR
Patrice Dumas, <pertusus@free.fr>

13.8 Texinfo::Convert::Converter COPYRIGHT AND LICENSE
Copyright 2011- Free Software Foundation, Inc. See the source file for all copyright years.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.
14 Texinfo::Convert::Info

14.1 Texinfo::Convert::Info NAME
Texinfo::Convert::Info - Convert Texinfo tree to Info

14.2 Texinfo::Convert::Info SYNOPSIS

my $converter
    = Texinfo::Convert::Info->converter({'parser' => $parser});

$converter->output($tree);
$converter->convert($tree);
$converter->convert_tree($tree);

14.3 Texinfo::Convert::Info NOTES

The Texinfo Perl module main purpose is to be used in texi2any to convert Texinfo to other formats. There is no promise of API stability.

14.4 Texinfo::Convert::Info DESCRIPTION
Texinfo::Convert::Info converts a Texinfo tree to Info.

14.5 Texinfo::Convert::Info METHODS

$converter = Texinfo::Convert::Info->converter($options)
     Initialize converter from Texinfo to Info.
     The $options hash reference holds options for the converter. In this option hash reference a Section 3.1 [parser object], page 12, may be associated with the parser key. The other options are Texinfo customization options and a few other options that can be passed to the converter. Most of the customization options are described in the Texinfo manual. Those customization options, when appropriate, override the document content. The parser should not be available directly anymore after getting the associated information.
     See Section 13.1 [Texinfo::Convert::Converter], page 58, for more information.

$converter->output($tree)
     Convert a Texinfo tree $tree and output the result in files as described in the Texinfo manual.

$result = $converter->convert($tree)
     Convert a Texinfo tree $tree and return the resulting output.

$result = $converter->convert_tree($tree)
     Convert a Texinfo tree portion $tree and return the resulting output. This function does not try to output a full document but only portions. For a full document use convert.
14.6 Texinfo::Convert::Info AUTHOR
Patrice Dumas, <pertusus@free.fr>

14.7 Texinfo::Convert::Info COPYRIGHT AND LICENSE
Copyright 2010- Free Software Foundation, Inc. See the source file for all copyright years.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.
15 Texinfo::Convert::HTML

15.1 Texinfo::Convert::HTML NAME
Texinfo::Convert::HTML - Convert Texinfo tree to HTML

15.2 Texinfo::Convert::HTML SYNOPSIS

my $converter
   = Texinfo::Convert::HTML->converter({'parser' => $parser});

$converter->output($tree);
$converter->convert($tree);
$converter->convert_tree($tree);
$converter->output_internal_links(); # HTML only

15.3 Texinfo::Convert::HTML NOTES
The Texinfo Perl module main purpose is to be used in texi2any to convert Texinfo to other formats. There is no promise of API stability.

15.4 Texinfo::Convert::HTML DESCRIPTION
Texinfo::Convert::HTML converts a Texinfo tree to HTML.

15.5 Texinfo::Convert::HTML METHODS
$converter = Texinfo::Convert::HTML->converter($options)
   Initialize converter from Texinfo to HTML.
   The $options hash reference holds options for the converter. In this option hash reference a Section 3.1 [parser object], page 12, may be associated with the parser key. The other options are Texinfo customization options and a few other options that can be passed to the converter. Most of the customization options are described in the Texinfo manual. Those customization options, when appropriate, override the document content. The parser should not be available directly anymore after getting the associated information.
   See Section 13.1 [Texinfo::Convert::Converter], page 58, for more information.

$converter->output($tree)
   Convert a Texinfo tree $tree and output the result in files as described in the Texinfo manual.

$result = $converter->convert($tree)
   Convert a Texinfo tree $tree and return the resulting output.

$result = $converter->convert_tree($tree)
   Convert a Texinfo tree portion $tree and return the resulting output. This function does not try to output a full document but only portions. For a full document use convert.
$result = $converter->output_internal_links()

Returns text representing the links in the document. The format should follow the
--internal-links option of the texi2any specification. This is only
supported in (and relevant for) HTML.

15.6 Texinfo::Convert::HTML AUTHOR

Patrice Dumas, <pertusus@free.fr>

15.7 Texinfo::Convert::HTML COPYRIGHT AND LICENSE

Copyright 2010- Free Software Foundation, Inc. See the source file for all copyright years.

This library is free software; you can redistribute it and/or modify it under the terms
of the GNU General Public License as published by the Free Software Foundation; either
version 3 of the License, or (at your option) any later version.
16 Texinfo::Convert::DocBook

16.1 Texinfo::Convert::DocBook NAME
Texinfo::Convert::DocBook - Convert Texinfo tree to DocBook

16.2 Texinfo::Convert::DocBook SYNOPSIS

my $converter = Texinfo::Convert::DocBook->converter({'parser' => $parser});

$converter->output($tree);
$converter->convert($tree);
$converter->convert_tree($tree);

16.3 Texinfo::Convert::DocBook NOTES
The Texinfo Perl module main purpose is to be used in texi2any to convert Texinfo to other formats. There is no promise of API stability.

16.4 Texinfo::Convert::DocBook DESCRIPTION
Texinfo::Convert::DocBook converts a Texinfo tree to DocBook.

16.5 Texinfo::Convert::DocBook METHODS

$converter = Texinfo::Convert::DocBook->converter($options)
Initialize converter from Texinfo to DocBook.

The $options hash reference holds options for the converter. In this option hash reference a Section 3.1 [parser object], page 12, may be associated with the parser key. The other options are Texinfo customization options and a few other options that can be passed to the converter. Most of the customization options are described in the Texinfo manual. Those customization options, when appropriate, override the document content. The parser should not be available directly anymore after getting the associated information.

See Section 13.1 [Texinfo::Convert::Converter], page 58, for more information.

$converter->output($tree)
Convert a Texinfo tree $tree and output the result in files as described in the Texinfo manual.

$result = $converter->convert($tree)
Convert a Texinfo tree $tree and return the resulting output.

$result = $converter->convert_tree($tree)
Convert a Texinfo tree portion $tree and return the resulting output. This function does not try to output a full document but only portions. For a full document use convert.
16.6 Texinfo::Convert::DocBook AUTHOR
Patrice Dumas, <pertusus@free.fr>

16.7 Texinfo::Convert::DocBook COPYRIGHT AND LICENSE
Copyright 2010- Free Software Foundation, Inc. See the source file for all copyright years.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.
17 Texinfo::Convert::TexinfoMarkup

17.1 Texinfo::Convert::TexinfoMarkup NAME
Texinfo::Convert::TexinfoMarkup - Convert Texinfo tree to element and attribute markup

17.2 Texinfo::Convert::TexinfoMarkup SYNOPSIS
package Texinfo::Convert::TexinfoMyMarkup;

use Texinfo::Convert::TexinfoMarkup;

@ISA = qw(Texinfo::Convert::TexinfoMarkup);

sub converter_defaults ($$) {
    return %myconverter_defaults;
}

sub txi_markup_protect_text($$) {
    my $self = shift;
    ...
}

17.3 Texinfo::Convert::TexinfoMarkup NOTES
The Texinfo Perl module main purpose is to be used in texi2any to convert Texinfo to other formats. There is no promise of API stability.

17.4 Texinfo::Convert::TexinfoMarkup DESCRIPTION
Texinfo::Convert::TexinfoMarkup converts a Texinfo tree to the Texinfo Markup Language which is based on nested elements with attributes, similar to XML. All the information present in the Texinfo tree, after expansion of @macro, @value and inclusion of include files is kept. Texinfo::Convert::TexinfoMarkup is an abstract class, to be used as a super class for modules implementing specific markup formatting functions called by Texinfo::Convert::TexinfoMarkup.

The Texinfo Markup Language elements and attributes are not documented, but the Texinfo XML output by the Texinfo::Convert::TexinfoXML subclass (Section 18.1 [Texinfo::Convert::TexinfoXML], page 74) is a straightforward formatting as XML, and is described by the texinfo DTD. Therefore the texinfo DTD can be used as a description of the structure of both Texinfo XML and of the more abstract Texinfo Markup Language.

17.5 Texinfo::Convert::TexinfoMarkup METHODS
17.5.1 Markup formatting methods defined by subclasses

The following methods should be implemented by the modules inheriting from Texinfo::Convert::TexinfoMarkup:

\$result = $converter-&gt;txi_markup_atom($atom)

Format the \$atom symbol string in a simpler way than with an element. For example in XML the formatting of the symbol is achieved with an entity.

\$result = $converter-&gt;txi_markup_comment($comment_string)

Format $comment_string as a comment.

\$result = $converter-&gt;txi_markup_convert_text($element)

Called to format the Texinfo tree $element text, which is a reference on a hash. The $element text is in the text key. The type key value may also be set to distinguish the type of text (Section 3.6.2.2 [Texinfo::Parser Types for text elements], page 19). Texinfo tree elements are described in details in Section 3.6 [Texinfo::Parser TEXINFO TREE], page 17.

\$result = $converter-&gt;txi_markup_element($format_element, $attributes)
\$result = $converter-&gt;txi_markup_open_element($format_element, $attributes)
\$result = $converter-&gt;txi_markup_close_element($format_element, $attributes)

txi_markup_element is called for the formatting of empty elements. Otherwise, txi_markup_open_element is called when an element is opened, and txi_markup_close_element is called when an element is closed. $format_element is the element name, $attributes is a reference on an array containing references on arrays of pairs, one pair for each attribute, with the attribute name as the first item of the pair and the attribute text as the second item of the pair.

\$result = $converter-&gt;txi_markup_header()

Called to format a header at the beginning of output files.

\$result = $converter-&gt;txi_markup_protect_text($string)

Protect special character in text for text fragments out of text texinfo tree elements. For example, for spaces at end of line that are ignorable in most output formats, for @set or @macro arguments.

17.5.2 Formatting state information

A method is available for subclasses to gather information on the formatting state:

$ converter-&gt;in_monospace()

Return 1 if in a context where spacing should be kept and \-\-\- or \ ' left as is, for example in @code, @example.

17.6 Texinfo::Convert::TexinfoMarkup AUTHOR

Patrice Dumas, &lt;pertusus@free.fr&gt;

17.7 Texinfo::Convert::TexinfoMarkup SEE ALSO

Section 13.1 [Texinfo::Convert::Converter], page 58. Section 18.1 [Texinfo::Convert::TexinfoXML], page 74. The Texinfo::Convert::TexinfoSXML is another subclass, which outputs SXML. It is not much documented.
17.8 Texinfo::Convert::TexinfoMarkup COPYRIGHT AND LICENSE

Copyright 2010- Free Software Foundation, Inc. See the source file for all copyright years.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.
18 Texinfo::Convert::TexinfoXML

18.1 Texinfo::Convert::TexinfoXML NAME
Texinfo::Convert::TexinfoXML - Convert Texinfo tree to TexinfoXML

18.2 Texinfo::Convert::TexinfoXML SYNOPSIS

my $converter
  = Texinfo::Convert::TexinfoXML->converter({'parser' => $parser});

$converter->output($tree);
$converter->convert($tree);
$converter->convert_tree($tree);

18.3 Texinfo::Convert::TexinfoXML NOTES
The Texinfo Perl module main purpose is to be used in texi2any to convert Texinfo to other formats. There is no promise of API stability.

18.4 Texinfo::Convert::TexinfoXML DESCRIPTION
Texinfo::Convert::TexinfoXML converts a Texinfo tree to TexinfoXML.

18.5 Texinfo::Convert::TexinfoXML METHODS

$converter = Texinfo::Convert::TexinfoXML->converter($options)
Initialize converter from Texinfo to TexinfoXML.

The $options hash reference holds options for the converter. In this option hash reference a Section 3.1 [parser object], page 12, may be associated with the parser key. The other options are Texinfo customization options and a few other options that can be passed to the converter. Most of the customization options are described in the Texinfo manual. Those customization options, when appropriate, override the document content. The parser should not be available directly anymore after getting the associated information.
See Section 13.1 [Texinfo::Convert::Converter], page 58, for more information.

$converter->output($tree)
Convert a Texinfo tree $tree and output the result in files as described in the Texinfo manual.

$kresult = $converter->convert($tree)
Convert a Texinfo tree $tree and return the resulting output.

$kresult = $converter->convert_tree($tree)
Convert a Texinfo tree portion $tree and return the resulting output. This function does not try to output a full document but only portions. For a full document use convert.
18.6 Texinfo::Convert::TexinfoXML AUTHOR
Patrice Dumas, <pertusus@free.fr>

18.7 Texinfo::Convert::TexinfoXML COPYRIGHT AND LICENSE
Copyright 2010- Free Software Foundation, Inc. See the source file for all copyright years.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.
19 Texinfo::Convert::Plaintext

19.1 Texinfo::Convert::Plaintext NAME
Texinfo::Convert::Plaintext - Convert Texinfo tree to Plaintext

19.2 Texinfo::Convert::Plaintext SYNOPSIS

my $converter = Texinfo::Convert::Plaintext->converter({'parser' => $parser});

$converter->output($tree);
$converter->convert($tree);
$converter->convert_tree($tree);

19.3 Texinfo::Convert::Plaintext NOTES
The Texinfo Perl module main purpose is to be used in texi2any to convert Texinfo to other formats. There is no promise of API stability.

19.4 Texinfo::Convert::Plaintext DESCRIPTION
Texinfo::Convert::Plaintext converts a Texinfo tree to Plaintext.

19.5 Texinfo::Convert::Plaintext METHODS

$converter = Texinfo::Convert::Plaintext->converter($options)
Initialize converter from Texinfo to Plaintext.

    The $options hash reference holds options for the converter. In this option
    hash reference a Section 3.1 [parser object], page 12, may be associated with the parser key. The other options are Texinfo customization options and a few other options that can be passed to the converter. Most of the customization options are described in the Texinfo manual. Those customization options, when appropriate, override the document content. The parser should not be available directly anymore after getting the associated information.
    See Section 13.1 [Texinfo::Convert::Converter], page 58, for more information.

$converter->output($tree)
    Convert a Texinfo tree $tree and output the result in files as described in the Texinfo manual.

$result = $converter->convert($tree)
    Convert a Texinfo tree $tree and return the resulting output.

$result = $converter->convert_tree($tree)
    Convert a Texinfo tree portion $tree and return the resulting output. This function does not try to output a full document but only portions. For a full document use convert.
19.6 Texinfo::Convert::Plaintext AUTHOR
Patrice Dumas, <pertusus@free.fr>

19.7 Texinfo::Convert::Plaintext COPYRIGHT AND LICENSE
Copyright 2010- Free Software Foundation, Inc. See the source file for all copyright years.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.
Appendix A  Index

%  
%accent_commands ........................................... 1
%all_commands ............................................... 6
%block_commands ........................................... 1
%blockitem_commands ....................................... 2
%brace_code_commands .................................... 2
%brace_commands ........................................... 2
%close_paragraph_commands ................................ 2
%commands_args_number ................................... 3
%contain_basic_inline_commands ......................... 3
%contain_plain_text ....................................... 3
%def_aliases ................................................ 7
%def_commands ............................................... 3
%def_no_var_arg_commands ................................ 7
%default_index_commands ................................ 3
%explained_commands ...................................... 3
%formattable_line_commands .............................. 3
%formatted_line_commands ................................ 3
%formatted_nobrace_commands .............................. 3
%heading_spec_commands .................................. 3
%in_heading_spec_commands ................................. 4
%index_names ................................................ 1
%inline_conditionnal_commands ........................... 4
%inline_format_commands .................................. 4
%letter_no_arg_commands .................................. 4
%line_commands ............................................. 4
%math_commands ............................................. 4
%no_paragraph_commands ................................... 4
%nobrace_commands ......................................... 4
%nobrace_symbol_text ..................................... 7
%non_formattable_block_commands ....................... 4
%preamble_commands ....................................... 4
%preformatted_code_commands ................................
%preformatted_commands ...................................
%ref_commands ................................................
%root_commands .............................................
%sectioning_heading_commands ............................
%small_block_associated_command .........................
%texinfo_output_formats ..................................
%variadic_commands ....................................... 5

A
add_heading_number ....................................... 47
asciiAccent ................................................. 56
asciiAccentFallback ....................................... 56
associate_internal_references ........................... 32

B
brace_no_arg_command .................................... 50

C
check_nodes_are_referred ................................ 32
check_unicode_point_conversion ....................... 50
collect_commands_in_tree ................................ 8
collect_commands_list_in_tree ......................... 8
comma_index_subentries_tree ............................. 61
complete_node_tree_with_menus ........................ 32
complete_tree_nodes_menus ................................ 43
complete_tree_nodes_missing_menu ..................... 43
convert .................................................... 59
convert_accent ............................................. 61
convert_document_sections .............................. 61
convert_to_normalized .................................... 53
convert_to_texinfo ........................................ 46
convert_to_text ........................................... 55
convert_tree ................................................ 59
converter ................................................... 59
converter_defaults ....................................... 59
converter_initial .......................................... 60
create_destination_directory ........................... 61

d
definition_arguments_content ............................ 48
definition_category_tree .................................. 48
determine_files_and_directory ........................... 61
document_error ............................................. 39
document_warn ............................................... 39

e
element_associated_processing_encoding ............. 8
element_is_inline .......................................... 8
elements_directions ....................................... 32
elements_file_directions ................................ 33
encoded_accents ......................................... 51
encoded_input_file_name .................................. 48, 62
encoded_output_file_name ................................ 48, 62
enumerate_item_representation ........................ 8
errors ......................................................... 39
expand_today ............................................... 48
expand_verbatiminclude .................................. 48

F
fill_gaps_in_sectioning .................................. 43
find_innermostAccentContents ............................
find_parent_root_command ................................ 8
float_name_caption ........................................ 62
float_type_number ......................................... 62
floats_information ........................................ 15
force_conf .................................................. 60
format_comment_or_return_end_line ................... 62
Appendix A: Index

G
gdt .................................................. 41
get_conf ............................................. 60
get_node_node_childs_from_sectioning ...... 33
global_commands_information .................. 15
global_information ............................... 14

I
index_entry_sort_string .......................... 33
indices_information .............................. 15
insert_nodes_for_sectioning_commands ......... 43
internal_references_information ............... 15
is_content_empty .................................. 8

L
labels_information ................................. 15
line_error ........................................ 39
line_warn .......................................... 39
locate_include_file .............................. 9

M
menu_to_simple_menu .............................. 44
merge_indices ..................................... 33
move_index_entries_after_items_in_tree ....... 9

N
new_block_command ................................ 33
new_complete_node_menu ......................... 33
new_master_menu .................................. 34
new_node_menu_entry .............................. 34
node_information_filename ....................... 62
nodes_tree ........................................ 34
normalize_node ................................... 53
normalize_top_node_name ......................... 9
normalize_transliterate_texinfo ............... 54
normalized_sectioning_command_filename .... 63
number_floats ..................................... 34

O
output .............................................. 59
output_internal_links ............................ 63, 68

P
parse_texi_file .................................... 14
parse_texi_line .................................... 14
parse_texi_piece ................................... 14
parse_texi_text .................................... 14
Parser initialization .............................. 13
pgdt ................................................ 42
present_bug_message .............................. 63
protect_colon_in_tree ............................ 9
protect_comma_in_tree ............................ 9
protect_first_parenthesis ....................... 9
protect_hashtag_at_line_beginning ............. 44
protect_node_after_label_in_tree .............. 9

R
reference_to_arg_in_tree ......................... 44
regenerate_master_menu ......................... 44
registered_errors ............................... 14
relate_index_entries_to_table_items_in_tree .... 9
replace_convert_substrings ..................... 42

S
section_level ...................................... 9
section_level_adjusted_command_name ........ 34
sectioning_structure ............................. 34
set_conf ........................................... 60
set_global_document_command ................... 10
set_global_document_commands ................. 63
set_informative_command_value .................. 10
set_menus_node_directions ..................... 35
set_menus_to_simple_menu ....................... 44
set_output_encodings ............................. 10
setup_index_entry_keys_formatting ............ 35
sort_indices ...................................... 35
split_by_node ..................................... 35
split_by_section .................................. 36
split_custom_heading_command_contents ....... 10
split_pages ........................................ 36
string_width ...................................... 51

T
table_item_content_tree ......................... 63
Texinfo tree element extra key .................. 24
Texinfo tree element structure ................ 17
Texinfo tree elements ............................. 17
Texinfo::Convert::Converter initialization .... 50
Texinfo::Parser::parser ........................... 43
Texinfo::Report::new .............................. 38
text_accents ...................................... 56
top_node_filename ................................ 63
transliterate_protect_filename ................. 63
transliterate_texinfo ............................. 54
trim_spaces_comment_from_content ............. 10
Appendix A: Index

U
unicoded_accen

unicode_point_decoded_in_encoding

unicode_text

V
valid_option
valid_tree_transformation

W
warn_non_empty_parts

X
xml_accent
xml_accents
xml_comment
xml_format_text_with_numeric_entities
xml_numeric_entity_accent
xml_protect_text