The Declt Reference Manual

Documentation Extractor from Common Lisp to Texinfo, version 2.3 "Robert April"

Didier Verna <didier@didierverna.net>

This manual was generated automatically by Declt 2.3 "Robert April" on Sat Dec 02 13:00:17 2017 GMT+1.

Table of Contents

Copying

1 Introduction

`Declt` (pronounce "dec'let") is a reference manual generator for Common-Lisp libraries. It works by loading an ASDF system and introspecting its contents. The generated documentation contains the description of the system itself and its local dependencies (other systems in the same distribution): components (modules and files), packages and definitions found in those packages.

Exported and internal definitions are listed separately. This allows the reader to have a quick view on the library's public API. Within each section, definitions are sorted lexicographically.

In addition to ASDF system components and packages, `Declt` documents the following definitions: constants, special variables, symbol macros, macros, `setf` expanders, compiler macros, functions (including `setf` ones), generic functions and methods (including `setf` ones), method combinations, conditions, structures, classes and types.

The generated documentation includes every possible bit of information that introspecting can provide: documentation strings, lambda lists (including qualifiers and specializers where appropriate), slots (including type, allocation and initialization arguments), definition source file *etc.*

Every documented item provides a full set of cross-references to related items: ASDF component dependencies, parents and children, classes direct methods, super and subclasses, slot readers and writers, `setf` expanders access and update functions *etc.*

Finally, `Declt` produces exhaustive and multiple-entry indexes for every documented item.

Reference manuals are generated in Texinfo format (compatible, but not requiring Texinfo 5). From there it is possible to produce readable / printable output in info, HTML, PDF, DVI and PostScript with tools such as `makeinfo`, `texi2dvi` or `texi2pdf`.

The `Declt` Reference Manual, which you are currently reading, is the primary example of documentation generated by `Declt` itself. See *The Declt User Manual*, for a more human-readable guide to using `Declt`.

2 Systems

The main system appears first, followed by any subsystem dependency.

2.1 net.didierverna.declt

Long Name
Documentation Extractor from Common Lisp to Texinfo

Author Didier Verna

Contact didier@didierverna.net

Home Page
http://www.lrde.epita.fr/~didier/software/lisp/misc.php#declt

Source Control
https://github.com/didierverna/declt

License BSD

Description
A reference manual generator for Common Lisp libraries

Long Description
Declt (pronounce dec'let) is a reference manual generator for Common Lisp.
It extracts and formats documentation from ASDF systems, including the system itself, its local dependencies (subsystems), components, packages and an extensive list of definitions (variables, functions etc.). The formatted documentation comes with full indexing and cross-references.

Reference manuals are generated in Texinfo format which can subsequently be converted into info, HTML, DVI, PostScript or PDF.

Version 2.3

If Feature sbcl

Dependencies
- [net.didierverna.declt.setup], page 6, (system)
- [net.didierverna.declt.core], page 5, (system)

Source [net.didierverna.declt.asd], page 9, (file)

2.2 net.didierverna.declt.core

Long Name
Documentation Extractor from Common Lisp to Texinfo, core library

Author Didier Verna

Contact didier@didierverna.net

Home Page
http://www.lrde.epita.fr/~didier/software/lisp/misc.php#declt

Source Control
https://github.com/didierverna/declt

License BSD

Description
A reference manual generator for Common Lisp libraries

Long Description
Declt's core functionality. For a more complete description of Declt, see the net.didierverna.declt system.

Version 2.3

If Feature `sbcl`

Dependencies
- [`net.didierverna.declt.setup`], page 6, (system)
- `sb-introspect` (for feature `sbcl`)

Source [`net.didierverna.declt.core.asd`], page 9, (file)

Components
- [`quickutil.lisp`], page 9, (file)
- [`meta.lisp`], page 9, (file)
- [`src`], page 7, (module)

2.3 `net.didierverna.declt.setup`

Long Name
Documentation Extractor from Common Lisp to Texinfo, setup library

Author Didier Verna

Contact `didier@didierverna.net`

Home Page
`http://www.lrde.epita.fr/~didier/software/lisp/misc.php#declt`

Source Control
`https://github.com/didierverna/declt`

License BSD

Description
Declt's preload setup library

Long Description
Declt's setup library provides support for various preload configuration parameters. For a more complete description of Declt, see the net.didierverna.declt system.

Source [`net.didierverna.declt.setup.asd`], page 9, (file)

Component
[`setup.lisp`], page 28, (file)

3 Modules

Modules are listed depth-first from the system components tree.

3.1 net.didierverna.declt.core/src

Dependency
[meta.lisp], page 9, (file)

Parent [net.didierverna.declt.core], page 5, (system)

Location core/src/

Components
- [util], page 7, (module)
- [item], page 7, (module)
- [doc], page 7, (module)
- [declt.lisp], page 27, (file)

3.2 net.didierverna.declt.core/src/util

Parent [src], page 7, (module)

Location core/src/util/

Components
- [misc.lisp], page 10, (file)
- [asdf.lisp], page 10, (file)
- [texi.lisp], page 10, (file)

3.3 net.didierverna.declt.core/src/item

Dependency
[util], page 7, (module)

Parent [src], page 7, (module)

Location core/src/item/

Components
- [item.lisp], page 12, (file)
- [symbol.lisp], page 12, (file)
- [package.lisp], page 23, (file)
- [asdf.lisp], page 23, (file)

3.4 net.didierverna.declt.core/src/doc

Dependency
[item], page 7, (module)

Parent [src], page 7, (module)

Location core/src/doc/

Components
- [doc.lisp], page 24, (file)
- [symbol.lisp], page 24, (file)
- [package.lisp], page 26, (file)
- [asdf.lisp], page 26, (file)

4 Files

Files are sorted by type and then listed depth-first from the systems components trees.

4.1 Lisp

4.1.1 net.didierverna.declt.asd

Location net.didierverna.declt.asd

Systems [net.didierverna.declt], page 5, (system)

4.1.2 net.didierverna.declt.core.asd

Location core/net.didierverna.declt.core.asd

Systems [net.didierverna.declt.core], page 5, (system)

4.1.3 net.didierverna.declt.setup.asd

Location setup/net.didierverna.declt.setup.asd

Systems [net.didierverna.declt.setup], page 6, (system)

4.1.4 net.didierverna.declt.core/quickutil.lisp

Parent [net.didierverna.declt.core], page 5, (system)

Location core/quickutil.lisp

Packages [quickutil], page 29,

Exported Definitions

- [when-let], page 47, (macro)
- [when-let*], page 48, (macro)

4.1.5 net.didierverna.declt.core/meta.lisp

Dependency

[quickutil.lisp], page 9, (file)

Parent [net.didierverna.declt.core], page 5, (system)

Location core/meta.lisp

Packages [net.didierverna.declt], page 29,

Exported Definitions

[nickname-package], page 49, (function)

Internal Definitions

- [*readtable*], page 50, (special variable)
- [clindent], page 61, (function)
- [defindent], page 54, (macro)
- [generate-quickutils], page 69, (function)
- [i-reader], page 71, (function)
- [in-readtable], page 54, (macro)
- [tilde-reader], page 87, (function)

4.1.6 net.didierverna.declt.core/src/util/misc.lisp

Parent [util], page 7, (module)

Location core/src/util/misc.lisp

Internal Definitions

- [current-time-string], page 66, (function)
- [defsystem-dependencies], page 67, (function)
- [endpush], page 54, (macro)
- [parse-contact(s)], page 79, (function)
- [parse-contact-string], page 79, (function)

4.1.7 net.didierverna.declt.core/src/util/asdf.lisp

Dependency

[misc.lisp], page 10, (file)

Parent [util], page 7, (module)

Location core/src/util/asdf.lisp

Internal Definitions

- [components], page 62, (function)
- [lisp-components], page 72, (function)
- [module-components], page 77, (function)
- [relative-location], page 80, (function)
- [sub-component-p], page 86, (function)
- [subsystems], page 86, (function)
- [system-base-name], page 86, (function)
- [system-dependencies], page 86, (function)
- [system-dependency-subsystem], page 103, (generic function)
- [system-dependency-subsystem], page 103, (method)
- [system-dependency-subsystem], page 103, (method)
- [system-directory], page 86, (function)
- [system-file-name], page 87, (function)
- [system-file-type], page 87, (function)

4.1.8 net.didierverna.declt.core/src/util/texi.lisp

Dependency

[asdf.lisp], page 10, (file)

Parent [util], page 7, (module)

Location core/src/util/texi.lisp

Internal Definitions

- [*section-names*], page 50, (special variable)
- [@anchor], page 55, (function)
- [@defclass], page 50, (macro)
- [@defcombination], page 50, (macro)
- [@defcompilermacro], page 51, (macro)

- [(setf node-before-menu-contents)], page 78, (function)
- [node-children], page 78, (function)
- [(setf node-children)], page 78, (function)
- [node-name], page 78, (function)
- [(setf node-name)], page 78, (function)
- [node-next], page 78, (function)
- [(setf node-next)], page 78, (function)
- [node-p], page 78, (function)
- [node-previous], page 78, (function)
- [(setf node-previous)], page 78, (function)
- [node-section-name], page 78, (function)
- [(setf node-section-name)], page 78, (function)
- [node-section-type], page 78, (function)
- [(setf node-section-type)], page 78, (function)
- [node-synopsis], page 78, (function)
- [(setf node-synopsis)], page 78, (function)
- [node-up], page 78, (function)
- [(setf node-up)], page 78, (function)
- [pretty-specializer], page 100, (generic function)
- [pretty-specializer], page 100, (method)
- [pretty-specializer], page 100, (method)
- [read-next-line], page 79, (function)
- [render-lambda-list], page 81, (function)
- [render-node], page 81, (function)
- [render-text], page 82, (function)
- [render-to-string], page 55, (macro)
- [render-top-node], page 82, (function)
- [reveal], page 101, (generic function)
- [reveal], page 101, (method)
- [reveal], page 101, (method)

4.1.9 net.didierverna.declt.core/src/item/item.lisp

Parent [item], page 7, (module)

Location core/src/item/item.lisp

Internal Definitions

- [docstring], page 91, (generic function)
- [source], page 101, (generic function)
- [type-name], page 103, (generic function)

4.1.10 net.didierverna.declt.core/src/item/symbol.lisp

Dependency

[item.lisp], page 12, (file)

Parent [item], page 7, (module)

Location `core/src/item/symbol.lisp`

Internal Definitions

- [`*categories*`], page 50, (special variable)
- [`accessor-definition`], page 105, (structure)
- [`accessor-definition-access-expander`], page 56, (function)
- [`(setf accessor-definition-access-expander)`], page 56, (function)
- [`accessor-definition-foreignp`], page 56, (function)
- [`(setf accessor-definition-foreignp)`], page 56, (function)
- [`accessor-definition-function`], page 56, (function)
- [`(setf accessor-definition-function)`], page 56, (function)
- [`accessor-definition-p`], page 56, (function)
- [`accessor-definition-symbol`], page 56, (function)
- [`(setf accessor-definition-symbol)`], page 56, (function)
- [`accessor-definition-update-expander`], page 57, (function)
- [`(setf accessor-definition-update-expander)`], page 57, (function)
- [`accessor-definition-writer`], page 57, (function)
- [`(setf accessor-definition-writer)`], page 57, (function)
- [`accessor-method-definition`], page 106, (structure)
- [`accessor-method-definition-foreignp`], page 57, (function)
- [`(setf accessor-method-definition-foreignp)`], page 57, (function)
- [`accessor-method-definition-method`], page 57, (function)
- [`(setf accessor-method-definition-method)`], page 57, (function)
- [`accessor-method-definition-p`], page 57, (function)
- [`accessor-method-definition-symbol`], page 57, (function)
- [`(setf accessor-method-definition-symbol)`], page 57, (function)
- [`accessor-method-definition-writer`], page 57, (function)
- [`(setf accessor-method-definition-writer)`], page 57, (function)
- [`add-definition`], page 58, (function)
- [`add-symbol-definition`], page 59, (function)
- [`add-symbol-definitions`], page 59, (function)
- [`boolean-to-feature-expression`], page 59, (function)
- [`category-definitions`], page 89, (generic function)
- [`category-definitions`], page 89, (method)
- [`category-definitions`], page 89, (method)
- [`category-definitions`], page 89, (method)
- [`category-definitions`], page 89, (method)
- [`category-definitions`], page 89, (method)
- [`category-definitions`], page 90, (method)
- [`category-definitions`], page 90, (method)
- [`category-definitions`], page 90, (method)
- [`class-definition`], page 106, (structure)
- [`class-definition-children`], page 59, (function)
- [`(setf class-definition-children)`], page 59, (function)

4.1.11 net.didierverna.declt.core/src/item/package.lisp

Dependency
[symbol.lisp], page 12, (file)

Parent [item], page 7, (module)

Location core/src/item/package.lisp

Internal Definitions
- [definition-package-definitions], page 91, (generic function)
- [definition-package-definitions], page 91, (method)
- [definition-package-definitions], page 91, (method)
- [definition-package-definitions], page 91, (method)
- [definition-package-definitions], page 91, (method)
- [definition-package-definitions], page 91, (method)
- [definition-package-definitions], page 91, (method)
- [docstring], page 91, (method)
- [name], page 99, (method)
- [package-definitions], page 79, (function)
- [package-external-symbols], page 79, (function)
- [package-internal-symbols], page 79, (function)
- [source], page 102, (method)
- [type-name], page 104, (method)

4.1.12 net.didierverna.declt.core/src/item/asdf.lisp

Dependency
[package.lisp], page 23, (file)

Parent [item], page 7, (module)

Location core/src/item/asdf.lisp

Internal Definitions
- [definition-file-definitions], page 90, (generic function)
- [definition-file-definitions], page 90, (method)
- [definition-file-definitions], page 90, (method)
- [definition-file-definitions], page 90, (method)
- [definition-file-definitions], page 90, (method)
- [definition-file-definitions], page 91, (method)
- [definition-file-definitions], page 91, (method)
- [file-definitions], page 67, (function)
- [file-packages], page 67, (function)
- [lisp-pathnames], page 72, (function)
- [name], page 99, (method)
- [name], page 99, (method)
- [system-external-symbols], page 87, (function)
- [system-internal-symbols], page 87, (function)
- [system-packages], page 87, (function)

- [type-name], page 103, (method)
- [type-name], page 103, (method)
- [type-name], page 103, (method)

4.1.13 net.didierverna.declt.core/src/doc/doc.lisp

Parent [doc], page 7, (module)

Location core/src/doc/doc.lisp

Internal Definitions

- [anchor], page 59, (function)
- [anchor-and-index], page 59, (function)
- [anchor-name], page 89, (generic function)
- [anchor-name], page 89, (method)
- [context], page 109, (structure)
- [context-directory], page 63, (function)
- [context-external-definitions], page 63, (function)
- [(setf context-external-definitions)], page 63, (function)
- [context-hyperlinksp], page 63, (function)
- [(setf context-hyperlinksp)], page 63, (function)
- [context-internal-definitions], page 63, (function)
- [(setf context-internal-definitions)], page 63, (function)
- [context-p], page 63, (function)
- [context-packages], page 63, (function)
- [(setf context-packages)], page 63, (function)
- [context-systems], page 64, (function)
- [(setf context-systems)], page 64, (function)
- [copy-context], page 64, (function)
- [document], page 93, (generic function)
- [index], page 96, (generic function)
- [make-context], page 74, (function)
- [reference], page 100, (generic function)
- [render-docstring], page 80, (function)
- [render-location], page 81, (function)
- [render-references], page 82, (function)
- [render-source], page 82, (function)
- [title], page 103, (generic function)
- [title], page 103, (method)

4.1.14 net.didierverna.declt.core/src/doc/symbol.lisp

Dependency

[doc.lisp], page 24, (file)

Parent [doc], page 7, (module)

Location core/src/doc/symbol.lisp

Internal Definitions

- [index], page 98, (method)
- [index], page 98, (method)
- [index], page 98, (method)
- [index], page 98, (method)
- [reference], page 101, (method)
- [render-classoid], page 54, (macro)
- [render-combination], page 54, (macro)
- [render-definition-core], page 80, (function)
- [render-external-definitions-references], page 80, (function)
- [render-funcoid], page 54, (macro)
- [render-headline], page 81, (function)
- [render-initargs], page 81, (function)
- [render-internal-definitions-references], page 81, (function)
- [render-method], page 54, (macro)
- [render-method-combination], page 81, (function)
- [render-slot], page 82, (function)
- [render-slot-property], page 82, (function)
- [render-slots], page 82, (function)
- [render-varoid], page 55, (macro)

4.1.15 net.didierverna.declt.core/src/doc/package.lisp

Dependency
[symbol.lisp], page 24, (file)

Parent [doc], page 7, (module)

Location core/src/doc/package.lisp

Internal Definitions
- [add-packages-node], page 58, (function)
- [anchor-name], page 89, (method)
- [document], page 93, (method)
- [index], page 97, (method)
- [reference], page 101, (method)
- [render-use-list], page 82, (function)
- [title], page 103, (method)

4.1.16 net.didierverna.declt.core/src/doc/asdf.lisp

Dependency
[package.lisp], page 26, (file)

Parent [doc], page 7, (module)

Location core/src/doc/asdf.lisp

Internal Definitions
- [add-files-node], page 58, (function)
- [add-modules-node], page 58, (function)
- [add-systems-node], page 59, (function)

- [anchor-name], page 89, (method)
- [document], page 93, (method)
- [document], page 93, (method)
- [document], page 93, (method)
- [document], page 93, (method)
- [document], page 93, (method)
- [file-node], page 67, (function)
- [index], page 96, (method)
- [index], page 96, (method)
- [index], page 97, (method)
- [index], page 97, (method)
- [index], page 97, (method)
- [index], page 97, (method)
- [index], page 97, (method)
- [index], page 97, (method)
- [module-node], page 77, (function)
- [reference], page 100, (method)
- [reference], page 101, (method)
- [reference], page 101, (method)
- [reference], page 101, (method)
- [reference-component], page 80, (function)
- [render-dependencies], page 80, (function)
- [render-dependency], page 101, (generic function)
- [render-dependency], page 101, (method)
- [render-dependency], page 101, (method)
- [render-packages-references], page 81, (function)
- [system-node], page 87, (function)
- [title], page 103, (method)
- [virtual-path], page 105, (generic function)
- [virtual-path], page 105, (method)
- [virtual-path], page 105, (method)

4.1.17 net.didierverna.declt.core/src/declt.lisp

Dependency
[doc], page 7, (module)

Parent [src], page 7, (module)

Location core/src/declt.lisp

Exported Definitions
[declt], page 48, (function)

Internal Definitions
- [*licenses*], page 50, (special variable)
- [add-definitions], page 58, (function)
- [add-external-definitions], page 58, (function)

- [add-internal-definitions], page 58, (function)
- [add-packages], page 58, (function)
- [render-header], page 80, (function)

4.1.18 net.didierverna.declt.setup/setup.lisp

Parent [net.didierverna.declt.setup], page 6, (system)

Location setup/setup.lisp

Packages [net.didierverna.declt.setup], page 46,

Exported Definitions

- [*release-major-level*], page 47, (special variable)
- [*release-minor-level*], page 47, (special variable)
- [*release-name*], page 47, (special variable)
- [*release-status*], page 47, (special variable)
- [*release-status-level*], page 47, (special variable)
- [configuration], page 48, (function)
- [configure], page 48, (function)
- [version], page 49, (function)

Internal Definitions

- [%version], page 55, (function)
- [*configuration*], page 50, (special variable)
- [release-status-number], page 80, (function)

5 Packages

Packages are listed by definition order.

5.1 quickutil

Package that contains the actual utility functions.

Source [quickutil.lisp], page 9, (file)

Nickname qtl

Use List common-lisp

Exported Definitions

- [when-let], page 47, (macro)
- [when-let*], page 48, (macro)

5.2 net.didierverna.declt

The Documentation Extractor from Common Lisp to Texinfo package.

Source [meta.lisp], page 9, (file)

Nickname declt

Use List

- [net.didierverna.declt.setup], page 46,
- common-lisp

Exported Definitions

- [declt], page 48, (function)
- [nickname-package], page 49, (function)

Internal Definitions

- [*categories*], page 50, (special variable)
- [*licenses*], page 50, (special variable)
- [*readtable*], page 50, (special variable)
- [*section-names*], page 50, (special variable)
- [@anchor], page 55, (function)
- [@defclass], page 50, (macro)
- [@defcombination], page 50, (macro)
- [@defcompilermacro], page 51, (macro)
- [@defcond], page 51, (macro)
- [@defconstant], page 51, (macro)
- [@deffn], page 51, (macro)
- [@deffnx], page 55, (function)
- [@defgeneric], page 51, (macro)
- [@defgenericx], page 55, (function)
- [@defmacro], page 51, (macro)
- [@defmethod], page 51, (macro)
- [@defmethodx], page 55, (function)
- [@defsetf], page 52, (macro)

5.3 net.didierverna.declt.setup

Documentation Extractor from Common Lisp to Texinfo.

Source [setup.lisp], page 28, (file)

Use List common-lisp

Used By List
[net.didierverna.declt], page 29,

Exported Definitions

- [*release-major-level*], page 47, (special variable)
- [*release-minor-level*], page 47, (special variable)
- [*release-name*], page 47, (special variable)
- [*release-status*], page 47, (special variable)
- [*release-status-level*], page 47, (special variable)
- [configuration], page 48, (function)
- [configure], page 48, (function)
- [version], page 49, (function)

Internal Definitions

- [%version], page 55, (function)
- [*configuration*], page 50, (special variable)
- [release-status-number], page 80, (function)

6 Definitions

Definitions are sorted by export status, category, package, and then by lexicographic order.

6.1 Exported definitions

6.1.1 Special variables

`*release-major-level*` [Special Variable]
The major level of this release.

Package [`net.didierverna.declt.setup`], page 46,

Source [`setup.lisp`], page 28, (file)

`*release-minor-level*` [Special Variable]
The minor level of this release.

Package [`net.didierverna.declt.setup`], page 46,

Source [`setup.lisp`], page 28, (file)

`*release-name*` [Special Variable]
The name of this release.
The general naming theme for Declt is "Star Trek characters".

Package [`net.didierverna.declt.setup`], page 46,

Source [`setup.lisp`], page 28, (file)

`*release-status*` [Special Variable]
The status of this release.

Package [`net.didierverna.declt.setup`], page 46,

Source [`setup.lisp`], page 28, (file)

`*release-status-level*` [Special Variable]
The status level of this release.

Package [`net.didierverna.declt.setup`], page 46,

Source [`setup.lisp`], page 28, (file)

6.1.2 Macros

`when-let` *BINDINGS* **&body** *FORMS* [Macro]
Creates new variable bindings, and conditionally executes FORMS.

BINDINGS must be either single binding of the form:

(variable initial-form)

or a list of bindings of the form:

((variable-1 initial-form-1)
(variable-2 initial-form-2)
...

(variable-n initial-form-n))

All initial-forms are executed sequentially in the specified order. Then all the variables are bound to the corresponding values.

If all variables were bound to true values, then FORMS are executed as an implicit PROGN.

Package [quickutil], page 29,

Source [quickutil.lisp], page 9, (file)

`when-let*` *BINDINGS* **&body** *FORMS* [Macro]

Creates new variable bindings, and conditionally executes FORMS.

BINDINGS must be either single binding of the form:

(variable initial-form)

or a list of bindings of the form:

((variable-1 initial-form-1)
(variable-2 initial-form-2)
...
(variable-n initial-form-n))

Each initial-form is executed in turn, and the variable bound to the corresponding value. Initial-form expressions can refer to variables previously bound by the WHEN-LET*.

Execution of WHEN-LET* stops immediately if any initial-form evaluates to NIL. If all initial-forms evaluate to true, then FORMS are executed as an implicit PROGN.

Package [quickutil], page 29,

Source [quickutil.lisp], page 9, (file)

6.1.3 Functions

`configuration` *KEY* [Function]

Return KEY's value in the current Declt configuration.

Package [net.didierverna.declt.setup], page 46,

Source [setup.lisp], page 28, (file)

`configure` *KEY VALUE* [Function]

Set KEY to VALUE in the current Declt configuration.

Package [net.didierverna.declt.setup], page 46,

Source [setup.lisp], page 28, (file)

`declt` *SYSTEM-NAME* **&key** *LIBRARY TAGLINE VERSION CONTACT COPYRIGHT LICENSE INTRODUCTION CONCLUSION TEXI-FILE INFO-FILE HYPERLINKS DECLT-NOTICE* **&aux** *SYSTEM TEXI-NAME CURRENT-TIME-STRING CONTACT-NAMES CONTACT-EMAILS* [Function]

Generate a reference manual in Texinfo format for ASDF SYSTEM-NAME.

- SYSTEM-NAME is a system designator, as understood by ASDF.
- LIBRARY: defaults to SYSTEM-NAME.
- TAGLINE: defaults to the system's long name or description.
- VERSION: defaults to the system version.
- CONTACT: defaults to the system's maintainer(s) and author(s). If the library doesn't provide that information, "John Doe" is used. If you
provide it by hand, use an author string or a list of such. An author string contains a name, optionally followed by an <email@ddress>.

- COPYRIGHT: defaults to the current year. Possible values are nil or any string.
- LICENSE: defaults to nil (possible other values are: :mit, :bsd, :gpl and :lgpl).
- INTRODUCTION: contents for an optional introduction chapter.
- CONCLUSION: contents for an optional conclusion chapter.

- TEXI-FILE: full path to the Texinfo file. Defaults to LIBRARY.texi.
- INFO-FILE: info file basename sans extension. The default is built from TEXI-FILE.
- HYPERLINKS: whether to create hyperlinks to files or directories in the reference manual. Note that those links being specific to the machine on which the manual was generated, it is preferable to keep it to NIL for creating reference manuals meant to be put online.
- DECLT-NOTICE: small paragraph about automatic manual generation by
Declt. Possible values are nil, :short and :long (the default).

Both the INTRODUCTION and the CONCLUSION may contain Texinfo directives (no post-processing will occur). All other textual material is considered raw text and will be properly escaped for Texinfo.

Package [net.didierverna.declt], page 29,

Source [declt.lisp], page 27, (file)

`nickname-package` **&optional** *NICKNAME* [Function]

Add NICKNAME (:DECLT by default) to the :NET.DIDIERVERNA.DECLT package.

Package [net.didierverna.declt], page 29,

Source [meta.lisp], page 9, (file)

`version` **&optional** *TYPE* [Function]

Return the current version of Declt.
TYPE can be one of :number, :short or :long.

A version number is computed as major*10000 + minor*100 + patchlevel, leaving two digits for each level. Alpha, beta and rc status are ignored in version numbers.

A short version is something like 1.3{a,b,rc}4, or 1.3.4 for patchlevel. Alpha, beta or rc levels start at 1. Patchlevels start at 0 but are ignored
in the output, so that 1.3.0 appears as just 1.3.

A long version is something like
1.3 {alpha,beta,release candidate,patchlevel} 4 "James T. Kirk". As for
the short version, a patchlevel of 0 is ignored in the output.

Package [net.didierverna.declt.setup], page 46,

Source [setup.lisp], page 28, (file)

6.2 Internal definitions

6.2.1 Special variables

`*categories*` [Special Variable]
The list of definition categories.
Each category is of type (:KEYWORD DESCRIPTION-STRING).

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`*configuration*` [Special Variable]
The Declt configuration settings.
This variable contains a property list of configuration options.
Current options are:
- :swank-eval-in-emacs (Boolean)

See section A.1 of the user manual for more information.

Package [`net.didierverna.declt.setup`], page 46,

Source [`setup.lisp`], page 28, (file)

`*licenses*` [Special Variable]

Package [`net.didierverna.declt`], page 29,

Source [`declt.lisp`], page 27, (file)

`*readtable*` [Special Variable]
The Declt readtable.

Package [`net.didierverna.declt`], page 29,

Source [`meta.lisp`], page 9, (file)

`*section-names*` [Special Variable]
The numbered, unumbered and appendix section names sorted by level.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

6.2.2 Macros

`@defclass` *NAME* **&body** *BODY* [Macro]
Execute BODY within a @deftp {Class} NAME environment.
NAME is escaped for Texinfo prior to rendering.
BODY should render on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@defcombination` *NAME KIND* **&body** *BODY* [Macro]
Execute BODY within a @deftp {KIND Method Combination} NAME environment.
NAME is escaped for Texinfo prior to rendering.
BODY should render on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@defcompilermacro` *NAME LAMBDA-LIST* **&body** *BODY* [Macro]
Execute BODY within a @deffn {Compiler Macro} NAME LAMBDA-LIST environment.
NAME and LAMBDA-LIST are escaped for Texinfo prior to rendering.
BODY should render on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@defcond` *NAME* **&body** *BODY* [Macro]
Execute BODY within a @deftp {Condition} NAME environment.
NAME is escaped for Texinfo prior to rendering.
BODY should render on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@defconstant` *NAME* **&body** *BODY* [Macro]
Execute BODY within a @defvr {Constant} NAME environment.
NAME is escaped for Texinfo prior to rendering.
BODY should render on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@deffn` (*CATEGORY NAME LAMBDA-LIST* **&optional** *SPECIALIZERS QUALIFIERS*) **&body** *BODY* [Macro]
Execute BODY within a @deffn CATEGORY NAME LAMBDA-LIST environment.
CATEGORY, NAME, LAMBDA-LIST, SPECIALIZERS and QUALIFIERS are escaped for Texinfo prior to rendering.
BODY should render on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@defgeneric` *NAME LAMBDA-LIST* **&body** *BODY* [Macro]
Execute BODY within a @deffn {Generic Function} NAME LAMBDA-LIST environment.
NAME and LAMBDA-LIST are escaped for Texinfo prior to rendering.
BODY should render on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@defmacro` *NAME LAMBDA-LIST* **&body** *BODY* [Macro]
Execute BODY within a @deffn Macro NAME LAMBDA-LIST environment.
NAME and LAMBDA-LIST are escaped for Texinfo prior to rendering.
BODY should render on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@defmethod` *NAME LAMBDA-LIST SPECIALIZERS QUALIFIERS* **&body** *BODY* [Macro]
Execute BODY within a @deffn {Method} NAME LAMBDA-LIST environment.
NAME, LAMBDA-LIST, SPECIALIZERS and QUALIFIERS are escaped for Texinfo prior

to rendering.
BODY should render on *standard-output*.

Package [net.didierverna.declt], page 29,

Source [texi.lisp], page 10, (file)

`@defsetf` *NAME LAMBDA-LIST* **&body** *BODY* [Macro]
Execute BODY within a @deffn {Setf Expander} NAME LAMBDA-LIST environment.
NAME and LAMBDA-LIST are escaped for Texinfo prior to rendering.
BODY should render on *standard-output*.

Package [net.didierverna.declt], page 29,

Source [texi.lisp], page 10, (file)

`@defslot` *NAME* **&body** *BODY* [Macro]
Execute BODY within a @defvr {Slot} Name environment.
NAME is escaped for Texinfo prior to rendering.
BODY should render on *standard-output*.

Package [net.didierverna.declt], page 29,

Source [texi.lisp], page 10, (file)

`@defspecial` *NAME* **&body** *BODY* [Macro]
Execute BODY within a @defvr {Special Variable} NAME environment.
NAME is escaped for Texinfo prior to rendering.
BODY should render on *standard-output*.

Package [net.didierverna.declt], page 29,

Source [texi.lisp], page 10, (file)

`@defstruct` *NAME* **&body** *BODY* [Macro]
Execute BODY within a @deftp {Structure} NAME environment.
NAME is escaped for Texinfo prior to rendering.
BODY should render on *standard-output*.

Package [net.didierverna.declt], page 29,

Source [texi.lisp], page 10, (file)

`@defsymbolmacro` *NAME* **&body** *BODY* [Macro]
Execute BODY within a @defvr {Symbol Macro} NAME environment.
NAME is escaped for Texinfo prior to rendering.
BODY should render on *standard-output*.

Package [net.didierverna.declt], page 29,

Source [texi.lisp], page 10, (file)

`@deftp` (*CATEGORY NAME* **&optional** *LAMBDA-LIST*) **&body** *BODY* [Macro]
Execute BODY within a @deftp {CATEGORY} NAME [LAMBDA-LIST] environment.
CATEGORY, NAME and LAMBDA-LIST are escaped for Texinfo prior to rendering. BODY should render on *standard-output*.

Package [net.didierverna.declt], page 29,

Source [texi.lisp], page 10, (file)

`@deftype` (*NAME* **&optional** *LAMBDA-LIST*) **&body** *BODY* [Macro]
Execute BODY within a @deftp {Type} NAME [LAMBDA-LIST] environment.
NAME and LAMBDA-LIST are escaped for Texinfo prior to rendering.
BODY should render on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@defun` *NAME LAMBDA-LIST* **&body** *BODY* [Macro]
Execute BODY within a @deffn Function NAME LAMBDA-LIST environment.
NAME and LAMBDA-LIST are escaped for Texinfo prior to rendering.
BODY should render on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@defvr` *CATEGORY NAME* **&body** *BODY* [Macro]
Execute BODY within a @defvr {CATEGORY} NAME environment.
CATEGORY and NAME are escaped for Texinfo prior to rendering.
BODY should render on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@item` **&body** *BODY* [Macro]
Execute BODY within an itemize @item.
BODY should render on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@itemize` (**&optional** *KIND*) **&body** *BODY* [Macro]
Execute BODY within an @itemize KIND environment.
BODY should render on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@multitable` (**&rest** *FRACTIONS*) **&body** *BODY* [Macro]
Execute BODY within a @multitable environment.
FRACTIONS is the list of column fractions to use.
BODY should render on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@table` (**&optional** *KIND*) **&body** *BODY* [Macro]
Execute BODY within a @table KIND environment.
BODY should render on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@tableitem` *TITLE* **&body** *BODY* [Macro]
Execute BODY within a table @item TITLE.
BODY should render on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`defindent` *SYMBOL INDENT* [Macro]
Set SYMBOL's indentation to INDENT in (X)Emacs.
SYMBOL and INDENT need not be quoted.
See CLINDENT for more information.

Package [`net.didierverna.declt`], page 29,

Source [`meta.lisp`], page 9, (file)

`endpush` *OBJECT PLACE* [Macro]
Push OBJECT at the end of PLACE.

Package [`net.didierverna.declt`], page 29,

Source [`misc.lisp`], page 10, (file)

`in-readtable` *NAME* [Macro]
Set the current readtable to the value of NAME::*READTABLE*.

Package [`net.didierverna.declt`], page 29,

Source [`meta.lisp`], page 9, (file)

`render-classoid` *KIND CLASSOID CONTEXT* **&body** *BODY* [Macro]
Render CLASSOID's definition of KIND in CONTEXT.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 24, (file)

`render-combination` *KIND COMBINATION CONTEXT* **&body** *BODY* [Macro]
Render method COMBINATION's definition of KIND in CONTEXT.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 24, (file)

`render-funcoid` *KIND funcoid*(*s*) *CONTEXT* **&body** *BODY* [Macro]
Render FUNCOID(S) definition of KIND in CONTEXT.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 24, (file)

`render-method` *method*(*s*) *CONTEXT GENERIC-SOURCE* [Macro]
Render METHOD(S) definition in CONTEXT.
GENERIC-SOURCE is the source of the generic function. METHOD(S) sources are not advertised if they are the same as GENERIC-SOURCE.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 24, (file)

`render-to-string` **&body** *BODY* [Macro]
Execute BODY with *standard-output* redirected to a string.
Return that string.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`render-varoid` *KIND VAROID CONTEXT* **&body** *BODY* [Macro]
Render VAROID definition of KIND in CONTEXT.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 24, (file)

6.2.3 Functions

`%version` *TYPE MAJOR MINOR STATUS LEVEL NAME* [Function]

Package [`net.didierverna.declt.setup`], page 46,

Source [`setup.lisp`], page 28, (file)

`@anchor` *ANCHOR* [Function]
Render ANCHOR as an @anchor.
ANCHOR is escaped for Texinfo prior to rendering.
Rendering is done on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@deffnx` *CATEGORY NAME LAMBDA-LIST* **&optional** *SPECIALIZERS QUALIFIERS* [Function]
Render @deffnx CATEGORY NAME LAMBDA-LIST on *standard-output*.
CATEGORY, NAME, LAMBDA-LIST, SPECIALIZERS and QUALIFIERS are escaped for Texinfo prior to rendering.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@defgenericx` *NAME LAMBDA-LIST* [Function]
Render @deffnx {Generic Function} NAME LAMBDA-LIST on *standard-output*

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@defmethodx` *NAME LAMBDA-LIST SPECIALIZERS QUALIFIERS* [Function]
Render @deffnx {Method} NAME LAMBDA-LIST on *standard-output*.
NAME, LAMBDA-LIST, SPECIALIZERS and QUALIFIERS are escaped for Texinfo prior to rendering.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@defsetfx` *NAME LAMBDA-LIST* [Function]
Render @deffnx {Setf Expander} NAME LAMBDA-LIST on *standard-output*

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@defunx` *NAME LAMBDA-LIST* [Function]
Render @deffnx Function NAME LAMBDA-LIST on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@itemize-list` *LIST* **&key** *RENDERER KIND FORMAT KEY* [Function]
Render a LIST of items within an @itemize KIND environment.
If RENDERER is non-nil, it must be a function of one argument (every LIST element) that performs the rendering on *standard-output* directly. Otherwise, the rendering is done by calling format, as explained below.

- FORMAT is the format string to use for every LIST element.
- KEY is a function of one argument (every LIST element) used to provide
the necessary arguments to the FORMAT string. If multiple arguments are needed, they should be returned by KEY as multiple values.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`@ref` *ANCHOR LABEL* [Function]
Render ANCHOR as an @ref with online and printed LABEL.
Both ANCHOR and LABEL are escaped for Texinfo prior to rendering.
LABEL is rendered in teletype.
Rendering is done on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`accessor-definition-access-expander` *INSTANCE* [Function]
`(setf accessor-definition-access-expander)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`accessor-definition-foreignp` *INSTANCE* [Function]
`(setf accessor-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`accessor-definition-function` *INSTANCE* [Function]
`(setf accessor-definition-function)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`accessor-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`accessor-definition-symbol` *INSTANCE* [Function]
`(setf accessor-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`accessor-definition-update-expander` *INSTANCE* [Function]
`(setf accessor-definition-update-expander)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`accessor-definition-writer` *INSTANCE* [Function]
`(setf accessor-definition-writer)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`accessor-method-definition-foreignp` *INSTANCE* [Function]
`(setf accessor-method-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`accessor-method-definition-method` *INSTANCE* [Function]
`(setf accessor-method-definition-method)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`accessor-method-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`accessor-method-definition-symbol` *INSTANCE* [Function]
`(setf accessor-method-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`accessor-method-definition-writer` *INSTANCE* [Function]
`(setf accessor-method-definition-writer)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`add-categories-node` *PARENT CONTEXT STATUS DEFINITIONS* [Function]
Add the STATUS DEFINITIONS categories nodes to PARENT in CONTEXT.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 24, (file)

`add-category-node` *PARENT CONTEXT STATUS CATEGORY DEFINITIONS* [Function]
Add the STATUS CATEGORY node to PARENT for DEFINITIONS in CONTEXT.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 24, (file)

`add-child` *PARENT CHILD* **&aux** *PREVIOUS* [Function]
Add CHILD node to PARENT node and return CHILD.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`add-definition` *SYMBOL CATEGORY DEFINITION POOL* [Function]
Add CATEGORY kind of DEFINITION for SYMBOL to POOL.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`add-definitions` *CONTEXT* [Function]
Add all definitions to CONTEXT.

Package [`net.didierverna.declt`], page 29,

Source [`declt.lisp`], page 27, (file)

`add-definitions-node` *PARENT CONTEXT* **&aux** *EXTERNAL-DEFINITIONS EXTERNAL-DEFINITIONS-NUMBER INTERNAL-DEFINITIONS INTERNAL-DEFINITIONS-NUMBER* [Function]
Add the definitions node to PARENT in CONTEXT.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 24, (file)

`add-external-definitions` *CONTEXT* [Function]
Add all external definitions to CONTEXT.

Package [`net.didierverna.declt`], page 29,

Source [`declt.lisp`], page 27, (file)

`add-files-node` *PARENT CONTEXT* **&aux** *SYSTEMS LISP-FILES OTHER-FILES FILES-NODE LISP-FILES-NODE* [Function]
Add the files node to PARENT in CONTEXT.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 26, (file)

`add-internal-definitions` *CONTEXT* [Function]
Add all internal definitions to CONTEXT.

Package [`net.didierverna.declt`], page 29,

Source [`declt.lisp`], page 27, (file)

`add-modules-node` *PARENT CONTEXT* **&aux** *MODULES* [Function]
Add the modules node to PARENT in CONTEXT.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 26, (file)

`add-packages` *CONTEXT* [Function]
Add all package definitions to CONTEXT.

Package [`net.didierverna.declt`], page 29,

Source [`declt.lisp`], page 27, (file)

`add-packages-node` *PARENT CONTEXT* **&aux** *PACKAGES* [Function]
Add the packages node to PARENT in CONTEXT.

Package [`net.didierverna.declt`], page 29,

Source [`package.lisp`], page 26, (file)

`add-status-definitions-node` *PARENT CONTEXT STATUS DEFINITIONS* [Function]
Add the STATUS DEFINITIONS node to PARENT in CONTEXT.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 24, (file)

`add-symbol-definition` *SYMBOL CATEGORY POOL* [Function]
Add and return the CATEGORY kind of definition for SYMBOL to pool, if any.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`add-symbol-definitions` *SYMBOL POOL* [Function]
Add all categorized definitions for SYMBOL to POOL.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`add-systems-node` *PARENT CONTEXT* **&aux** *SYSTEMS-NODE* [Function]
Add the systems node to PARENT in CONTEXT.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 26, (file)

`anchor` *ITEM* [Function]
Render ITEM's anchor.

Package [`net.didierverna.declt`], page 29,

Source [`doc.lisp`], page 24, (file)

`anchor-and-index` *ITEM* [Function]
Anchor and index ITEM.

Package [`net.didierverna.declt`], page 29,

Source [`doc.lisp`], page 24, (file)

`boolean-to-feature-expression` *VALUE* [Function]
Convert boolean VALUE to a form suitable for feature testing.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`class-definition-children` *INSTANCE* [Function]
`(setf class-definition-children)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`class-definition-foreignp` *INSTANCE* [Function]
`(setf class-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`class-definition-methods` *INSTANCE* [Function]
`(setf class-definition-methods)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`class-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`class-definition-parents` *INSTANCE* [Function]
`(setf class-definition-parents)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`class-definition-slots` *INSTANCE* [Function]
`(setf class-definition-slots)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`class-definition-symbol` *INSTANCE* [Function]
`(setf class-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`classoid-definition-children` *INSTANCE* [Function]
`(setf classoid-definition-children)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`classoid-definition-foreignp` *INSTANCE* [Function]
`(setf classoid-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`classoid-definition-methods` *INSTANCE* [Function]
`(setf classoid-definition-methods)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`classoid-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`classoid-definition-parents` *INSTANCE* [Function]
`(setf classoid-definition-parents)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`classoid-definition-slots` *INSTANCE* [Function]
`(setf classoid-definition-slots)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`classoid-definition-symbol` *INSTANCE* [Function]
`(setf classoid-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`clindent` *SYMBOL INDENT* [Function]

Set SYMBOL's indentation to INDENT in (X)Emacs.
This function sets SYMBOL's common-lisp-indent-function property.
If INDENT is a symbol, use its indentation definition.
Otherwise, INDENT is considered as an indentation definition.

Package [`net.didierverna.declt`], page 29,

Source [`meta.lisp`], page 9, (file)

`combination-definition-combination` *INSTANCE* [Function]
`(setf combination-definition-combination)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`combination-definition-foreignp` *INSTANCE* [Function]
`(setf combination-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`combination-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`combination-definition-symbol` *INSTANCE* [Function]
`(setf combination-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`combination-definition-users` *INSTANCE* [Function]
`(setf combination-definition-users)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`compiler-macro-definition-foreignp` *INSTANCE* [Function]
`(setf compiler-macro-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`compiler-macro-definition-function` *INSTANCE* [Function]
`(setf compiler-macro-definition-function)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`compiler-macro-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`compiler-macro-definition-symbol` *INSTANCE* [Function]
`(setf compiler-macro-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`components` *MODULE TYPE* [Function]

Return the list of all components of (sub)TYPE from ASDF MODULE.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 10, (file)

`condition-definition-children` *INSTANCE* [Function]
`(setf condition-definition-children)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`condition-definition-foreignp` *INSTANCE* [Function]
`(setf condition-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`condition-definition-methods` *INSTANCE* [Function]
`(setf condition-definition-methods)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`condition-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`condition-definition-parents` *INSTANCE* [Function]
`(setf condition-definition-parents)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`condition-definition-slots` *INSTANCE* [Function]
`(setf condition-definition-slots)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`condition-definition-symbol` *INSTANCE* [Function]
`(setf condition-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`constant-definition-foreignp` *INSTANCE* [Function]
`(setf constant-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`constant-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`constant-definition-symbol` *INSTANCE* [Function]
`(setf constant-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`context-directory` *CONTEXT* [Function]
Return CONTEXT's main system directory.

Package [`net.didierverna.declt`], page 29,

Source [`doc.lisp`], page 24, (file)

`context-external-definitions` *INSTANCE* [Function]
`(setf context-external-definitions)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`doc.lisp`], page 24, (file)

`context-hyperlinksp` *INSTANCE* [Function]
`(setf context-hyperlinksp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`doc.lisp`], page 24, (file)

`context-internal-definitions` *INSTANCE* [Function]
`(setf context-internal-definitions)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`doc.lisp`], page 24, (file)

`context-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`doc.lisp`], page 24, (file)

`context-packages` *INSTANCE* [Function]
`(setf context-packages)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`doc.lisp`], page 24, (file)

`context-systems` *INSTANCE* [Function]
`(setf context-systems)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`doc.lisp`], page 24, (file)

`copy-accessor-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-accessor-method-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-class-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-classoid-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-combination-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-compiler-macro-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-condition-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-constant-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-context` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`doc.lisp`], page 24, (file)

`copy-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-funcoid-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-function-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-generic-accessor-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-generic-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-generic-writer-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-long-combination-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-macro-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-method-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-node` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`copy-setf-expander-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-short-combination-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-slot-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-special-definition` *INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`copy-structure-definition` *INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`copy-symbol-macro-definition` *INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`copy-type-definition` *INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`copy-writer-definition` *INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`copy-writer-method-definition` *INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`current-time-string` () [Function]

Return the current time as a string.

Package [net.didierverna.declt], page 29,

Source [misc.lisp], page 10, (file)

`definition-foreignp` *INSTANCE* [Function]
`(setf definition-foreignp)` *VALUE INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`definition-p` *OBJECT* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`definition-package` *DEFINITION* [Function]

Return DEFINITION's symbol home package.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`definition-package-name` *DEFINITION* [Function]

Return DEFINITION's symbol home package name.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`definition-source` *OBJECT* [Function]

Return OBJECT's definition source.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`definition-source-by-name` *DEFINITION TYPE* **&key** *NAME* [Function]
Return DEFINITION's source for TYPE.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`definition-symbol` *INSTANCE* [Function]
`(setf definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`definitions-pool-size` *POOL* [Function]
Return the number of elements in definitions POOL.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`defsystem-dependencies` *SYSTEM* [Function]
Return ASDF SYSTEM's defsystem dependencies.

Package [`net.didierverna.declt`], page 29,

Source [`misc.lisp`], page 10, (file)

`escape` *OBJECT* **&optional** *OTHER-CHARS* [Function]
When OBJECT, escape its name for Texinfo.
The escaped characters are @, {, } and optionally a list of OTHER-CHARS.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`escape-anchor` *OBJECT* **&optional** *OTHER-CHARS* [Function]
When OBJECT, escape its name for use as a Texinfo anchor.
The escaped characters are @, {, } and optionally a list of OTHER-CHARS. Additionally, periods, commas and colons are wrapped inside special `<>` constructs.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`file-definitions` *FILE DEFINITIONS* [Function]
Return the subset of DEFINITIONS that belong to FILE.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 23, (file)

`file-node` *FILE CONTEXT* [Function]
Create and return a FILE node in CONTEXT.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 26, (file)

`file-packages` *FILE* [Function]
Return the list of all packages defined in FILE.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 23, (file)

`finalize-definitions` *POOL1 POOL2* [Function]
Finalize the definitions in POOL1 and POOL2.
Currently, this means resolving:
- classes subclasses,
- classes superclasses,
- classes direct methods,
- slots readers,
- slots writers,
- generic functions method combinations,
- method combinations operators (for short ones) and users (for both),
- heterogeneous accessors,
- (generic) functions and macros (short form) setf expanders definitions.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`find-method-definition` *METHOD POOL* [Function]
Find a method definition for METHOD in POOL.
Return NIL if not found.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`first-word-length` *STRING* [Function]
Return the length of the first word in STRING.
Initial whitespace characters are skipped.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`funcoid-definition-foreignp` *INSTANCE* [Function]
`(setf funcoid-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`funcoid-definition-function` *INSTANCE* [Function]
`(setf funcoid-definition-function)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`funcoid-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`funcoid-definition-symbol` *INSTANCE* [Function]
`(setf funcoid-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`function-definition-foreignp` *INSTANCE* [Function]
`(setf function-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`function-definition-function` *INSTANCE* [Function]
`(setf function-definition-function)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`function-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`function-definition-symbol` *INSTANCE* [Function]
`(setf function-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`function-definition-update-expander` *INSTANCE* [Function]
`(setf function-definition-update-expander)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generate-quickutils` () [Function]
Generate the offline quickutil file.

Package [`net.didierverna.declt`], page 29,

Source [`meta.lisp`], page 9, (file)

`generic-accessor-definition-access-expander` *INSTANCE* [Function]
`(setf generic-accessor-definition-access-expander)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-accessor-definition-combination` *INSTANCE* [Function]
`(setf generic-accessor-definition-combination)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-accessor-definition-foreignp` *INSTANCE* [Function]
`(setf generic-accessor-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-accessor-definition-function` *INSTANCE* [Function]
`(setf generic-accessor-definition-function)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-accessor-definition-methods` *INSTANCE* [Function]
`(setf generic-accessor-definition-methods)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-accessor-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-accessor-definition-symbol` *INSTANCE* [Function]
`(setf generic-accessor-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-accessor-definition-update-expander` *INSTANCE* [Function]
`(setf generic-accessor-definition-update-expander)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-accessor-definition-writer` *INSTANCE* [Function]
`(setf generic-accessor-definition-writer)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-definition-combination` *INSTANCE* [Function]
`(setf generic-definition-combination)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-definition-foreignp` *INSTANCE* [Function]
`(setf generic-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-definition-function` *INSTANCE* [Function]
`(setf generic-definition-function)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-definition-methods` *INSTANCE* [Function]
`(setf generic-definition-methods)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-definition-symbol` *INSTANCE* [Function]
`(setf generic-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-definition-update-expander` *INSTANCE* [Function]
`(setf generic-definition-update-expander)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-writer-definition-combination` *INSTANCE* [Function]
`(setf generic-writer-definition-combination)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-writer-definition-foreignp` *INSTANCE* [Function]
`(setf generic-writer-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-writer-definition-function` *INSTANCE* [Function]
`(setf generic-writer-definition-function)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-writer-definition-methods` *INSTANCE* [Function]
`(setf generic-writer-definition-methods)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-writer-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-writer-definition-reader` *INSTANCE* [Function]
`(setf generic-writer-definition-reader)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-writer-definition-symbol` *INSTANCE* [Function]
`(setf generic-writer-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`generic-writer-definition-update-expander` *INSTANCE* [Function]
`(setf generic-writer-definition-update-expander)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`i-reader` *STREAM SUBCHAR ARG* [Function]
Read an argument list for the DEFINDENT macro.

Package [`net.didierverna.declt`], page 29,

Source [`meta.lisp`], page 9, (file)

`lisp-components` *MODULE* [Function]
Return the list of all Lisp source file components from ASDF MODULE.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 10, (file)

`lisp-pathnames` *SYSTEM* **&aux** *FILE LISP-PATHNAMES* [Function]
Return the list of all ASDF SYSTEM's Lisp source file pathnames.
The list includes the system definition file.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 23, (file)

`long-combination-definition-combination` *INSTANCE* [Function]
`(setf long-combination-definition-combination)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`long-combination-definition-foreignp` *INSTANCE* [Function]
`(setf long-combination-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`long-combination-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`long-combination-definition-symbol` *INSTANCE* [Function]
`(setf long-combination-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`long-combination-definition-users` *INSTANCE* [Function]
`(setf long-combination-definition-users)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`macro-definition-access-expander` *INSTANCE* [Function]
`(setf macro-definition-access-expander)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`macro-definition-foreignp` *INSTANCE* [Function]
`(setf macro-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`macro-definition-function` *INSTANCE* [Function]
`(setf macro-definition-function)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`macro-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`macro-definition-symbol` *INSTANCE* [Function]
`(setf macro-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`macro-definition-update-expander` *INSTANCE* [Function]
`(setf macro-definition-update-expander)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`make-accessor-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*FUNCTION* **FUNCTION**) (*UPDATE-EXPANDER* **UPDATE-EXPANDER**) (*WRITER* **WRITER**) (*ACCESS-EXPANDER* **ACCESS-EXPANDER**) [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`make-accessor-method-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*METHOD* **METHOD**) (*WRITER* **WRITER**) [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`make-class-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*PARENTS* **PARENTS**) (*CHILDREN* **CHILDREN**) (*METHODS* **METHODS**) (*SLOTS* **SLOTS**) [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`make-classoid-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*PARENTS* **PARENTS**) (*CHILDREN* **CHILDREN**) (*METHODS* **METHODS**) (*SLOTS* **SLOTS**) [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`make-combination-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*COMBINATION* **COMBINATION**) (*USERS* **USERS**) [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`make-compiler-macro-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*FUNCTION* **FUNCTION**) [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`make-condition-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*PARENTS* **PARENTS**) (*CHILDREN* **CHILDREN**) (*METHODS* **METHODS**) (*SLOTS* **SLOTS**) [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`make-constant-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`make-context` **&key** (*SYSTEMS* **SYSTEMS**) (*PACKAGES* **PACKAGES**) (*EXTERNAL-DEFINITIONS* **EXTERNAL-DEFINITIONS**) (*INTERNAL-DEFINITIONS* **INTERNAL-DEFINITIONS**) (*HYPERLINKSP* **HYPERLINKSP**) [Function]

Package [`net.didierverna.declt`], page 29,

Source [`doc.lisp`], page 24, (file)

`make-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`make-definitions-pool` () [Function]

Create and return a new definitions pool.
A definitions pool is a hash table of categorized definitions.
Keys must be of the form (NAME :CATEGORY).
- NAME is the symbol naming the definition,
- :CATEGORY is one listed in *CATEGORIES*.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`make-funcoid-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*FUNCTION* **FUNCTION**) [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`make-function-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*FUNCTION* **FUNCTION**) (*UPDATE-EXPANDER* **UPDATE-EXPANDER**) [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`make-generic-accessor-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*FUNCTION* **FUNCTION**) (*UPDATE-EXPANDER* **UPDATE-EXPANDER**) (*COMBINATION* **COMBINATION**) (*METHODS* **METHODS**) (*WRITER* **WRITER**) (*ACCESS-EXPANDER* **ACCESS-EXPANDER**) [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`make-generic-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*FUNCTION* **FUNCTION**) (*UPDATE-EXPANDER* **UPDATE-EXPANDER**) (*COMBINATION* **COMBINATION**) (*METHODS* **METHODS**) [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`make-generic-writer-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*FUNCTION* **FUNCTION**) (*UPDATE-EXPANDER* **UPDATE-EXPANDER**) (*COMBINATION* **COMBINATION**) (*METHODS* **METHODS**) (*READER* **READER**) [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`make-long-combination-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*COMBINATION* **COMBINATION**) (*USERS* **USERS**) [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`make-macro-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*FUNCTION* **FUNCTION**) (*ACCESS-EXPANDER* **ACCESS-EXPANDER**) (*UPDATE-EXPANDER* **UPDATE-EXPANDER**) [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`make-method-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*METHOD* **METHOD**) [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`make-node` **&key** (*NAME* **NAME**) (*SYNOPSIS* **SYNOPSIS**) (*SECTION-TYPE* **SECTION-TYPE**) (*SECTION-NAME* **SECTION-NAME**) (*NEXT* **NEXT**) (*PREVIOUS* **PREVIOUS**) (*UP* **UP**) (*CHILDREN* **CHILDREN**) (*BEFORE-MENU-CONTENTS* **BEFORE-MENU-CONTENTS**) (*AFTER-MENU-CONTENTS* **AFTER-MENU-CONTENTS**) [Function]

Package [net.didierverna.declt], page 29,

Source [texi.lisp], page 10, (file)

`make-setf-expander-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*ACCESS* **ACCESS**) (*UPDATE* **UPDATE**) [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`make-short-combination-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*COMBINATION* **COMBINATION**) (*USERS* **USERS**) (*OPERATOR* **OPERATOR**) [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`make-slot-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*SLOT* **SLOT**) (*READERS* **READERS**) (*WRITERS* **WRITERS**) [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`make-slot-definitions` *CLASS* [Function]

Return a list of direct slot definitions for CLASS.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`make-special-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`make-structure-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*PARENTS* **PARENTS**) (*CHILDREN* **CHILDREN**) (*METHODS* **METHODS**) (*SLOTS* **SLOTS**) [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`make-symbol-macro-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`make-type-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`make-writer-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*FUNCTION* **FUNCTION**) (*UPDATE-EXPANDER* **UPDATE-EXPANDER**) (*READER* **READER**) [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`make-writer-method-definition` **&key** (*SYMBOL* **SYMBOL**) (*FOREIGNP* **FOREIGNP**) (*METHOD* **METHOD**) [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`mapcan-definitions-pool` *FUNCTION POOL* [Function]

Like MAPCAN, but work on a definitions POOL.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`method-definition-foreignp` *INSTANCE* [Function]
`(setf method-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`method-definition-method` *INSTANCE* [Function]
`(setf method-definition-method)` *VALUE INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`method-definition-p` *OBJECT* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`method-definition-symbol` *INSTANCE* [Function]
`(setf method-definition-symbol)` *VALUE INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`method-name` *METHOD* **&aux** *NAME* [Function]

Return METHOD's name.
Return a second value of T if METHOD is a writer method.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`module-components` *MODULE* [Function]

Return the list of all module components from ASDF MODULE.

Package [net.didierverna.declt], page 29,

Source [asdf.lisp], page 10, (file)

`module-node` *MODULE CONTEXT* [Function]

Create and return a MODULE node in CONTEXT.

Package [net.didierverna.declt], page 29,

Source [asdf.lisp], page 26, (file)

`node-after-menu-contents` *INSTANCE* [Function]
`(setf node-after-menu-contents)` *VALUE INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [texi.lisp], page 10, (file)

`node-before-menu-contents` *INSTANCE* [Function]
`(setf node-before-menu-contents)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`node-children` *INSTANCE* [Function]
`(setf node-children)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`node-name` *INSTANCE* [Function]
`(setf node-name)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`node-next` *INSTANCE* [Function]
`(setf node-next)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`node-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`node-previous` *INSTANCE* [Function]
`(setf node-previous)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`node-section-name` *INSTANCE* [Function]
`(setf node-section-name)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`node-section-type` *INSTANCE* [Function]
`(setf node-section-type)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`node-synopsis` *INSTANCE* [Function]
`(setf node-synopsis)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`node-up` *INSTANCE* [Function]
`(setf node-up)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`package-definitions` *PACKAGE DEFINITIONS* [Function]
Return the subset of DEFINITIONS that belong to PACKAGE.

Package [`net.didierverna.declt`], page 29,

Source [`package.lisp`], page 23, (file)

`package-external-symbols` *PACKAGE* **&aux** *EXTERNAL-SYMBOLS* [Function]
Return the list of external symbols from PACKAGE.

Package [`net.didierverna.declt`], page 29,

Source [`package.lisp`], page 23, (file)

`package-internal-symbols` *PACKAGE* **&aux** *EXTERNAL-SYMBOLS* [Function]
INTERNAL-SYMBOLS
Return the list of internal definitions from PACKAGE.

Package [`net.didierverna.declt`], page 29,

Source [`package.lisp`], page 23, (file)

`parse-contact(s)` *contact*(*s*) [Function]
Parse CONTACT(S) as either a contact string, or a list of such.
Return a list of name(s) an email(s) as two values.
See 'PARSE-CONTACT-STRING' for more information.

Package [`net.didierverna.declt`], page 29,

Source [`misc.lisp`], page 10, (file)

`parse-contact-string` *STRING* **&aux** *POS-<* *POS->* [Function]
Parse STRING as "NAME <EMAIL>".
Return NAME and EMAIL as two values.

Package [`net.didierverna.declt`], page 29,

Source [`misc.lisp`], page 10, (file)

`pool-combination-users` *POOL COMBINATION* [Function]
Return a list of all generic definitions in POOL using method COMBINATION.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`qualifiers` *METHOD* [Function]
Return METHOD's qualifiers.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`read-next-line` *STREAM* [Function]
Read one line from STREAM.
Return a list of two values:
- the line itself, or STREAM,
- whether a newline character is missing at the end of the line.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`reference-component` *COMPONENT* [Function]
Render COMPONENT's reference.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 26, (file)

`relative-location` *COMPONENT RELATIVE-TO* [Function]
Return COMPONENT's location RELATIVE-TO.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 10, (file)

`release-status-number` *RELEASE-STATUS* [Function]

Package [`net.didierverna.declt.setup`], page 46,

Source [`setup.lisp`], page 28, (file)

`render-definition-core` *DEFINITION CONTEXT* [Function]
Render DEFINITION's documentation core in CONTEXT.
The documentation core includes all common definition attributes:
- package,
- source location.

Each element is rendered as a table item.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 24, (file)

`render-dependencies` *DEPENDENCIES COMPONENT RELATIVE-TO* **&optional** *PREFIX* **&aux** *LENGTH* [Function]
Render COMPONENT's DEPENDENCIES RELATIVE-TO.
Optionally PREFIX the title.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 26, (file)

`render-docstring` *ITEM* [Function]
Render ITEM's documentation string.
Rendering is done on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`doc.lisp`], page 24, (file)

`render-external-definitions-references` *DEFINITIONS* [Function]
Render references to a list of external DEFINITIONS.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 24, (file)

`render-header` *LIBRARY TAGLINE VERSION CONTACT-NAMES CONTACT-EMAILS COPYRIGHT LICENSE TEXI-NAME INFO-FILE DECLT-NOTICE CURRENT-TIME-STRING* [Function]
Render the header of the Texinfo file.

Package [`net.didierverna.declt`], page 29,

Source [`declt.lisp`], page 27, (file)

`render-headline` *DEFINITION* [Function]
Render a headline for DEFINITION. Also anchor and index it.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 24, (file)

`render-initargs` *CLASSOID* [Function]
Render CLASSOID's direct default initargs.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 24, (file)

`render-internal-definitions-references` *DEFINITIONS* [Function]
Render references to a list of internal DEFINITIONS.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 24, (file)

`render-lambda-list` *LAMBDA-LIST* **&optional** *SPECIALIZERS* **&aux** *FIRSTP AFTER-REQUIRED-ARGS-P* [Function]
Render LAMBDA-LIST with potential SPECIALIZERS.
LAMBDA-LIST and SPECIALIZERS are escaped for Texinfo prior to rendering. Rendering is done on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`render-location` *PATHNAME CONTEXT* **&optional** *TITLE* **&aux** *PROBED-PATHNAME RELATIVE-TO HYPERLINKP* [Function]
Render an itemized location line for PATHNAME in CONTEXT.
Rendering is done on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`doc.lisp`], page 24, (file)

`render-method-combination` *GENERIC* **&aux** *COMBINATION* [Function]
Render GENERIC definition's method combination documentation.
The standard method combination is not rendered.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 24, (file)

`render-node` *NODE LEVEL* **&aux** *NODE-NAME SAFE-NODE-NAME* [Function]
Render NODE at LEVEL and all its children at LEVEL+1.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`render-packages-references` *PACKAGES* [Function]
Render a list of PACKAGES references.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 26, (file)

`render-references` *LIST TITLE* **&aux** *LENGTH* [Function]
Render references to a LIST of items.
References are rendered in a table item named TITLE as a list, unless there is only one item in LIST.

Rendering is done on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`doc.lisp`], page 24, (file)

`render-slot` *SLOT* [Function]
Render SLOT's documentation.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 24, (file)

`render-slot-property` *SLOT PROPERTY* **&key** *RENDERER* **&aux** *VALUE* [Function]
Render SLOT definition's PROPERTY value as a table item.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 24, (file)

`render-slots` *CLASSOID* [Function]
Render CLASSOID's direct slots documentation.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 24, (file)

`render-source` *ITEM CONTEXT* [Function]
Render an itemized source line for ITEM in CONTEXT.
Rendering is done on *standard-output*.

Package [`net.didierverna.declt`], page 29,

Source [`doc.lisp`], page 24, (file)

`render-text` *TEXT* [Function]
Render TEXT for Texinfo.
Rendering is done on *standard-output*.
TEXT is assumed to be plain 80 columns.
The rendering takes care of escaping the text for Texinfo, and attempts to embellish the output by detecting potential paragraphs from standalone lines.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`render-top-node` *NODE* [Function]
Render the whole nodes hierarchy starting at toplevel NODE.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

`render-use-list` *LIST TITLE CONTEXT* [Function]
Render a package use/used-by LIST with TITLE in CONTEXT.

Package [`net.didierverna.declt`], page 29,

Source [`package.lisp`], page 26, (file)

`sbcl-has-setf-inverse-meta-info ()` [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`setf-expander-definition-access` *INSTANCE* [Function]
`(setf setf-expander-definition-access)` *VALUE INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`setf-expander-definition-foreignp` *INSTANCE* [Function]
`(setf setf-expander-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`setf-expander-definition-p` *OBJECT* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`setf-expander-definition-symbol` *INSTANCE* [Function]
`(setf setf-expander-definition-symbol)` *VALUE INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`setf-expander-definition-update` *INSTANCE* [Function]
`(setf setf-expander-definition-update)` *VALUE INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`setf-expander-p` *SYMBOL* [Function]

Return whether SYMBOL defines a setf-expander.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`short-combination-definition-combination` *INSTANCE* [Function]
`(setf short-combination-definition-combination)` *VALUE* [Function]
INSTANCE

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`short-combination-definition-foreignp` *INSTANCE* [Function]
`(setf short-combination-definition-foreignp)` *VALUE* [Function]
INSTANCE

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`short-combination-definition-operator` *INSTANCE* [Function]
`(setf short-combination-definition-operator)` *VALUE* [Function]
INSTANCE

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`short-combination-definition-p` *OBJECT* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`short-combination-definition-symbol` *INSTANCE* [Function]
`(setf short-combination-definition-symbol)` *VALUE INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`short-combination-definition-users` *INSTANCE* [Function]
`(setf short-combination-definition-users)` *VALUE INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`slot-definition-foreignp` *INSTANCE* [Function]
`(setf slot-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`slot-definition-p` *OBJECT* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`slot-definition-readers` *INSTANCE* [Function]
`(setf slot-definition-readers)` *VALUE INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`slot-definition-slot` *INSTANCE* [Function]
`(setf slot-definition-slot)` *VALUE INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`slot-definition-symbol` *INSTANCE* [Function]
`(setf slot-definition-symbol)` *VALUE INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`slot-definition-writers` *INSTANCE* [Function]
`(setf slot-definition-writers)` *VALUE INSTANCE* [Function]

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`slot-property` *SLOT PROPERTY* [Function]
Return SLOT definition's PROPERTY value.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

`special-definition-foreignp` *INSTANCE* [Function]
`(setf special-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`special-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`special-definition-symbol` *INSTANCE* [Function]
`(setf special-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`specializers` *METHOD* [Function]
Return METHOD's specializers.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`structure-definition-children` *INSTANCE* [Function]
`(setf structure-definition-children)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`structure-definition-foreignp` *INSTANCE* [Function]
`(setf structure-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`structure-definition-methods` *INSTANCE* [Function]
`(setf structure-definition-methods)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`structure-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`structure-definition-parents` *INSTANCE* [Function]
`(setf structure-definition-parents)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`structure-definition-slots` *INSTANCE* [Function]
`(setf structure-definition-slots)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`structure-definition-symbol` *INSTANCE* [Function]
`(setf structure-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`sub-component-p` *COMPONENT RELATIVE-TO* **&aux** *COMPONENT-PATHNAME* [Function]
Return T if COMPONENT can be found under RELATIVE-TO.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 10, (file)

`subsystems` *SYSTEM RELATIVE-TO* [Function]
Return the list of SYSTEM subsystems RELATIVE-TO.
This function recursively descends all found subsystems.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 10, (file)

`symbol-macro-definition-foreignp` *INSTANCE* [Function]
`(setf symbol-macro-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`symbol-macro-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`symbol-macro-definition-symbol` *INSTANCE* [Function]
`(setf symbol-macro-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`system-base-name` *SYSTEM* **&aux** *FILE* [Function]
Return the basename part of ASDF SYSTEM's definition file.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 10, (file)

`system-dependencies` *SYSTEM* [Function]
Return all SYSTEM dependencies.
This includes both :defsystem-depends-on and :depends-on.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 10, (file)

`system-directory` *SYSTEM* [Function]
Return ASDF SYSTEM's directory.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 10, (file)

`system-external-symbols` *SYSTEM* [Function]
Return the list of ASDF SYSTEM's external symbols.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 23, (file)

`system-file-name` *SYSTEM* **&aux** *FILE* [Function]
Return the name part of ASDF SYSTEM's definition file.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 10, (file)

`system-file-type` *SYSTEM* **&aux** *FILE* [Function]
Return the type part of ASDF SYSTEM's definition file.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 10, (file)

`system-internal-symbols` *SYSTEM* [Function]
Return the list of ASDF SYSTEM's internal symbols.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 23, (file)

`system-node` *SYSTEM CONTEXT* [Function]
Create and return a SYSTEM node in CONTEXT.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 26, (file)

`system-packages` *SYSTEM* [Function]
Return the list of packages defined in ASDF SYSTEM.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 23, (file)

`tilde-reader` *STREAM CHAR* [Function]
Read a series of `~"string"` to be concatenated together.

Package [`net.didierverna.declt`], page 29,

Source [`meta.lisp`], page 9, (file)

`type-definition-foreignp` *INSTANCE* [Function]
`(setf type-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`type-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`type-definition-symbol` *INSTANCE* [Function]
`(setf type-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`writer-definition-foreignp` *INSTANCE* [Function]
`(setf writer-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`writer-definition-function` *INSTANCE* [Function]
`(setf writer-definition-function)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`writer-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`writer-definition-reader` *INSTANCE* [Function]
`(setf writer-definition-reader)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`writer-definition-symbol` *INSTANCE* [Function]
`(setf writer-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`writer-definition-update-expander` *INSTANCE* [Function]
`(setf writer-definition-update-expander)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`writer-method-definition-foreignp` *INSTANCE* [Function]
`(setf writer-method-definition-foreignp)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`writer-method-definition-method` *INSTANCE* [Function]
`(setf writer-method-definition-method)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`writer-method-definition-p` *OBJECT* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

`writer-method-definition-symbol` *INSTANCE* [Function]
`(setf writer-method-definition-symbol)` *VALUE INSTANCE* [Function]

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

6.2.4 Generic functions

`anchor-name` *ITEM* [Generic Function]
Return ITEM's anchor name.

Package [`net.didierverna.declt`], page 29,

Source [`doc.lisp`], page 24, (file)

Methods

`anchor-name` (*COMPONENT* `component`) [Method]
Return COMPONENT's anchor name.

Source [`asdf.lisp`], page 26, (file)

`anchor-name` (*PACKAGE* `package`) [Method]
Return PACKAGE's anchor name.

Source [`package.lisp`], page 26, (file)

`anchor-name` (*METHOD* `method-definition`) [Method]
Return METHOD's qualified symbol name, specializers and qualifiers .

Source [`symbol.lisp`], page 24, (file)

`anchor-name` *DEFINITION* [Method]
Return DEFINITION's qualified symbol name.
This is the default method for most definitions.

Source [`symbol.lisp`], page 24, (file)

`anchor-name` *ITEM* `around` [Method]
Surround ITEM's anchor name with "go to the [...] <type>".

`category-definitions` *CATEGORY POOL* [Generic Function]
Return all CATEGORY definitions from POOL.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

Methods

`category-definitions` *CATEGORY POOL* [Method]
Default method used for root CATEGORYs.

`category-definitions` (*CATEGORY* (`eql accessor`)) *POOL* [Method]
Method used for ordinary accessors.

`category-definitions` (*CATEGORY* (`eql writer`)) *POOL* [Method]
Method used for ordinary writers.
Note that this only returns standalone (toplevel) writers.

`category-definitions` (*CATEGORY* (`eql generic-accessor`)) *POOL* [Method]
Method used for generic accessors.

`category-definitions` (*CATEGORY* (`eql generic-writer`)) *POOL* [Method]
Method used for generic writers.
Note that this only returns standalone (toplevel) generic writers.

`category-definitions` (*CATEGORY* (`eql` `short-combination`)) *POOL* [Method]
Method used for short method combinations.

`category-definitions` (*CATEGORY* (`eql` `long-combination`)) *POOL* [Method]
Method used for long method combinations.

`category-definitions` (*CATEGORY* (`eql` `setf-expander`)) *POOL* [Method]
Method used for setf expanders.

`definition-combination-users` *DEFINITION COMBINATION* [Generic Function]
Return a list of definitions using method COMBINATION.
The list may boil down to a generic function definition, but may also contain both a reader and a writer.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

Methods

`definition-combination-users` *DEFINITION COMBINATION* [Method]
Default method, for non generic function definitions.
Return nil.

`definition-combination-users` (*DEFINITION* `generic-definition`) *COMBINATION* [Method]
Method for simple generic and writer definitions.

`definition-combination-users` (*DEFINITION* `generic-accessor-definition`) *COMBINATION* [Method]
Method for generic accessor definitions.

`definition-file-definitions` *DEFINITION FILE* [Generic Function]
Return the list of definitions from DEFINITION that belong to FILE.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 23, (file)

Methods

`definition-file-definitions` *DEFINITION FILE* [Method]
Default method for definitions not containing sub-definitions.

`definition-file-definitions` (*MACRO* `macro-definition`) *FILE* [Method]
Handle MACRO and its setf expander.

`definition-file-definitions` (*ACCESSOR* `accessor-definition`) *FILE* [Method]
Handle ACCESSOR, its writer and its setf expander.

`definition-file-definitions` (*ACCESSOR-METHOD* `accessor-method-definition`) *FILE* [Method]
Handle ACCESSOR-METHOD and its writer method.

`definition-file-definitions` (*GENERIC* `generic-definition`) *FILE* [Method]
Handle GENERIC function and its methods.

`definition-file-definitions` (*GENERIC-ACCESSOR* `generic-accessor-definition`) *FILE* [Method]
Handle GENERIC-ACCESSOR, its generic writer and its setf expander.

`definition-package-definitions` *DEFINITION PACKAGE* [Generic Function]
Return the list of definitions from DEFINITION that belong to PACKAGE.

Package [`net.didierverna.declt`], page 29,

Source [`package.lisp`], page 23, (file)

Methods

`definition-package-definitions` *DEFINITION PACKAGE* [Method]
Default method for definitions not containing sub-definitions.

`definition-package-definitions` (*MACRO* `macro-definition`) *PACKAGE* [Method]
Handle MACRO and its setf expander.

`definition-package-definitions` (*ACCESSOR* `accessor-definition`) *PACKAGE* [Method]
Handle ACCESSOR, its writer and its setf expander.

`definition-package-definitions` (*ACCESSOR-METHOD* `accessor-method-definition`) *PACKAGE* [Method]
Handle ACCESSOR-METHOD and its writer method.

`definition-package-definitions` (*GENERIC* `generic-definition`) *PACKAGE* [Method]
Handle GENERIC function and its methods.

`definition-package-definitions` (*GENERIC-ACCESSOR* `generic-accessor-definition`) *PACKAGE* [Method]
Handle GENERIC-ACCESSOR, its generic writer and its setf expander.

`docstring` *ITEM* [Generic Function]
Return ITEM's docstring (Lisp documentation).

Package [`net.didierverna.declt`], page 29,

Source [`item.lisp`], page 12, (file)

Methods

`docstring` (*PACKAGE* `package`) [Method]
Return PACKAGE's docstring.

Source [`package.lisp`], page 23, (file)

`docstring` (*TYPE* `type-definition`) [Method]
Return TYPE's docstring.

Source [`symbol.lisp`], page 12, (file)

`docstring` (*CLASSOID* `classoid-definition`) [Method]
Return CLASSOID's docstring.

Source [`symbol.lisp`], page 12, (file)

`docstring` (*COMBINATION* `combination-definition`) [Method]
Return method COMBINATION's docstring.

Source [`symbol.lisp`], page 12, (file)

`docstring` (*SLOT* `slot-definition`) [Method]
Return SLOT's docstring.

Source [`symbol.lisp`], page 12, (file)

`docstring` (*EXPANDER* `setf-expander-definition`) [Method]
Return setf EXPANDER's docstring.

Source [`symbol.lisp`], page 12, (file)

`docstring` (*WRITER* `generic-writer-definition`) [Method]
Return generic WRITER's docstring.

Source [`symbol.lisp`], page 12, (file)

`docstring` (*METHOD* `method-definition`) [Method]
Return METHOD's docstring.

Source [`symbol.lisp`], page 12, (file)

`docstring` (*WRITER* `writer-definition`) [Method]
Return WRITER's docstring.

Source [`symbol.lisp`], page 12, (file)

`docstring` (*COMPILER-MACRO* `compiler-macro-definition`) [Method]
Return COMPILER-MACRO's docstring.

Source [`symbol.lisp`], page 12, (file)

`docstring` (*FUNCOID* `funcoid-definition`) [Method]
Return FUNCOID's docstring.

Source [`symbol.lisp`], page 12, (file)

`docstring` (*SYMBOL-MACRO* `symbol-macro-definition`) [Method]
Return NIL because symbol macros don't have a docstring.

Source [`symbol.lisp`], page 12, (file)

`docstring` (*SPECIAL* `special-definition`) [Method]
Return SPECIAL variable's docstring.

Source [`symbol.lisp`], page 12, (file)

`docstring` (*CONSTANT* `constant-definition`) [Method]
Return CONSTANT's docstring.

Source [`symbol.lisp`], page 12, (file)

`document` *ITEM CONTEXT* **&key** *ADDITIONAL-METHODS DOCUMENT-WRITERS GENERIC-SOURCE* **&allow-other-keys** [Generic Function]

Render ITEM's documentation in CONTEXT.

Package [`net.didierverna.declt`], page 29,

Source [`doc.lisp`], page 24, (file)

Methods

`document` (*SYSTEM* `system`) *CONTEXT* **&key** [Method]

Render SYSTEM's documentation in CONTEXT.

Source [`asdf.lisp`], page 26, (file)

`document` (*MODULE* `module`) *CONTEXT* **&key** [Method]

Render MODULE's documentation in CONTEXT.

Source [`asdf.lisp`], page 26, (file)

`document` (*FILE* `cl-source-file`) *CONTEXT* **&key &aux** *PATHNAME* [Method]

Render lisp FILE's documentation in CONTEXT.

Source [`asdf.lisp`], page 26, (file)

`document` (*COMPONENT* `component`) *CONTEXT* **&key &aux** *RELATIVE-TO* [Method]

Render COMPONENT's documentation in CONTEXT.

Source [`asdf.lisp`], page 26, (file)

`document` (*COMPONENT* `component`) *CONTEXT* **&key** `around` [Method]

Anchor and index COMPONENT in CONTEXT. Document it in a @table environment.

Source [`asdf.lisp`], page 26, (file)

`document` (*PACKAGE* `package`) *CONTEXT* **&key** [Method]

Render PACKAGE's documentation in CONTEXT.

Source [`package.lisp`], page 26, (file)

`document` (*TYPE* `type-definition`) *CONTEXT* **&key** [Method]

Render TYPE's documentation in CONTEXT.

Source [`symbol.lisp`], page 24, (file)

`document` (*CLASS* `class-definition`) *CONTEXT* **&key** [Method]

Render CLASS's documentation in CONTEXT.

Source [`symbol.lisp`], page 24, (file)

`document` (*STRUCTURE* `structure-definition`) *CONTEXT* **&key** [Method]

Render STRUCTURE's documentation in CONTEXT.

Source [`symbol.lisp`], page 24, (file)

`document` (*CONDITION* `condition-definition`) *CONTEXT* **&key** [Method]

Render CONDITION's documentation in CONTEXT.

Source [`symbol.lisp`], page 24, (file)

`document` (*COMBINATION* `long-combination-definition`) *CONTEXT* **&key** [Method]

Render long method COMBINATION's documentation in CONTEXT.

Source [`symbol.lisp`], page 24, (file)

`document` (*COMBINATION* `short-combination-definition`) *CONTEXT* **&key** [Method]

Render short method COMBINATION's documentation in CONTEXT.

Source [`symbol.lisp`], page 24, (file)

`document` (*EXPANDER* `setf-expander-definition`) *CONTEXT* **&key** [Method]

Render setf EXPANDER's documentation in CONTEXT.

Source [`symbol.lisp`], page 24, (file)

`document` (*ACCESSOR* `generic-accessor-definition`) *CONTEXT* **&key &aux** *ACCESS-EXPANDER UPDATE-EXPANDER WRITER* [Method]

Render generic ACCESSOR's documentation in CONTEXT.

Source [`symbol.lisp`], page 24, (file)

`document` (*WRITER* `generic-writer-definition`) *CONTEXT* **&key** *ADDITIONAL-METHODS* [Method]

Render generic WRITER's documentation in CONTEXT.

Source [`symbol.lisp`], page 24, (file)

`document` (*GENERIC* `generic-definition`) *CONTEXT* **&key** [Method]

Render GENERIC's documentation in CONTEXT.

Source [`symbol.lisp`], page 24, (file)

`document` (*METHOD* `accessor-method-definition`) *CONTEXT* **&key** *DOCUMENT-WRITERS GENERIC-SOURCE* [Method]

Render accessor METHOD's documentation in CONTEXT.

Source [`symbol.lisp`], page 24, (file)

`document` (*METHOD* `method-definition`) *CONTEXT* **&key** *GENERIC-SOURCE* [Method]

Render METHOD's documentation in CONTEXT.
GENERIC-SOURCE is the source of METHOD's generic function.

Source [`symbol.lisp`], page 24, (file)

`document` (*ACCESSOR* `accessor-definition`) *CONTEXT* **&key &aux** *ACCESS-EXPANDER UPDATE-EXPANDER WRITER MERGE-EXPANDER MERGE-WRITER MERGE-SETTERS* [Method]

Render ACCESSOR's documentation in CONTEXT.

Source [`symbol.lisp`], page 24, (file)

`document` (*WRITER* `writer-definition`) *CONTEXT* **&key** [Method]

Render WRITER's documentation in CONTEXT.

Source [`symbol.lisp`], page 24, (file)

`document` (*FUNCTION* `function-definition`) *CONTEXT* **&key** [Method]

Render FUNCTION's documentation in CONTEXT.

Source [`symbol.lisp`], page 24, (file)

`document` (*COMPILER-MACRO* `compiler-macro-definition`) *CONTEXT* **&key** [Method]

Render COMPILER-MACRO's documentation in CONTEXT.

Source [`symbol.lisp`], page 24, (file)

`document` (*MACRO* `macro-definition`) *CONTEXT* **&key &aux** *ACCESS-EXPANDER UPDATE-EXPANDER MERGE* [Method]

Render MACRO's documentation in CONTEXT.

Source [`symbol.lisp`], page 24, (file)

`document` (*SYMBOL-MACRO* `symbol-macro-definition`) *CONTEXT* **&key** [Method]

Render SYMBOL-MACRO's documentation in CONTEXT.

Source [`symbol.lisp`], page 24, (file)

`document` (*SPECIAL* `special-definition`) *CONTEXT* **&key** [Method]

Render SPECIAL variable's documentation in CONTEXT.

Source [`symbol.lisp`], page 24, (file)

`document` (*CONSTANT* `constant-definition`) *CONTEXT* **&key** [Method]

Render CONSTANT's documentation in CONTEXT.

Source [`symbol.lisp`], page 24, (file)

`find-definition` *NAME CATEGORY POOL* **&optional** *ERRORP* [Generic Function]

Find a CATEGORY definition for NAME in POOL.
If ERRORP, throw an error if not found. Otherwise, just return NIL.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

Methods

`find-definition` *NAME CATEGORY POOL* **&optional** *ERRORP* **&aux** *DEFINITION* [Method]
Default method used for root CATEGORYs

`find-definition` *NAME* (*CATEGORY* `(eql accessor)`) *POOL* **&optional** *ERRORP* **&aux** *DEFINITION* [Method]
Method used to find accessor definitions.

`find-definition` *NAME* (*CATEGORY* `(eql writer)`) *POOL* **&optional** *ERRORP* **&aux** *DEFINITION* [Method]
Method used to find writer definitions.
Name must be that of the reader (not the SETF form).

`find-definition` *NAME* (*CATEGORY* `(eql generic-accessor)`) *POOL* **&optional** *ERRORP* **&aux** *DEFINITION* [Method]
Method used to find generic accessor definitions.

`find-definition` *NAME* (*CATEGORY* `(eql generic-writer)`) *POOL* **&optional** *ERRORP* **&aux** *DEFINITION* [Method]
Method used to find generic writer definitions.
Name must be that of the reader (not the SETF form).

`headline-function` *DEFINITION* [Generic Function]
Return a suitable headline function for DEFINITION.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 24, (file)

Methods

`headline-function` (*FUNCTION* `function-definition`) [Method]
Return #'@DEFUNX.

`headline-function` (*GENERIC* `generic-definition`) [Method]
Return #'@DEFGENERICX.

`headline-function` (*EXPANDER* `setf-expander-definition`) [Method]
Return #'@DEFSETFX.

`index` *ITEM* [Generic Function]
Render ITEM's indexing command.

Package [`net.didierverna.declt`], page 29,

Source [`doc.lisp`], page 24, (file)

Methods

`index` (*SYSTEM* `system`) [Method]
Render SYSTEM's indexing command.

Source [`asdf.lisp`], page 26, (file)

`index` (*MODULE* `module`) [Method]
Render MODULE's indexing command.

Source [`asdf.lisp`], page 26, (file)

`index` (*HTML-FILE* `html-file`) [Method]
Render HTML-FILE's indexing command.

Source [`asdf.lisp`], page 26, (file)

`index` (*DOC-FILE* `doc-file`) [Method]
Render DOC-FILE's indexing command.

Source [`asdf.lisp`], page 26, (file)

`index` (*STATIC-FILE* `static-file`) [Method]
Render STATIC-FILE's indexing command.

Source [`asdf.lisp`], page 26, (file)

`index` (*JAVA-FILE* `java-source-file`) [Method]
Render JAVA-FILE's indexing command.

Source [`asdf.lisp`], page 26, (file)

`index` (*C-FILE* `c-source-file`) [Method]
Render C-FILE's indexing command.

Source [`asdf.lisp`], page 26, (file)

`index` (*LISP-FILE* `cl-source-file`) [Method]
Render LISP-FILE's indexing command.

Source [`asdf.lisp`], page 26, (file)

`index` (*PACKAGE* `package`) [Method]
Render PACKAGE's indexing command.

Source [`package.lisp`], page 26, (file)

`index` (*TYPE* `type-definition`) [Method]
Render TYPE's indexing command.

Source [`symbol.lisp`], page 24, (file)

`index` (*CLASS* `class-definition`) [Method]
Render CLASS's indexing command.

Source [`symbol.lisp`], page 24, (file)

`index` (*STRUCTURE* `structure-definition`) [Method]
Render STRUCTURE's indexing command.

Source [`symbol.lisp`], page 24, (file)

`index` (*CONDITION* `condition-definition`) [Method]
Render CONDITION's indexing command.

Source [`symbol.lisp`], page 24, (file)

`index` (*COMBINATION* `long-combination-definition`) [Method]
Render long method COMBINATION's indexing command.

Source [`symbol.lisp`], page 24, (file)

`index` (*COMBINATION* `short-combination-definition`) [Method]
Render short method COMBINATION's indexing command.

Source [`symbol.lisp`], page 24, (file)

`index` (*SLOT* `slot-definition`) [Method]
Render SLOT's indexing command.

Source [`symbol.lisp`], page 24, (file)

`index` (*EXPANDER* `setf-expander-definition`) [Method]
Render setf EXPANDER's indexing command.

Source [`symbol.lisp`], page 24, (file)

`index` (*GENERIC* `generic-definition`) [Method]
Render GENERIC's indexing command.

Source [`symbol.lisp`], page 24, (file)

`index` (*METHOD* `method-definition`) [Method]
Render METHOD's indexing command.

Source [`symbol.lisp`], page 24, (file)

`index` (*FUNCTION* `function-definition`) [Method]
Render FUNCTION's indexing command.

Source [`symbol.lisp`], page 24, (file)

`index` (*COMPILER-MACRO* `compiler-macro-definition`) [Method]
Render COMPILER-MACRO's indexing command.

Source [`symbol.lisp`], page 24, (file)

`index` (*MACRO* `macro-definition`) [Method]
Render MACRO's indexing command.

Source [`symbol.lisp`], page 24, (file)

`index` (*SYMBOL-MACRO* `symbol-macro-definition`) [Method]
Render SYMBOL-MACRO's indexing command.

Source [`symbol.lisp`], page 24, (file)

`index` (*SPECIAL* `special-definition`) [Method]
Render SPECIAL's indexing command.

Source [`symbol.lisp`], page 24, (file)

`index` (*CONSTANT* `constant-definition`) [Method]
Render CONSTANT's indexing command.

Source [`symbol.lisp`], page 24, (file)

`lambda-list` *OBJECT* [Generic Function]
Return OBJECT's lambda-list.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

Methods

`lambda-list` (*FUNCTION* `function`) [Method]
Return FUNCTION's lambda-list.

`lambda-list` (*FUNCOID* `funcoid-definition`) [Method]
Return FUNCOID's lambda-list.

`lambda-list` (*EXPANDER* `setf-expander-definition`) **&aux** *UPDATE* [Method]
Return setf EXPANDER's lambda-list.

`lambda-list` (*METHOD* `method-definition`) [Method]
Return METHOD's lambda-list.

`lambda-list` (*TYPE* `type-definition`) [Method]
Return TYPE's lambda-list.

`name` *OBJECT* [Generic Function]
Return OBJECT's name as a string.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

Methods

`name` (*SOURCE-FILE* `source-file`) **&aux** *NAME* *EXTENSION* [Method]
Return SOURCE-FILE's name, possibly adding its extension.

Source [`asdf.lisp`], page 23, (file)

`name` (*COMPONENT* `component`) [Method]
Return COMPONENT's name.

Source [`asdf.lisp`], page 23, (file)

`name` (*PACKAGE* `package`) [Method]
Return PACKAGE's name.

Source [`package.lisp`], page 23, (file)

`name` (*EXPANDER* `setf-expander-definition`) [Method]
Return setf EXPANDER's name, that is (setf <name>).

Source [`symbol.lisp`], page 12, (file)

`name` (*GENERIC-WRITER* `generic-writer-definition`) [Method]
Return GENERIC-WRITER's name, that is (setf <name>).

Source [`symbol.lisp`], page 12, (file)

`name` (*WRITER-METHOD* `writer-method-definition`) [Method]
Return WRITER-METHOD's name, that is (setf <name>).

Source [`symbol.lisp`], page 12, (file)

`name` (*WRITER* `writer-definition`) [Method]
Return WRITER's name, that is (setf <name>).

Source [`symbol.lisp`], page 12, (file)

`name` (*DEFINITION* `definition`) [Method]
Return DEFINITION's symbol name.

Source [`symbol.lisp`], page 12, (file)

`name` *OBJECT* [Method]
Princ object to a string.

`name` (*SYMBOL* `symbol`) [Method]
Return SYMBOL's name.

`name` (*CHAR* `character`) [Method]
Return revealed CHAR.

`name` (*STRING* `string`) [Method]
Return STRING.

`name` (*PATHNAME* `pathname`) [Method]
Return PATHNAME's name.

`pretty-specializer` *SPECIALIZER* **&optional** *QUALIFY* [Generic Function]
Return a printable form of SPECIALIZER.
If QUALIFY, also qualify the symbols.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

Methods

`pretty-specializer` *SPECIALIZER* **&optional** *QUALIFY* [Method]
Return either SPECIALIZER itself, or its class name when appropriate.

`pretty-specializer` (*SPECIALIZER* `eql-specializer`) [Method]
&optional *QUALIFY*
Return the (eql object) list corresponding to SPECIALIZER in a string.

`reader-definitions` *SLOT POOL1 POOL2* [Generic Function]
Return a list of reader definitions for SLOT.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

Methods

`reader-definitions` *SLOT POOL1 POOL2* [Method]
Defaut method for class and condition slots.

`reader-definitions` (*SLOT* [Method]
`structure-direct-slot-definition`) *POOL1 POOL2*
Method for structure slots.

`reference` *ITEM* [Generic Function]
Render ITEM's reference.

Package [`net.didierverna.declt`], page 29,

Source [`doc.lisp`], page 24, (file)

Methods

`reference` (*SYSTEM* `system`) [Method]
Render SYSTEM's reference.

Source [`asdf.lisp`], page 26, (file)

`reference` (*MODULE* `module`) [Method]
Render MODULE's reference.

Source [`asdf.lisp`], page 26, (file)

`reference` (*SOURCE-FILE* `source-file`) [Method]
Render SOURCE-FILE's reference.

Source [`asdf.lisp`], page 26, (file)

`reference` (*COMPONENT* `component`) [Method]
Render unreferenced COMPONENT.

Source [`asdf.lisp`], page 26, (file)

`reference` (*PACKAGE* `package`) [Method]
Render PACKAGE's reference.

Source [`package.lisp`], page 26, (file)

`reference` (*DEFINITION* `definition`) [Method]
Render DEFINITION's reference.

Source [`symbol.lisp`], page 24, (file)

`render-dependency` *DEPENDENCY-DEF COMPONENT RELATIVE-TO* [Generic Function]
Render COMPONENT's DEPENDENCY-DEF RELATIVE-TO.
Dependencies are referenced only if they are RELATIVE-TO the system being documented. Otherwise, they are just listed.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 26, (file)

Methods

`render-dependency` *SIMPLE-COMPONENT-NAME COMPONENT RELATIVE-TO* **&aux** *DEPENDENCY* [Method]
Render COMPONENT's SIMPLE-COMPONENT-NAME dependency RELATIVE-TO.

`render-dependency` (*DEPENDENCY-DEF* `list`) *COMPONENT RELATIVE-TO* [Method]
Render COMPONENT's DEPENDENCY-DEF (a list) RELATIVE-TO.

`reveal` *OBJECT* [Generic Function]
Replace invisible characters in OBJECT with unicode symbols.

Package [`net.didierverna.declt`], page 29,

Source [`texi.lisp`], page 10, (file)

Methods

`reveal` (*CHAR* `character`) [Method]

`reveal` (*STRING* `string`) [Method]

`source` *ITEM* [Generic Function]
Return ITEM's definition source pathname.

Package [`net.didierverna.declt`], page 29,

Source [`item.lisp`], page 12, (file)

Methods

`source` (*PACKAGE* `package`) [Method]
Return PACKAGE's definition source.

Source [`package.lisp`], page 23, (file)

`source` (*TYPE* `type-definition`) [Method]
Return TYPE's definition source.

Source [`symbol.lisp`], page 12, (file)

`source` (*CLASS* `class-definition`) [Method]
Return CLASS's definition source.

Source [`symbol.lisp`], page 12, (file)

`source` (*STRUCTURE* `structure-definition`) [Method]
Return STRUCTURE's definition source.

Source [`symbol.lisp`], page 12, (file)

`source` (*CONDITION* `condition-definition`) [Method]
Return CONDITION's definition source.

Source [`symbol.lisp`], page 12, (file)

`source` (*COMBINATION* `combination-definition`) [Method]
Return method COMBINATION's definition source.

Source [`symbol.lisp`], page 12, (file)

`source` (*METHOD* `method-definition`) [Method]
Return METHOD's definition source.

Source [`symbol.lisp`], page 12, (file)

`source` (*EXPANDER* `setf-expander-definition`) **&aux** *UPDATE* [Method]

Source [`symbol.lisp`], page 12, (file)

`source` (*FUNCOID* `funcoid-definition`) [Method]
Return FUNCOID's definition source.

Source [`symbol.lisp`], page 12, (file)

`source` (*SYMBOL-MACRO* `symbol-macro-definition`) [Method]
Return SYMBOL-MACRO's definition source.

Source [`symbol.lisp`], page 12, (file)

`source` (*SPECIAL* `special-definition`) [Method]
Return SPECIAL's definition source.

Source [`symbol.lisp`], page 12, (file)

`source` (*CONSTANT* `constant-definition`) [Method]
Return CONSTANT's definition source.

Source [`symbol.lisp`], page 12, (file)

`system-dependency-subsystem` *DEPENDENCY-DEF SYSTEM RELATIVE-TO* [Generic Function]

Return SYSTEM's subsystem from DEPENDENCY-DEF or nil.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 10, (file)

Methods

`system-dependency-subsystem` [Method]
SIMPLE-COMPONENT-NAME SYSTEM RELATIVE-TO **&aux** *DEPENDENCY*

Return SYSTEM's subsystem named SIMPLE-COMPONENT-NAME or nil.

`system-dependency-subsystem` (*DEPENDENCY-DEF* `list`) *SYSTEM RELATIVE-TO* [Method]

Return SYSTEM's subsystem from DEPENDENCY-DEF or nil.

`title` *ITEM* [Generic Function]

Return ITEM's title.

Package [`net.didierverna.declt`], page 29,

Source [`doc.lisp`], page 24, (file)

Methods

`title` (*COMPONENT* `component`) [Method]

Return COMPONENT's title.

Source [`asdf.lisp`], page 26, (file)

`title` (*PACKAGE* `package`) [Method]

Return PACKAGE's title.

Source [`package.lisp`], page 26, (file)

`title` *ITEM* `around` [Method]

Surround ITEM's title with "the [...] <type>".

`type-name` *ITEM* [Generic Function]

Return ITEM's type name.

Package [`net.didierverna.declt`], page 29,

Source [`item.lisp`], page 12, (file)

Methods

`type-name` (*SYSTEM* `system`) [Method]

Return "system"

Source [`asdf.lisp`], page 23, (file)

`type-name` (*MODULE* `module`) [Method]

Return "module"

Source [`asdf.lisp`], page 23, (file)

`type-name` (*SOURCE-FILE* `source-file`) [Method]

Return "file"

Source [`asdf.lisp`], page 23, (file)

`type-name` (*PACKAGE* `package`) [Method]
Return "package".

Source [`package.lisp`], page 23, (file)

`type-name` (*TYPE* `type-definition`) [Method]
Return "type"

Source [`symbol.lisp`], page 12, (file)

`type-name` (*CLASS* `class-definition`) [Method]
Return "class"

Source [`symbol.lisp`], page 12, (file)

`type-name` (*STRUCTURE* `structure-definition`) [Method]
Return "structure"

Source [`symbol.lisp`], page 12, (file)

`type-name` (*CONDITION* `condition-definition`) [Method]
Return "condition"

Source [`symbol.lisp`], page 12, (file)

`type-name` (*COMBINATION* `long-combination-definition`) [Method]
Return "long method combination".

Source [`symbol.lisp`], page 12, (file)

`type-name` (*COMBINATION* `short-combination-definition`) [Method]
Return "short method combination".

Source [`symbol.lisp`], page 12, (file)

`type-name` (*EXPANDER* `setf-expander-definition`) [Method]
Return "setf expander"

Source [`symbol.lisp`], page 12, (file)

`type-name` (*METHOD* `method-definition`) [Method]
Return "method"

Source [`symbol.lisp`], page 12, (file)

`type-name` (*GENERIC* `generic-definition`) [Method]
Return "generic function"

Source [`symbol.lisp`], page 12, (file)

`type-name` (*FUNCTION* `function-definition`) [Method]
Return "function"

Source [`symbol.lisp`], page 12, (file)

`type-name` (*COMPILER-MACRO* `compiler-macro-definition`) [Method]
Return "compiler macro"

Source [`symbol.lisp`], page 12, (file)

`type-name` (*MACRO* `macro-definition`) [Method]
Return "macro"

Source [`symbol.lisp`], page 12, (file)

`type-name` (*SYMBOL-MACRO* `symbol-macro-definition`) [Method]
Return "symbol macro"

Source [`symbol.lisp`], page 12, (file)

`type-name` (*SPECIAL* `special-definition`) [Method]
Return "special variable"

Source [`symbol.lisp`], page 12, (file)

`type-name` (*CONSTANT* `constant-definition`) [Method]
Return "constant"

Source [`symbol.lisp`], page 12, (file)

`virtual-path` *COMPONENT* [Generic Function]
Return CONMPONENT's virtual path.
This is the string of successive component names to access COMPONENT from the toplevel system, separated by slashes. File components also get their extension at the end.

Package [`net.didierverna.declt`], page 29,

Source [`asdf.lisp`], page 26, (file)

Methods

`virtual-path` *COMPONENT* [Method]
Default method for all components.

`virtual-path` (*SOURCE-FILE* `source-file`) **&aux** *VIRTUAL-PATH EXTENSION* `around` [Method]
Potentially add SOURCE-FILE's extension at the end of the virtual path.

`writer-definitions` *SLOT POOL1 POOL2* [Generic Function]
Return a list of writer definitions for SLOT.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

Methods

`writer-definitions` *SLOT POOL1 POOL2* [Method]
Default method for class and condition slots.

`writer-definitions` (*SLOT* `structure-direct-slot-definition`) *POOL1 POOL2* [Method]
Method for structure slots.

6.2.5 Structures

`accessor-definition` () [Structure]
Structure for accessor function definitions.
This structure holds a writer and a setf expander definition that expands this function.

Package [`net.didierverna.declt`], page 29,

Source [symbol.lisp], page 12, (file)

Direct superclasses
[function-definition], page 110, (structure)

Direct methods
- [document], page 95, (method)
- [definition-file-definitions], page 90, (method)
- [definition-package-definitions], page 91, (method)

Direct slots

writer [Slot]
Readers [accessor-definition-writer], page 57, (function)
Writers [(setf accessor-definition-writer)], page 57, (function)

access-expander [Slot]
Readers [accessor-definition-access-expander], page 56, (function)
Writers [(setf accessor-definition-access-expander)], page 56, (function)

accessor-method-definition () [Structure]
Structure for accessor method definitions.
This structure holds the writer method definition.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

Direct superclasses
[method-definition], page 113, (structure)

Direct methods
- [document], page 94, (method)
- [definition-file-definitions], page 90, (method)
- [definition-package-definitions], page 91, (method)

Direct slots

writer [Slot]
Readers [accessor-method-definition-writer], page 57, (function)
Writers [(setf accessor-method-definition-writer)], page 57, (function)

class-definition () [Structure]
Structure for class definitions.
This structure holds the direct superclasses and direct subclasses definitions.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

Direct superclasses
[classoid-definition], page 107, (structure)

Direct methods
- [document], page 93, (method)
- [index], page 97, (method)

- [type-name], page 104, (method)
- [source], page 102, (method)

`classoid-definition` () [Structure]

Base structure for class-like (supporting inheritance) values.
This structure holds links to the direct ancestors and descendants definitions, direct methods definitions and direct slots.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

Direct superclasses

[definition], page 109, (structure)

Direct subclasses

- [condition-definition], page 108, (structure)
- [structure-definition], page 117, (structure)
- [class-definition], page 106, (structure)

Direct methods

[docstring], page 92, (method)

Direct slots

`parents` [Slot]

Readers [classoid-definition-parents], page 60, (function)

Writers [(setf classoid-definition-parents)], page 60, (function)

`children` [Slot]

Readers [classoid-definition-children], page 60, (function)

Writers [(setf classoid-definition-children)], page 60, (function)

`methods` [Slot]

Readers [classoid-definition-methods], page 60, (function)

Writers [(setf classoid-definition-methods)], page 60, (function)

`slots` [Slot]

Readers [classoid-definition-slots], page 61, (function)

Writers [(setf classoid-definition-slots)], page 61, (function)

`combination-definition` () [Structure]

Structure for method combination definitions.
This structure holds the method combination object and a list of users, that is, generic functions using this method combination.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

Direct superclasses

[definition], page 109, (structure)

Direct subclasses

- [short-combination-definition], page 116, (structure)
- [long-combination-definition], page 113, (structure)

Direct methods

- [docstring], page 92, (method)
- [source], page 102, (method)

Direct slots

`combination` [Slot]

Readers [combination-definition-combination], page 61, (function)

Writers [(setf combination-definition-combination)], page 61, (function)

`users` [Slot]

Readers [combination-definition-users], page 61, (function)

Writers [(setf combination-definition-users)], page 61, (function)

`compiler-macro-definition` () [Structure]

Structure for compiler macro definitions.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

Direct superclasses

[funcoid-definition], page 110, (structure)

Direct methods

- [document], page 95, (method)
- [index], page 98, (method)
- [type-name], page 104, (method)
- [docstring], page 92, (method)

`condition-definition` () [Structure]

Structure for condition definitions.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

Direct superclasses

[classoid-definition], page 107, (structure)

Direct methods

- [document], page 94, (method)
- [index], page 97, (method)
- [type-name], page 104, (method)
- [source], page 102, (method)

`constant-definition` () [Structure]

Structure for constant definitions.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

Direct superclasses

[definition], page 109, (structure)

Direct methods

- [document], page 95, (method)

- [index], page 98, (method)
- [type-name], page 105, (method)
- [docstring], page 92, (method)
- [source], page 102, (method)

`context` () [Structure]

The documentation context structure.

Package [net.didierverna.declt], page 29,

Source [doc.lisp], page 24, (file)

Direct superclasses

`structure-object` (structure)

Direct slots

`systems` [Slot]

Readers [context-systems], page 64, (function)

Writers [(setf context-systems)], page 64, (function)

`packages` [Slot]

Readers [context-packages], page 63, (function)

Writers [(setf context-packages)], page 63, (function)

`external-definitions` [Slot]

Readers [context-external-definitions], page 63, (function)

Writers [(setf context-external-definitions)], page 63, (function)

`internal-definitions` [Slot]

Readers [context-internal-definitions], page 63, (function)

Writers [(setf context-internal-definitions)], page 63, (function)

`hyperlinksp` [Slot]

Readers [context-hyperlinksp], page 63, (function)

Writers [(setf context-hyperlinksp)], page 63, (function)

`definition` () [Structure]

Base structure for definitions named by symbols.
This structure holds the symbol naming the definition and a slot for marking foreign definitions, i.e. those which do not pertain to the system being documented.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

Direct superclasses

`structure-object` (structure)

Direct subclasses

- [constant-definition], page 108, (structure)
- [special-definition], page 117, (structure)
- [symbol-macro-definition], page 117, (structure)
- [funcoid-definition], page 110, (structure)

- [method-definition], page 113, (structure)
- [setf-expander-definition], page 115, (structure)
- [slot-definition], page 116, (structure)
- [combination-definition], page 107, (structure)
- [classoid-definition], page 107, (structure)
- [type-definition], page 117, (structure)

Direct methods

- [reference], page 101, (method)
- [name], page 99, (method)

Direct slots

`symbol` [Slot]

Readers [definition-symbol], page 67, (function)

Writers [(setf definition-symbol)], page 67, (function)

`foreignp` [Slot]

Readers [definition-foreignp], page 66, (function)

Writers [(setf definition-foreignp)], page 66, (function)

`funcoid-definition` () [Structure]

Base structure for definitions of functional values.
This structure holds the generic, ordinary or macro function.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

Direct superclasses

[definition], page 109, (structure)

Direct subclasses

- [macro-definition], page 113, (structure)
- [compiler-macro-definition], page 108, (structure)
- [function-definition], page 110, (structure)
- [generic-definition], page 111, (structure)

Direct methods

- [docstring], page 92, (method)
- [source], page 102, (method)
- [lambda-list], page 99, (method)

Direct slots

`function` [Slot]

Readers [funcoid-definition-function], page 68, (function)

Writers [(setf funcoid-definition-function)], page 68, (function)

`function-definition` () [Structure]

Structure for ordinary function definitions.
This structure holds a setf expander definition that expands to this function.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

Direct superclasses
[funcoid-definition], page 110, (structure)

Direct subclasses
- [writer-definition], page 118, (structure)
- [accessor-definition], page 105, (structure)

Direct methods
- [document], page 95, (method)
- [index], page 98, (method)
- [headline-function], page 96, (method)
- [type-name], page 104, (method)

Direct slots

update-expander [Slot]
Readers [function-definition-update-expander], page 69, (function)
Writers [(setf function-definition-update-expander)], page 69, (function)

generic-accessor-definition () [Structure]
Structure for generic accessor function definitions.
This structure holds a generic writer and a setf expander definition that expands this function.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

Direct superclasses
[generic-definition], page 111, (structure)

Direct methods
- [document], page 94, (method)
- [definition-file-definitions], page 91, (method)
- [definition-package-definitions], page 91, (method)
- [definition-combination-users], page 90, (method)

Direct slots

writer [Slot]
Readers [generic-accessor-definition-writer], page 70, (function)
Writers [(setf generic-accessor-definition-writer)], page 70, (function)

access-expander [Slot]
Readers [generic-accessor-definition-access-expander], page 69, (function)
Writers [(setf generic-accessor-definition-access-expander)], page 69, (function)

generic-definition () [Structure]
Structure for generic function definitions.
This structure holds a setf expander definition that expands to this function, the combination definition and the list of method definitions.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

Direct superclasses
[funcoid-definition], page 110, (structure)

Direct subclasses
- [generic-writer-definition], page 112, (structure)
- [generic-accessor-definition], page 111, (structure)

Direct methods
- [document], page 94, (method)
- [index], page 98, (method)
- [headline-function], page 96, (method)
- [definition-file-definitions], page 91, (method)
- [definition-package-definitions], page 91, (method)
- [type-name], page 104, (method)
- [definition-combination-users], page 90, (method)

Direct slots

update-expander [Slot]
Readers [generic-definition-update-expander], page 71, (function)
Writers [(setf generic-definition-update-expander)], page 71, (function)

combination [Slot]
Readers [generic-definition-combination], page 70, (function)
Writers [(setf generic-definition-combination)], page 70, (function)

methods [Slot]
Readers [generic-definition-methods], page 70, (function)
Writers [(setf generic-definition-methods)], page 70, (function)

generic-writer-definition () [Structure]
Structure for generic writer function definitions.
This structure holds the corresponding reader definition.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

Direct superclasses
[generic-definition], page 111, (structure)

Direct methods
- [document], page 94, (method)
- [docstring], page 92, (method)
- [name], page 99, (method)

Direct slots

reader [Slot]
Readers [generic-writer-definition-reader], page 71, (function)
Writers [(setf generic-writer-definition-reader)], page 71, (function)

`long-combination-definition` () [Structure]
Structure for long method combination definitions.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

Direct superclasses
[`combination-definition`], page 107, (structure)

Direct methods
- [`document`], page 94, (method)
- [`index`], page 97, (method)
- [`type-name`], page 104, (method)

`macro-definition` () [Structure]
Structure for macro definitions.
This structure holds a setf expander definition that expands this macro and a setf expander definition that expands to this macro.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

Direct superclasses
[`funcoid-definition`], page 110, (structure)

Direct methods
- [`document`], page 95, (method)
- [`index`], page 98, (method)
- [`definition-file-definitions`], page 90, (method)
- [`definition-package-definitions`], page 91, (method)
- [`type-name`], page 105, (method)

Direct slots

`access-expander` [Slot]
Readers [`macro-definition-access-expander`], page 72, (function)
Writers [`(setf macro-definition-access-expander)`], page 72, (function)

`update-expander` [Slot]
Readers [`macro-definition-update-expander`], page 73, (function)
Writers [`(setf macro-definition-update-expander)`], page 73, (function)

`method-definition` () [Structure]
Base structure for method definitions.
This structure holds the method object.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

Direct superclasses
[`definition`], page 109, (structure)

Direct subclasses
- [`writer-method-definition`], page 118, (structure)

- [accessor-method-definition], page 106, (structure)

Direct methods

- [document], page 94, (method)
- [index], page 98, (method)
- [anchor-name], page 89, (method)
- [type-name], page 104, (method)
- [docstring], page 92, (method)
- [source], page 102, (method)
- [lambda-list], page 99, (method)

Direct slots

`method` [Slot]

Readers [method-definition-method], page 77, (function)

Writers [(setf method-definition-method)], page 77, (function)

`node` () [Structure]

The NODE structure.
This structure holds Texinfo nodes.

Package [net.didierverna.declt], page 29,

Source [texi.lisp], page 10, (file)

Direct superclasses

`structure-object` (structure)

Direct slots

`name` [Slot]

Readers [node-name], page 78, (function)

Writers [(setf node-name)], page 78, (function)

`synopsis` [Slot]

Readers [node-synopsis], page 78, (function)

Writers [(setf node-synopsis)], page 78, (function)

`section-type` [Slot]

Initform `:numbered`

Readers [node-section-type], page 78, (function)

Writers [(setf node-section-type)], page 78, (function)

`section-name` [Slot]

Readers [node-section-name], page 78, (function)

Writers [(setf node-section-name)], page 78, (function)

`next` [Slot]

Readers [node-next], page 78, (function)

Writers [(setf node-next)], page 78, (function)

`previous` [Slot]

Readers [`node-previous`], page 78, (function)

Writers [`(setf node-previous)`], page 78, (function)

`up` [Slot]

Readers [`node-up`], page 78, (function)

Writers [`(setf node-up)`], page 78, (function)

`children` [Slot]

Readers [`node-children`], page 78, (function)

Writers [`(setf node-children)`], page 78, (function)

`before-menu-contents` [Slot]

Readers [`node-before-menu-contents`], page 78, (function)

Writers [`(setf node-before-menu-contents)`], page 78, (function)

`after-menu-contents` [Slot]

Readers [`node-after-menu-contents`], page 77, (function)

Writers [`(setf node-after-menu-contents)`], page 77, (function)

`setf-expander-definition ()` [Structure]

Structure for setf expander definitions.
This structure holds the access definition and the update object. For short forms, this object is a macro or (generic) function definition. For long forms, it's a function.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

Direct superclasses

[`definition`], page 109, (structure)

Direct methods

- [`document`], page 94, (method)
- [`index`], page 98, (method)
- [`headline-function`], page 96, (method)
- [`type-name`], page 104, (method)
- [`docstring`], page 92, (method)
- [`source`], page 102, (method)
- [`name`], page 99, (method)
- [`lambda-list`], page 99, (method)

Direct slots

`access` [Slot]

Readers [`setf-expander-definition-access`], page 83, (function)

Writers [`(setf setf-expander-definition-access)`], page 83, (function)

`update` [Slot]

Readers [`setf-expander-definition-update`], page 83, (function)

Writers [`(setf setf-expander-definition-update)`], page 83, (function)

`short-combination-definition` () [Structure]

Structure for short method combination definitions.
This structure holds the operator definition.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

Direct superclasses

[`combination-definition`], page 107, (structure)

Direct methods

- [`document`], page 94, (method)
- [`index`], page 97, (method)
- [`type-name`], page 104, (method)

Direct slots

`operator` [Slot]

Readers [`short-combination-definition-operator`], page 83, (function)

Writers [`(setf short-combination-definition-operator)`], page 83, (function)

`slot-definition` () [Structure]

Structure for slot definitions.
This structure holds the slot object and the readers and writers definitions.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

Direct superclasses

[`definition`], page 109, (structure)

Direct methods

- [`index`], page 98, (method)
- [`docstring`], page 92, (method)

Direct slots

`slot` [Slot]

Readers [`slot-definition-slot`], page 84, (function)

Writers [`(setf slot-definition-slot)`], page 84, (function)

`readers` [Slot]

Readers [`slot-definition-readers`], page 84, (function)

Writers [`(setf slot-definition-readers)`], page 84, (function)

`writers` [Slot]

Readers [`slot-definition-writers`], page 84, (function)

Writers [`(setf slot-definition-writers)`], page 84, (function)

`special-definition` () [Structure]
Structure for special variables definitions.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

Direct superclasses
[`definition`], page 109, (structure)

Direct methods
- [`document`], page 95, (method)
- [`index`], page 98, (method)
- [`type-name`], page 105, (method)
- [`docstring`], page 92, (method)
- [`source`], page 102, (method)

`structure-definition` () [Structure]
Structure for structure definition.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

Direct superclasses
[`classoid-definition`], page 107, (structure)

Direct methods
- [`document`], page 93, (method)
- [`index`], page 97, (method)
- [`type-name`], page 104, (method)
- [`source`], page 102, (method)

`symbol-macro-definition` () [Structure]
Structure for symbol macro definitions.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

Direct superclasses
[`definition`], page 109, (structure)

Direct methods
- [`document`], page 95, (method)
- [`index`], page 98, (method)
- [`type-name`], page 105, (method)
- [`docstring`], page 92, (method)
- [`source`], page 102, (method)

`type-definition` () [Structure]
Structure for type definitions.

Package [`net.didierverna.declt`], page 29,

Source [`symbol.lisp`], page 12, (file)

Direct superclasses
[`definition`], page 109, (structure)

Direct methods

- [document], page 93, (method)
- [index], page 97, (method)
- [type-name], page 104, (method)
- [docstring], page 91, (method)
- [source], page 102, (method)
- [lambda-list], page 99, (method)

`writer-definition` () [Structure]

Structure for ordinary writer function definitions.
This structure holds the corresponding reader definition.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

Direct superclasses

[function-definition], page 110, (structure)

Direct methods

- [document], page 95, (method)
- [docstring], page 92, (method)
- [name], page 99, (method)

Direct slots

`reader` [Slot]

Readers [writer-definition-reader], page 88, (function)

Writers [(setf writer-definition-reader)], page 88, (function)

`writer-method-definition` () [Structure]

Structure for writer method definitions.

Package [net.didierverna.declt], page 29,

Source [symbol.lisp], page 12, (file)

Direct superclasses

[method-definition], page 113, (structure)

Direct methods

[name], page 99, (method)

Appendix A Indexes

A.1 Concepts

F

L

M

N

A.2 Functions

%

(

@

A

B

C

D

E

F

G

H

I

L

M

N

P

Q

R

S

T

V

W

A.3 Variables

*

A

B

C

E

F

H

I

M

N

O

P

R

S

U

W

A.4 Data types