

# 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.

Copyright © 2010–2013, 2015–2017 Didier Verna

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided also that the section entitled “Copying” is included exactly as in the original.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be translated as well.

# Table of Contents

Copying .....	1
<b>1 Introduction .....</b>	<b>3</b>
<b>2 Systems .....</b>	<b>5</b>
2.1 net.didierverna.declt.....	5
2.2 net.didierverna.declt.core.....	5
2.3 net.didierverna.declt.setup.....	6
<b>3 Modules .....</b>	<b>7</b>
3.1 net.didierverna.declt.core/src.....	7
3.2 net.didierverna.declt.core/src/util.....	7
3.3 net.didierverna.declt.core/src/item.....	7
3.4 net.didierverna.declt.core/src/doc.....	7
<b>4 Files .....</b>	<b>9</b>
4.1 Lisp.....	9
4.1.1 net.didierverna.declt.asd.....	9
4.1.2 net.didierverna.declt.core.asd.....	9
4.1.3 net.didierverna.declt.setup.asd.....	9
4.1.4 net.didierverna.declt.core/quickutil.lisp.....	9
4.1.5 net.didierverna.declt.core/meta.lisp.....	9
4.1.6 net.didierverna.declt.core/src/util/misc.lisp.....	10
4.1.7 net.didierverna.declt.core/src/util/asdf.lisp.....	10
4.1.8 net.didierverna.declt.core/src/util/texti.lisp.....	10
4.1.9 net.didierverna.declt.core/src/item/item.lisp.....	12
4.1.10 net.didierverna.declt.core/src/item/symbol.lisp.....	12
4.1.11 net.didierverna.declt.core/src/item/package.lisp.....	23
4.1.12 net.didierverna.declt.core/src/item/asdf.lisp.....	23
4.1.13 net.didierverna.declt.core/src/doc/doc.lisp.....	24
4.1.14 net.didierverna.declt.core/src/doc/symbol.lisp.....	24
4.1.15 net.didierverna.declt.core/src/doc/package.lisp.....	26
4.1.16 net.didierverna.declt.core/src/doc/asdf.lisp.....	26
4.1.17 net.didierverna.declt.core/src/declt.lisp.....	27
4.1.18 net.didierverna.declt.setup/setup.lisp.....	28
<b>5 Packages .....</b>	<b>29</b>
5.1 quickutil.....	29
5.2 net.didierverna.declt.....	29
5.3 net.didierverna.declt.setup.....	46
<b>6 Definitions .....</b>	<b>47</b>
6.1 Exported definitions.....	47
6.1.1 Special variables.....	47
6.1.2 Macros.....	47

6.1.3	Functions .....	48
6.2	Internal definitions .....	50
6.2.1	Special variables .....	50
6.2.2	Macros .....	50
6.2.3	Functions .....	55
6.2.4	Generic functions .....	89
6.2.5	Structures .....	105
<b>Appendix A Indexes .....</b>		<b>119</b>
A.1	Concepts .....	119
A.2	Functions .....	120
A.3	Variables .....	130
A.4	Data types .....	132

## Copying

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THIS SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.



# 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/txei.lisp

##### Dependency

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

**Parent** [util], page 7, (module)

**Location** core/src/util/txei.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)

- [`@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)
- [`@defsetfx`], page 55, (function)
- [`@defslot`], page 52, (macro)
- [`@defspecial`], page 52, (macro)
- [`@defstruct`], page 52, (macro)
- [`@defsymbolmacro`], page 52, (macro)
- [`@deftp`], page 52, (macro)
- [`@deftype`], page 53, (macro)
- [`@defun`], page 53, (macro)
- [`@defunx`], page 56, (function)
- [`@defvr`], page 53, (macro)
- [`@item`], page 53, (macro)
- [`@itemize`], page 53, (macro)
- [`@itemize-list`], page 56, (function)
- [`@multitable`], page 53, (macro)
- [`@ref`], page 56, (function)
- [`@table`], page 53, (macro)
- [`@tableitem`], page 54, (macro)
- [`add-child`], page 57, (function)
- [`copy-node`], page 65, (function)
- [`escape`], page 67, (function)
- [`escape-anchor`], page 67, (function)
- [`first-word-length`], page 68, (function)
- [`make-node`], page 75, (function)
- [`name`], page 99, (generic function)
- [`name`], page 100, (method)
- [`name`], page 100, (method)
- [`name`], page 100, (method)
- [`name`], page 100, (method)
- [`name`], page 100, (method)
- [`node`], page 114, (structure)
- [`node-after-menu-contents`], page 77, (function)
- [`(setf node-after-menu-contents)`], page 77, (function)
- [`node-before-menu-contents`], page 78, (function)

- [(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)

- [class-definition-foreignp], page 59, (function)
- [(setf class-definition-foreignp)], page 59, (function)
- [class-definition-methods], page 60, (function)
- [(setf class-definition-methods)], page 60, (function)
- [class-definition-p], page 60, (function)
- [class-definition-parents], page 60, (function)
- [(setf class-definition-parents)], page 60, (function)
- [class-definition-slots], page 60, (function)
- [(setf class-definition-slots)], page 60, (function)
- [class-definition-symbol], page 60, (function)
- [(setf class-definition-symbol)], page 60, (function)
- [classoid-definition], page 107, (structure)
- [classoid-definition-children], page 60, (function)
- [(setf classoid-definition-children)], page 60, (function)
- [classoid-definition-foreignp], page 60, (function)
- [(setf classoid-definition-foreignp)], page 60, (function)
- [classoid-definition-methods], page 60, (function)
- [(setf classoid-definition-methods)], page 60, (function)
- [classoid-definition-p], page 60, (function)
- [classoid-definition-parents], page 60, (function)
- [(setf classoid-definition-parents)], page 60, (function)
- [classoid-definition-slots], page 61, (function)
- [(setf classoid-definition-slots)], page 61, (function)
- [classoid-definition-symbol], page 61, (function)
- [(setf classoid-definition-symbol)], page 61, (function)
- [combination-definition], page 107, (structure)
- [combination-definition-combination], page 61, (function)
- [(setf combination-definition-combination)], page 61, (function)
- [combination-definition-foreignp], page 61, (function)
- [(setf combination-definition-foreignp)], page 61, (function)
- [combination-definition-p], page 61, (function)
- [combination-definition-symbol], page 61, (function)
- [(setf combination-definition-symbol)], page 61, (function)
- [combination-definition-users], page 61, (function)
- [(setf combination-definition-users)], page 61, (function)
- [compiler-macro-definition], page 108, (structure)
- [compiler-macro-definition-foreignp], page 61, (function)
- [(setf compiler-macro-definition-foreignp)], page 61, (function)
- [compiler-macro-definition-function], page 62, (function)
- [(setf compiler-macro-definition-function)], page 62, (function)
- [compiler-macro-definition-p], page 62, (function)
- [compiler-macro-definition-symbol], page 62, (function)
- [(setf compiler-macro-definition-symbol)], page 62, (function)

- [condition-definition], page 108, (structure)
- [condition-definition-children], page 62, (function)
- [(setf condition-definition-children)], page 62, (function)
- [condition-definition-foreignp], page 62, (function)
- [(setf condition-definition-foreignp)], page 62, (function)
- [condition-definition-methods], page 62, (function)
- [(setf condition-definition-methods)], page 62, (function)
- [condition-definition-p], page 62, (function)
- [condition-definition-parents], page 62, (function)
- [(setf condition-definition-parents)], page 62, (function)
- [condition-definition-slots], page 62, (function)
- [(setf condition-definition-slots)], page 62, (function)
- [condition-definition-symbol], page 63, (function)
- [(setf condition-definition-symbol)], page 63, (function)
- [constant-definition], page 108, (structure)
- [constant-definition-foreignp], page 63, (function)
- [(setf constant-definition-foreignp)], page 63, (function)
- [constant-definition-p], page 63, (function)
- [constant-definition-symbol], page 63, (function)
- [(setf constant-definition-symbol)], page 63, (function)
- [copy-accessor-definition], page 64, (function)
- [copy-accessor-method-definition], page 64, (function)
- [copy-class-definition], page 64, (function)
- [copy-classoid-definition], page 64, (function)
- [copy-combination-definition], page 64, (function)
- [copy-compiler-macro-definition], page 64, (function)
- [copy-condition-definition], page 64, (function)
- [copy-constant-definition], page 64, (function)
- [copy-definition], page 64, (function)
- [copy-funcoid-definition], page 64, (function)
- [copy-function-definition], page 65, (function)
- [copy-generic-accessor-definition], page 65, (function)
- [copy-generic-definition], page 65, (function)
- [copy-generic-writer-definition], page 65, (function)
- [copy-long-combination-definition], page 65, (function)
- [copy-macro-definition], page 65, (function)
- [copy-method-definition], page 65, (function)
- [copy-setf-expander-definition], page 65, (function)
- [copy-short-combination-definition], page 65, (function)
- [copy-slot-definition], page 65, (function)
- [copy-special-definition], page 65, (function)
- [copy-structure-definition], page 66, (function)
- [copy-symbol-macro-definition], page 66, (function)



- [(setf funcoid-definition-function)], page 68, (function)
- [funcoid-definition-p], page 68, (function)
- [funcoid-definition-symbol], page 68, (function)
- [(setf funcoid-definition-symbol)], page 68, (function)
- [function-definition], page 110, (structure)
- [function-definition-foreignp], page 68, (function)
- [(setf function-definition-foreignp)], page 68, (function)
- [function-definition-function], page 69, (function)
- [(setf function-definition-function)], page 69, (function)
- [function-definition-p], page 69, (function)
- [function-definition-symbol], page 69, (function)
- [(setf function-definition-symbol)], page 69, (function)
- [function-definition-update-expander], page 69, (function)
- [(setf function-definition-update-expander)], page 69, (function)
- [generic-accessor-definition], page 111, (structure)
- [generic-accessor-definition-access-expander], page 69, (function)
- [(setf generic-accessor-definition-access-expander)], page 69, (function)
- [generic-accessor-definition-combination], page 69, (function)
- [(setf generic-accessor-definition-combination)], page 69, (function)
- [generic-accessor-definition-foreignp], page 69, (function)
- [(setf generic-accessor-definition-foreignp)], page 69, (function)
- [generic-accessor-definition-function], page 69, (function)
- [(setf generic-accessor-definition-function)], page 69, (function)
- [generic-accessor-definition-methods], page 69, (function)
- [(setf generic-accessor-definition-methods)], page 69, (function)
- [generic-accessor-definition-p], page 70, (function)
- [generic-accessor-definition-symbol], page 70, (function)
- [(setf generic-accessor-definition-symbol)], page 70, (function)
- [generic-accessor-definition-update-expander], page 70, (function)
- [(setf generic-accessor-definition-update-expander)], page 70, (function)
- [generic-accessor-definition-writer], page 70, (function)
- [(setf generic-accessor-definition-writer)], page 70, (function)
- [generic-definition], page 111, (structure)
- [generic-definition-combination], page 70, (function)
- [(setf generic-definition-combination)], page 70, (function)
- [generic-definition-foreignp], page 70, (function)
- [(setf generic-definition-foreignp)], page 70, (function)
- [generic-definition-function], page 70, (function)
- [(setf generic-definition-function)], page 70, (function)
- [generic-definition-methods], page 70, (function)
- [(setf generic-definition-methods)], page 70, (function)

- [generic-definition-p], page 70, (function)
- [generic-definition-symbol], page 70, (function)
- [(setf generic-definition-symbol)], page 70, (function)
- [generic-definition-update-expander], page 71, (function)
- [(setf generic-definition-update-expander)], page 71, (function)
- [generic-writer-definition], page 112, (structure)
- [generic-writer-definition-combination], page 71, (function)
- [(setf generic-writer-definition-combination)], page 71, (function)
- [generic-writer-definition-foreignp], page 71, (function)
- [(setf generic-writer-definition-foreignp)], page 71, (function)
- [generic-writer-definition-function], page 71, (function)
- [(setf generic-writer-definition-function)], page 71, (function)
- [generic-writer-definition-methods], page 71, (function)
- [(setf generic-writer-definition-methods)], page 71, (function)
- [generic-writer-definition-p], page 71, (function)
- [generic-writer-definition-reader], page 71, (function)
- [(setf generic-writer-definition-reader)], page 71, (function)
- [generic-writer-definition-symbol], page 71, (function)
- [(setf generic-writer-definition-symbol)], page 71, (function)
- [generic-writer-definition-update-expander], page 71, (function)
- [(setf generic-writer-definition-update-expander)], page 71, (function)
- [lambda-list], page 98, (generic function)
- [lambda-list], page 98, (method)
- [lambda-list], page 99, (method)
- [lambda-list], page 99, (method)
- [lambda-list], page 99, (method)
- [lambda-list], page 99, (method)
- [long-combination-definition], page 113, (structure)
- [long-combination-definition-combination], page 72, (function)
- [(setf long-combination-definition-combination)], page 72, (function)
- [long-combination-definition-foreignp], page 72, (function)
- [(setf long-combination-definition-foreignp)], page 72, (function)
- [long-combination-definition-p], page 72, (function)
- [long-combination-definition-symbol], page 72, (function)
- [(setf long-combination-definition-symbol)], page 72, (function)
- [long-combination-definition-users], page 72, (function)
- [(setf long-combination-definition-users)], page 72, (function)
- [macro-definition], page 113, (structure)
- [macro-definition-access-expander], page 72, (function)
- [(setf macro-definition-access-expander)], page 72, (function)
- [macro-definition-foreignp], page 72, (function)
- [(setf macro-definition-foreignp)], page 72, (function)

- [macro-definition-function], page 73, (function)
- [(setf macro-definition-function)], page 73, (function)
- [macro-definition-p], page 73, (function)
- [macro-definition-symbol], page 73, (function)
- [(setf macro-definition-symbol)], page 73, (function)
- [macro-definition-update-expander], page 73, (function)
- [(setf macro-definition-update-expander)], page 73, (function)
- [make-accessor-definition], page 73, (function)
- [make-accessor-method-definition], page 73, (function)
- [make-class-definition], page 73, (function)
- [make-classoid-definition], page 73, (function)
- [make-combination-definition], page 73, (function)
- [make-compiler-macro-definition], page 74, (function)
- [make-condition-definition], page 74, (function)
- [make-constant-definition], page 74, (function)
- [make-definition], page 74, (function)
- [make-definitions-pool], page 74, (function)
- [make-funcoid-definition], page 74, (function)
- [make-function-definition], page 74, (function)
- [make-generic-accessor-definition], page 75, (function)
- [make-generic-definition], page 75, (function)
- [make-generic-writer-definition], page 75, (function)
- [make-long-combination-definition], page 75, (function)
- [make-macro-definition], page 75, (function)
- [make-method-definition], page 75, (function)
- [make-setf-expander-definition], page 76, (function)
- [make-short-combination-definition], page 76, (function)
- [make-slot-definition], page 76, (function)
- [make-slot-definitions], page 76, (function)
- [make-special-definition], page 76, (function)
- [make-structure-definition], page 76, (function)
- [make-symbol-macro-definition], page 76, (function)
- [make-type-definition], page 76, (function)
- [make-writer-definition], page 76, (function)
- [make-writer-method-definition], page 77, (function)
- [mapcan-definitions-pool], page 77, (function)
- [method-definition], page 113, (structure)
- [method-definition-foreignp], page 77, (function)
- [(setf method-definition-foreignp)], page 77, (function)
- [method-definition-method], page 77, (function)
- [(setf method-definition-method)], page 77, (function)
- [method-definition-p], page 77, (function)
- [method-definition-symbol], page 77, (function)

- [(setf method-definition-symbol)], page 77, (function)
- [method-name], page 77, (function)
- [name], page 99, (method)
- [name], page 99, (method)
- [name], page 99, (method)
- [name], page 99, (method)
- [name], page 99, (method)
- [pool-combination-users], page 79, (function)
- [qualifiers], page 79, (function)
- [reader-definitions], page 100, (generic function)
- [reader-definitions], page 100, (method)
- [reader-definitions], page 100, (method)
- [sbcl-has-setf-inverse-meta-info], page 83, (function)
- [setf-expander-definition], page 115, (structure)
- [setf-expander-definition-access], page 83, (function)
- [(setf setf-expander-definition-access)], page 83, (function)
- [setf-expander-definition-foreignp], page 83, (function)
- [(setf setf-expander-definition-foreignp)], page 83, (function)
- [setf-expander-definition-p], page 83, (function)
- [setf-expander-definition-symbol], page 83, (function)
- [(setf setf-expander-definition-symbol)], page 83, (function)
- [setf-expander-definition-update], page 83, (function)
- [(setf setf-expander-definition-update)], page 83, (function)
- [setf-expander-p], page 83, (function)
- [short-combination-definition], page 116, (structure)
- [short-combination-definition-combination], page 83, (function)
- [(setf short-combination-definition-combination)], page 83, (function)
- [short-combination-definition-foreignp], page 83, (function)
- [(setf short-combination-definition-foreignp)], page 83, (function)
- [short-combination-definition-operator], page 83, (function)
- [(setf short-combination-definition-operator)], page 83, (function)
- [short-combination-definition-p], page 84, (function)
- [short-combination-definition-symbol], page 84, (function)
- [(setf short-combination-definition-symbol)], page 84, (function)
- [short-combination-definition-users], page 84, (function)
- [(setf short-combination-definition-users)], page 84, (function)
- [slot-definition], page 116, (structure)
- [slot-definition-foreignp], page 84, (function)
- [(setf slot-definition-foreignp)], page 84, (function)
- [slot-definition-p], page 84, (function)
- [slot-definition-readers], page 84, (function)
- [(setf slot-definition-readers)], page 84, (function)
- [slot-definition-slot], page 84, (function)



- [(setf slot-definition-slot)], page 84, (function)
- [slot-definition-symbol], page 84, (function)
- [(setf slot-definition-symbol)], page 84, (function)
- [slot-definition-writers], page 84, (function)
- [(setf slot-definition-writers)], page 84, (function)
- [slot-property], page 84, (function)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [special-definition], page 117, (structure)
- [special-definition-foreignp], page 85, (function)
- [(setf special-definition-foreignp)], page 85, (function)
- [special-definition-p], page 85, (function)
- [special-definition-symbol], page 85, (function)
- [(setf special-definition-symbol)], page 85, (function)
- [specializers], page 85, (function)
- [structure-definition], page 117, (structure)
- [structure-definition-children], page 85, (function)
- [(setf structure-definition-children)], page 85, (function)
- [structure-definition-foreignp], page 85, (function)
- [(setf structure-definition-foreignp)], page 85, (function)
- [structure-definition-methods], page 85, (function)
- [(setf structure-definition-methods)], page 85, (function)
- [structure-definition-p], page 85, (function)
- [structure-definition-parents], page 85, (function)
- [(setf structure-definition-parents)], page 85, (function)
- [structure-definition-slots], page 85, (function)
- [(setf structure-definition-slots)], page 85, (function)
- [structure-definition-symbol], page 86, (function)
- [(setf structure-definition-symbol)], page 86, (function)
- [symbol-macro-definition], page 117, (structure)
- [symbol-macro-definition-foreignp], page 86, (function)
- [(setf symbol-macro-definition-foreignp)], page 86, (function)
- [symbol-macro-definition-p], page 86, (function)
- [symbol-macro-definition-symbol], page 86, (function)

- [(setf symbol-macro-definition-symbol)], page 86, (function)
- [type-definition], page 117, (structure)
- [type-definition-foreignp], page 87, (function)
- [(setf type-definition-foreignp)], page 87, (function)
- [type-definition-p], page 87, (function)
- [type-definition-symbol], page 87, (function)
- [(setf type-definition-symbol)], page 87, (function)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 105, (method)
- [type-name], page 105, (method)
- [type-name], page 105, (method)
- [type-name], page 105, (method)
- [writer-definition], page 118, (structure)
- [writer-definition-foreignp], page 88, (function)
- [(setf writer-definition-foreignp)], page 88, (function)
- [writer-definition-function], page 88, (function)
- [(setf writer-definition-function)], page 88, (function)
- [writer-definition-p], page 88, (function)
- [writer-definition-reader], page 88, (function)
- [(setf writer-definition-reader)], page 88, (function)
- [writer-definition-symbol], page 88, (function)
- [(setf writer-definition-symbol)], page 88, (function)
- [writer-definition-update-expander], page 88, (function)
- [(setf writer-definition-update-expander)], page 88, (function)
- [writer-definitions], page 105, (generic function)
- [writer-definitions], page 105, (method)
- [writer-definitions], page 105, (method)
- [writer-method-definition], page 118, (structure)
- [writer-method-definition-foreignp], page 88, (function)
- [(setf writer-method-definition-foreignp)], page 88, (function)
- [writer-method-definition-method], page 88, (function)
- [(setf writer-method-definition-method)], page 88, (function)
- [writer-method-definition-p], page 88, (function)
- [writer-method-definition-symbol], page 88, (function)
- [(setf writer-method-definition-symbol)], page 88, (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



- [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)
- [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)

- [`@defsetfx`], page 55, (function)
- [`@defslot`], page 52, (macro)
- [`@defspecial`], page 52, (macro)
- [`@defstruct`], page 52, (macro)
- [`@defsymbolmacro`], page 52, (macro)
- [`@deftp`], page 52, (macro)
- [`@deftype`], page 53, (macro)
- [`@defun`], page 53, (macro)
- [`@defunx`], page 56, (function)
- [`@defvr`], page 53, (macro)
- [`@item`], page 53, (macro)
- [`@itemize`], page 53, (macro)
- [`@itemize-list`], page 56, (function)
- [`@multitable`], page 53, (macro)
- [`@ref`], page 56, (function)
- [`@table`], page 53, (macro)
- [`@tableitem`], page 54, (macro)
- [`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-categories-node`], page 57, (function)
- [`add-category-node`], page 57, (function)

- [add-child], page 57, (function)
- [add-definition], page 58, (function)
- [add-definitions], page 58, (function)
- [add-definitions-node], page 58, (function)
- [add-external-definitions], page 58, (function)
- [add-files-node], page 58, (function)
- [add-internal-definitions], page 58, (function)
- [add-modules-node], page 58, (function)
- [add-packages], page 58, (function)
- [add-packages-node], page 58, (function)
- [add-status-definitions-node], page 59, (function)
- [add-symbol-definition], page 59, (function)
- [add-symbol-definitions], page 59, (function)
- [add-systems-node], page 59, (function)
- [anchor], page 59, (function)
- [anchor-and-index], page 59, (function)
- [anchor-name], page 89, (generic function)
- [anchor-name], page 89, (method)
- [anchor-name], page 89, (method)
- [anchor-name], page 89, (method)
- [anchor-name], page 89, (method)
- [anchor-name], page 89, (method)
- [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 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)
- [class-definition-foreignp], page 59, (function)
- [(setf class-definition-foreignp)], page 59, (function)
- [class-definition-methods], page 60, (function)
- [(setf class-definition-methods)], page 60, (function)
- [class-definition-p], page 60, (function)
- [class-definition-parents], page 60, (function)
- [(setf class-definition-parents)], page 60, (function)
- [class-definition-slots], page 60, (function)

- [(setf class-definition-slots)], page 60, (function)
- [class-definition-symbol], page 60, (function)
- [(setf class-definition-symbol)], page 60, (function)
- [classoid-definition], page 107, (structure)
- [classoid-definition-children], page 60, (function)
- [(setf classoid-definition-children)], page 60, (function)
- [classoid-definition-foreignp], page 60, (function)
- [(setf classoid-definition-foreignp)], page 60, (function)
- [classoid-definition-methods], page 60, (function)
- [(setf classoid-definition-methods)], page 60, (function)
- [classoid-definition-p], page 60, (function)
- [classoid-definition-parents], page 60, (function)
- [(setf classoid-definition-parents)], page 60, (function)
- [classoid-definition-slots], page 61, (function)
- [(setf classoid-definition-slots)], page 61, (function)
- [classoid-definition-symbol], page 61, (function)
- [(setf classoid-definition-symbol)], page 61, (function)
- [clindent], page 61, (function)
- [combination-definition], page 107, (structure)
- [combination-definition-combination], page 61, (function)
- [(setf combination-definition-combination)], page 61, (function)
- [combination-definition-foreignp], page 61, (function)
- [(setf combination-definition-foreignp)], page 61, (function)
- [combination-definition-p], page 61, (function)
- [combination-definition-symbol], page 61, (function)
- [(setf combination-definition-symbol)], page 61, (function)
- [combination-definition-users], page 61, (function)
- [(setf combination-definition-users)], page 61, (function)
- [compiler-macro-definition], page 108, (structure)
- [compiler-macro-definition-foreignp], page 61, (function)
- [(setf compiler-macro-definition-foreignp)], page 61, (function)
- [compiler-macro-definition-function], page 62, (function)
- [(setf compiler-macro-definition-function)], page 62, (function)
- [compiler-macro-definition-p], page 62, (function)
- [compiler-macro-definition-symbol], page 62, (function)
- [(setf compiler-macro-definition-symbol)], page 62, (function)
- [components], page 62, (function)
- [condition-definition], page 108, (structure)
- [condition-definition-children], page 62, (function)
- [(setf condition-definition-children)], page 62, (function)
- [condition-definition-foreignp], page 62, (function)
- [(setf condition-definition-foreignp)], page 62, (function)
- [condition-definition-methods], page 62, (function)

- [(setf condition-definition-methods)], page 62, (function)
- [condition-definition-p], page 62, (function)
- [condition-definition-parents], page 62, (function)
- [(setf condition-definition-parents)], page 62, (function)
- [condition-definition-slots], page 62, (function)
- [(setf condition-definition-slots)], page 62, (function)
- [condition-definition-symbol], page 63, (function)
- [(setf condition-definition-symbol)], page 63, (function)
- [constant-definition], page 108, (structure)
- [constant-definition-foreignp], page 63, (function)
- [(setf constant-definition-foreignp)], page 63, (function)
- [constant-definition-p], page 63, (function)
- [constant-definition-symbol], page 63, (function)
- [(setf constant-definition-symbol)], page 63, (function)
- [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-accessor-definition], page 64, (function)
- [copy-accessor-method-definition], page 64, (function)
- [copy-class-definition], page 64, (function)
- [copy-classoid-definition], page 64, (function)
- [copy-combination-definition], page 64, (function)
- [copy-compiler-macro-definition], page 64, (function)
- [copy-condition-definition], page 64, (function)
- [copy-constant-definition], page 64, (function)
- [copy-context], page 64, (function)
- [copy-definition], page 64, (function)
- [copy-funcooid-definition], page 64, (function)
- [copy-function-definition], page 65, (function)
- [copy-generic-accessor-definition], page 65, (function)
- [copy-generic-definition], page 65, (function)
- [copy-generic-writer-definition], page 65, (function)
- [copy-long-combination-definition], page 65, (function)

- [copy-macro-definition], page 65, (function)
- [copy-method-definition], page 65, (function)
- [copy-node], page 65, (function)
- [copy-setf-expander-definition], page 65, (function)
- [copy-short-combination-definition], page 65, (function)
- [copy-slot-definition], page 65, (function)
- [copy-special-definition], page 65, (function)
- [copy-structure-definition], page 66, (function)
- [copy-symbol-macro-definition], page 66, (function)
- [copy-type-definition], page 66, (function)
- [copy-writer-definition], page 66, (function)
- [copy-writer-method-definition], page 66, (function)
- [current-time-string], page 66, (function)
- [defindent], page 54, (macro)
- [definition], page 109, (structure)
- [definition-combination-users], page 90, (generic function)
- [definition-combination-users], page 90, (method)
- [definition-combination-users], page 90, (method)
- [definition-combination-users], page 90, (method)
- [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)
- [definition-foreignp], page 66, (function)
- [(setf definition-foreignp)], page 66, (function)
- [definition-p], page 66, (function)
- [definition-package], page 66, (function)
- [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)
- [definition-package-name], page 66, (function)
- [definition-source], page 66, (function)
- [definition-source-by-name], page 67, (function)
- [definition-symbol], page 67, (function)
- [(setf definition-symbol)], page 67, (function)
- [definitions-pool-size], page 67, (function)



- [endpush], page 54, (macro)
- [escape], page 67, (function)
- [escape-anchor], page 67, (function)
- [file-definitions], page 67, (function)
- [file-node], page 67, (function)
- [file-packages], page 67, (function)
- [finalize-definitions], page 68, (function)
- [find-definition], page 95, (generic function)
- [find-definition], page 96, (method)
- [find-definition], page 96, (method)
- [find-definition], page 96, (method)
- [find-definition], page 96, (method)
- [find-definition], page 96, (method)
- [find-method-definition], page 68, (function)
- [first-word-length], page 68, (function)
- [funcoid-definition], page 110, (structure)
- [funcoid-definition-foreignp], page 68, (function)
- [(setf funcoid-definition-foreignp)], page 68, (function)
- [funcoid-definition-function], page 68, (function)
- [(setf funcoid-definition-function)], page 68, (function)
- [funcoid-definition-p], page 68, (function)
- [funcoid-definition-symbol], page 68, (function)
- [(setf funcoid-definition-symbol)], page 68, (function)
- [function-definition], page 110, (structure)
- [function-definition-foreignp], page 68, (function)
- [(setf function-definition-foreignp)], page 68, (function)
- [function-definition-function], page 69, (function)
- [(setf function-definition-function)], page 69, (function)
- [function-definition-p], page 69, (function)
- [function-definition-symbol], page 69, (function)
- [(setf function-definition-symbol)], page 69, (function)
- [function-definition-update-expander], page 69, (function)
- [(setf function-definition-update-expander)], page 69, (function)
- [generate-quickutils], page 69, (function)
- [generic-accessor-definition], page 111, (structure)
- [generic-accessor-definition-access-expander], page 69, (function)
- [(setf generic-accessor-definition-access-expander)], page 69, (function)
- [generic-accessor-definition-combination], page 69, (function)
- [(setf generic-accessor-definition-combination)], page 69, (function)
- [generic-accessor-definition-foreignp], page 69, (function)
- [(setf generic-accessor-definition-foreignp)], page 69, (function)
- [generic-accessor-definition-function], page 69, (function)



- [(setf generic-accessor-definition-function)], page 69, (function)
- [generic-accessor-definition-methods], page 69, (function)
- [(setf generic-accessor-definition-methods)], page 69, (function)
- [generic-accessor-definition-p], page 70, (function)
- [generic-accessor-definition-symbol], page 70, (function)
- [(setf generic-accessor-definition-symbol)], page 70, (function)
- [generic-accessor-definition-update-expander], page 70, (function)
- [(setf generic-accessor-definition-update-expander)], page 70, (function)
- [generic-accessor-definition-writer], page 70, (function)
- [(setf generic-accessor-definition-writer)], page 70, (function)
- [generic-definition], page 111, (structure)
- [generic-definition-combination], page 70, (function)
- [(setf generic-definition-combination)], page 70, (function)
- [generic-definition-foreignp], page 70, (function)
- [(setf generic-definition-foreignp)], page 70, (function)
- [generic-definition-function], page 70, (function)
- [(setf generic-definition-function)], page 70, (function)
- [generic-definition-methods], page 70, (function)
- [(setf generic-definition-methods)], page 70, (function)
- [generic-definition-p], page 70, (function)
- [generic-definition-symbol], page 70, (function)
- [(setf generic-definition-symbol)], page 70, (function)
- [generic-definition-update-expander], page 71, (function)
- [(setf generic-definition-update-expander)], page 71, (function)
- [generic-writer-definition], page 112, (structure)
- [generic-writer-definition-combination], page 71, (function)
- [(setf generic-writer-definition-combination)], page 71, (function)
- [generic-writer-definition-foreignp], page 71, (function)
- [(setf generic-writer-definition-foreignp)], page 71, (function)
- [generic-writer-definition-function], page 71, (function)
- [(setf generic-writer-definition-function)], page 71, (function)
- [generic-writer-definition-methods], page 71, (function)
- [(setf generic-writer-definition-methods)], page 71, (function)
- [generic-writer-definition-p], page 71, (function)
- [generic-writer-definition-reader], page 71, (function)
- [(setf generic-writer-definition-reader)], page 71, (function)
- [generic-writer-definition-symbol], page 71, (function)
- [(setf generic-writer-definition-symbol)], page 71, (function)
- [generic-writer-definition-update-expander], page 71, (function)
- [(setf generic-writer-definition-update-expander)], page 71, (function)
- [headline-function], page 96, (generic function)



- [(setf long-combination-definition-foreignp)], page 72, (function)
- [long-combination-definition-p], page 72, (function)
- [long-combination-definition-symbol], page 72, (function)
- [(setf long-combination-definition-symbol)], page 72, (function)
- [long-combination-definition-users], page 72, (function)
- [(setf long-combination-definition-users)], page 72, (function)
- [macro-definition], page 113, (structure)
- [macro-definition-access-expander], page 72, (function)
- [(setf macro-definition-access-expander)], page 72, (function)
- [macro-definition-foreignp], page 72, (function)
- [(setf macro-definition-foreignp)], page 72, (function)
- [macro-definition-function], page 73, (function)
- [(setf macro-definition-function)], page 73, (function)
- [macro-definition-p], page 73, (function)
- [macro-definition-symbol], page 73, (function)
- [(setf macro-definition-symbol)], page 73, (function)
- [macro-definition-update-expander], page 73, (function)
- [(setf macro-definition-update-expander)], page 73, (function)
- [make-accessor-definition], page 73, (function)
- [make-accessor-method-definition], page 73, (function)
- [make-class-definition], page 73, (function)
- [make-classoid-definition], page 73, (function)
- [make-combination-definition], page 73, (function)
- [make-compiler-macro-definition], page 74, (function)
- [make-condition-definition], page 74, (function)
- [make-constant-definition], page 74, (function)
- [make-context], page 74, (function)
- [make-definition], page 74, (function)
- [make-definitions-pool], page 74, (function)
- [make-funcoid-definition], page 74, (function)
- [make-function-definition], page 74, (function)
- [make-generic-accessor-definition], page 75, (function)
- [make-generic-definition], page 75, (function)
- [make-generic-writer-definition], page 75, (function)
- [make-long-combination-definition], page 75, (function)
- [make-macro-definition], page 75, (function)
- [make-method-definition], page 75, (function)
- [make-node], page 75, (function)
- [make-setf-expander-definition], page 76, (function)
- [make-short-combination-definition], page 76, (function)
- [make-slot-definition], page 76, (function)
- [make-slot-definitions], page 76, (function)
- [make-special-definition], page 76, (function)

- [make-structure-definition], page 76, (function)
- [make-symbol-macro-definition], page 76, (function)
- [make-type-definition], page 76, (function)
- [make-writer-definition], page 76, (function)
- [make-writer-method-definition], page 77, (function)
- [mapcan-definitions-pool], page 77, (function)
- [method-definition], page 113, (structure)
- [method-definition-foreignp], page 77, (function)
- [(setf method-definition-foreignp)], page 77, (function)
- [method-definition-method], page 77, (function)
- [(setf method-definition-method)], page 77, (function)
- [method-definition-p], page 77, (function)
- [method-definition-symbol], page 77, (function)
- [(setf method-definition-symbol)], page 77, (function)
- [method-name], page 77, (function)
- [module-components], page 77, (function)
- [module-node], page 77, (function)
- [name], page 99, (generic function)
- [name], page 99, (method)
- [name], page 99, (method)
- [name], page 99, (method)
- [name], page 99, (method)
- [name], page 99, (method)
- [name], page 99, (method)
- [name], page 99, (method)
- [name], page 99, (method)
- [name], page 99, (method)
- [name], page 100, (method)
- [name], page 100, (method)
- [name], page 100, (method)
- [name], page 100, (method)
- [name], page 100, (method)
- [name], page 100, (method)
- [node], page 114, (structure)
- [node-after-menu-contents], page 77, (function)
- [(setf node-after-menu-contents)], page 77, (function)
- [node-before-menu-contents], page 78, (function)
- [(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)
- [package-definitions], page 79, (function)
- [package-external-symbols], page 79, (function)
- [package-internal-symbols], page 79, (function)
- [parse-contact(s)], page 79, (function)
- [parse-contact-string], page 79, (function)
- [pool-combination-users], page 79, (function)
- [pretty-specializer], page 100, (generic function)
- [pretty-specializer], page 100, (method)
- [pretty-specializer], page 100, (method)
- [qualifiers], page 79, (function)
- [read-next-line], page 79, (function)
- [reader-definitions], page 100, (generic function)
- [reader-definitions], page 100, (method)
- [reader-definitions], page 100, (method)
- [reference], page 100, (generic function)
- [reference], page 100, (method)
- [reference], page 101, (method)
- [reference], page 101, (method)
- [reference], page 101, (method)
- [reference], page 101, (method)
- [reference], page 101, (method)
- [reference-component], page 80, (function)
- [relative-location], page 80, (function)
- [render-classoid], page 54, (macro)
- [render-combination], page 54, (macro)
- [render-definition-core], 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-docstring], page 80, (function)
- [render-external-definitions-references], page 80, (function)
- [render-funcooid], page 54, (macro)

- [render-header], page 80, (function)
- [render-headline], page 81, (function)
- [render-initargs], page 81, (function)
- [render-internal-definitions-references], page 81, (function)
- [render-lambda-list], page 81, (function)
- [render-location], page 81, (function)
- [render-method], page 54, (macro)
- [render-method-combination], page 81, (function)
- [render-node], page 81, (function)
- [render-packages-references], page 81, (function)
- [render-references], page 82, (function)
- [render-slot], page 82, (function)
- [render-slot-property], page 82, (function)
- [render-slots], page 82, (function)
- [render-source], page 82, (function)
- [render-text], page 82, (function)
- [render-to-string], page 55, (macro)
- [render-top-node], page 82, (function)
- [render-use-list], page 82, (function)
- [render-varoid], page 55, (macro)
- [reveal], page 101, (generic function)
- [reveal], page 101, (method)
- [reveal], page 101, (method)
- [sbcl-has-setf-inverse-meta-info], page 83, (function)
- [setf-expander-definition], page 115, (structure)
- [setf-expander-definition-access], page 83, (function)
- [(setf setf-expander-definition-access)], page 83, (function)
- [setf-expander-definition-foreignp], page 83, (function)
- [(setf setf-expander-definition-foreignp)], page 83, (function)
- [setf-expander-definition-p], page 83, (function)
- [setf-expander-definition-symbol], page 83, (function)
- [(setf setf-expander-definition-symbol)], page 83, (function)
- [setf-expander-definition-update], page 83, (function)
- [(setf setf-expander-definition-update)], page 83, (function)
- [setf-expander-p], page 83, (function)
- [short-combination-definition], page 116, (structure)
- [short-combination-definition-combination], page 83, (function)
- [(setf short-combination-definition-combination)], page 83, (function)
- [short-combination-definition-foreignp], page 83, (function)
- [(setf short-combination-definition-foreignp)], page 83, (function)
- [short-combination-definition-operator], page 83, (function)
- [(setf short-combination-definition-operator)], page 83, (function)
- [short-combination-definition-p], page 84, (function)

- [short-combination-definition-symbol], page 84, (function)
- [(setf short-combination-definition-symbol)], page 84, (function)
- [short-combination-definition-users], page 84, (function)
- [(setf short-combination-definition-users)], page 84, (function)
- [slot-definition], page 116, (structure)
- [slot-definition-foreignp], page 84, (function)
- [(setf slot-definition-foreignp)], page 84, (function)
- [slot-definition-p], page 84, (function)
- [slot-definition-readers], page 84, (function)
- [(setf slot-definition-readers)], page 84, (function)
- [slot-definition-slot], page 84, (function)
- [(setf slot-definition-slot)], page 84, (function)
- [slot-definition-symbol], page 84, (function)
- [(setf slot-definition-symbol)], page 84, (function)
- [slot-definition-writers], page 84, (function)
- [(setf slot-definition-writers)], page 84, (function)
- [slot-property], page 84, (function)
- [source], page 101, (generic function)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [source], page 102, (method)
- [special-definition], page 117, (structure)
- [special-definition-foreignp], page 85, (function)
- [(setf special-definition-foreignp)], page 85, (function)
- [special-definition-p], page 85, (function)
- [special-definition-symbol], page 85, (function)
- [(setf special-definition-symbol)], page 85, (function)
- [specializers], page 85, (function)
- [structure-definition], page 117, (structure)
- [structure-definition-children], page 85, (function)
- [(setf structure-definition-children)], page 85, (function)
- [structure-definition-foreignp], page 85, (function)
- [(setf structure-definition-foreignp)], page 85, (function)
- [structure-definition-methods], page 85, (function)

- [(setf structure-definition-methods)], page 85, (function)
- [structure-definition-p], page 85, (function)
- [structure-definition-parents], page 85, (function)
- [(setf structure-definition-parents)], page 85, (function)
- [structure-definition-slots], page 85, (function)
- [(setf structure-definition-slots)], page 85, (function)
- [structure-definition-symbol], page 86, (function)
- [(setf structure-definition-symbol)], page 86, (function)
- [sub-component-p], page 86, (function)
- [subsystems], page 86, (function)
- [symbol-macro-definition], page 117, (structure)
- [symbol-macro-definition-foreignp], page 86, (function)
- [(setf symbol-macro-definition-foreignp)], page 86, (function)
- [symbol-macro-definition-p], page 86, (function)
- [symbol-macro-definition-symbol], page 86, (function)
- [(setf symbol-macro-definition-symbol)], 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-external-symbols], page 87, (function)
- [system-file-name], page 87, (function)
- [system-file-type], page 87, (function)
- [system-internal-symbols], page 87, (function)
- [system-node], page 87, (function)
- [system-packages], page 87, (function)
- [tilde-reader], page 87, (function)
- [title], page 103, (generic function)
- [title], page 103, (method)
- [title], page 103, (method)
- [title], page 103, (method)
- [type-definition], page 117, (structure)
- [type-definition-foreignp], page 87, (function)
- [(setf type-definition-foreignp)], page 87, (function)
- [type-definition-p], page 87, (function)
- [type-definition-symbol], page 87, (function)
- [(setf type-definition-symbol)], page 87, (function)
- [type-name], page 103, (generic function)
- [type-name], page 103, (method)
- [type-name], page 103, (method)
- [type-name], page 103, (method)



- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 104, (method)
- [type-name], page 105, (method)
- [type-name], page 105, (method)
- [type-name], page 105, (method)
- [type-name], page 105, (method)
- [virtual-path], page 105, (generic function)
- [virtual-path], page 105, (method)
- [virtual-path], page 105, (method)
- [writer-definition], page 118, (structure)
- [writer-definition-foreignp], page 88, (function)
- [(setf writer-definition-foreignp)], page 88, (function)
- [writer-definition-function], page 88, (function)
- [(setf writer-definition-function)], page 88, (function)
- [writer-definition-p], page 88, (function)
- [writer-definition-reader], page 88, (function)
- [(setf writer-definition-reader)], page 88, (function)
- [writer-definition-symbol], page 88, (function)
- [(setf writer-definition-symbol)], page 88, (function)
- [writer-definition-update-expander], page 88, (function)
- [(setf writer-definition-update-expander)], page 88, (function)
- [writer-definitions], page 105, (generic function)
- [writer-definitions], page 105, (method)
- [writer-definitions], page 105, (method)
- [writer-method-definition], page 118, (structure)
- [writer-method-definition-foreignp], page 88, (function)
- [(setf writer-method-definition-foreignp)], page 88, (function)
- [writer-method-definition-method], page 88, (function)
- [(setf writer-method-definition-method)], page 88, (function)
- [writer-method-definition-p], page 88, (function)
- [writer-method-definition-symbol], page 88, (function)
- [(setf writer-method-definition-symbol)], page 88, (function)

### 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 PROG.

**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 PROG.

**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* [Function]

*CONTACT COPYRIGHT LICENSE INTRODUCTION CONCLUSION  
 TEXT-FILE INFO-FILE HYPERLINKS DECLT-NOTICE &aux SYSTEM  
 TEXT-NAME CURRENT-TIME-STRING CONTACT-NAMES  
 CONTACT-EMAILS*

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@address>.
- 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, unnumbered 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)

**@defnfx** *CATEGORY NAME LAMBDA-LIST &optional  
SPECIALIZERS QUALIFIERS* [Function]

Render @defnfx 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 @defnfx {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 @defnfx {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 @defnfx {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 @defunx 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)

<code>class-definition-methods</code> <i>INSTANCE</i>	[Function]
<code>(setf class-definition-methods) VALUE INSTANCE</code>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>class-definition-p</code> <i>OBJECT</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>class-definition-parents</code> <i>INSTANCE</i>	[Function]
<code>(setf class-definition-parents) VALUE INSTANCE</code>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>class-definition-slots</code> <i>INSTANCE</i>	[Function]
<code>(setf class-definition-slots) VALUE INSTANCE</code>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>class-definition-symbol</code> <i>INSTANCE</i>	[Function]
<code>(setf class-definition-symbol) VALUE INSTANCE</code>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>classoid-definition-children</code> <i>INSTANCE</i>	[Function]
<code>(setf classoid-definition-children) VALUE INSTANCE</code>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>classoid-definition-foreignp</code> <i>INSTANCE</i>	[Function]
<code>(setf classoid-definition-foreignp) VALUE INSTANCE</code>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>classoid-definition-methods</code> <i>INSTANCE</i>	[Function]
<code>(setf classoid-definition-methods) VALUE INSTANCE</code>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>classoid-definition-p</code> <i>OBJECT</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>classoid-definition-parents</code> <i>INSTANCE</i>	[Function]
<code>(setf classoid-definition-parents) VALUE INSTANCE</code>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	



<code>classoid-definition-slots</code> <i>INSTANCE</i>	[Function]
<code>(setf classoid-definition-slots)</code> <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>classoid-definition-symbol</code> <i>INSTANCE</i>	[Function]
<code>(setf classoid-definition-symbol)</code> <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>clindent</code> <i>SYMBOL INDENT</i>	[Function]
Set <i>SYMBOL</i> 's indentation to <i>INDENT</i> in (X)Emacs. This function sets <i>SYMBOL</i> 's common-lisp-indent-function property. If <i>INDENT</i> is a symbol, use its indentation definition. Otherwise, <i>INDENT</i> is considered as an indentation definition.	
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [meta.lisp], page 9, (file)	
<code>combination-definition-combination</code> <i>INSTANCE</i>	[Function]
<code>(setf combination-definition-combination)</code> <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>combination-definition-foreignp</code> <i>INSTANCE</i>	[Function]
<code>(setf combination-definition-foreignp)</code> <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>combination-definition-p</code> <i>OBJECT</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>combination-definition-symbol</code> <i>INSTANCE</i>	[Function]
<code>(setf combination-definition-symbol)</code> <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>combination-definition-users</code> <i>INSTANCE</i>	[Function]
<code>(setf combination-definition-users)</code> <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>compiler-macro-definition-foreignp</code> <i>INSTANCE</i>	[Function]
<code>(setf compiler-macro-definition-foreignp)</code> <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	

<p>compiler-macro-definition-function <i>INSTANCE</i> [Function]  (setf compiler-macro-definition-function) <i>VALUE INSTANCE</i> [Function]  <b>Package</b> [net.didierverna.declt], page 29,  <b>Source</b> [symbol.lisp], page 12, (file)</p>	
<p>compiler-macro-definition-p <i>OBJECT</i> [Function]  <b>Package</b> [net.didierverna.declt], page 29,  <b>Source</b> [symbol.lisp], page 12, (file)</p>	
<p>compiler-macro-definition-symbol <i>INSTANCE</i> [Function]  (setf compiler-macro-definition-symbol) <i>VALUE INSTANCE</i> [Function]  <b>Package</b> [net.didierverna.declt], page 29,  <b>Source</b> [symbol.lisp], page 12, (file)</p>	
<p>components <i>MODULE TYPE</i> [Function]  Return the list of all components of (sub)TYPE from ASDF MODULE.  <b>Package</b> [net.didierverna.declt], page 29,  <b>Source</b> [asdf.lisp], page 10, (file)</p>	
<p>condition-definition-children <i>INSTANCE</i> [Function]  (setf condition-definition-children) <i>VALUE INSTANCE</i> [Function]  <b>Package</b> [net.didierverna.declt], page 29,  <b>Source</b> [symbol.lisp], page 12, (file)</p>	
<p>condition-definition-foreignp <i>INSTANCE</i> [Function]  (setf condition-definition-foreignp) <i>VALUE INSTANCE</i> [Function]  <b>Package</b> [net.didierverna.declt], page 29,  <b>Source</b> [symbol.lisp], page 12, (file)</p>	
<p>condition-definition-methods <i>INSTANCE</i> [Function]  (setf condition-definition-methods) <i>VALUE INSTANCE</i> [Function]  <b>Package</b> [net.didierverna.declt], page 29,  <b>Source</b> [symbol.lisp], page 12, (file)</p>	
<p>condition-definition-p <i>OBJECT</i> [Function]  <b>Package</b> [net.didierverna.declt], page 29,  <b>Source</b> [symbol.lisp], page 12, (file)</p>	
<p>condition-definition-parents <i>INSTANCE</i> [Function]  (setf condition-definition-parents) <i>VALUE INSTANCE</i> [Function]  <b>Package</b> [net.didierverna.declt], page 29,  <b>Source</b> [symbol.lisp], page 12, (file)</p>	
<p>condition-definition-slots <i>INSTANCE</i> [Function]  (setf condition-definition-slots) <i>VALUE INSTANCE</i> [Function]  <b>Package</b> [net.didierverna.declt], page 29,  <b>Source</b> [symbol.lisp], page 12, (file)</p>	

condition-definition-symbol <i>INSTANCE</i>	[Function]
(setf condition-definition-symbol) <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
constant-definition-foreignp <i>INSTANCE</i>	[Function]
(setf constant-definition-foreignp) <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
constant-definition-p <i>OBJECT</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
constant-definition-symbol <i>INSTANCE</i>	[Function]
(setf constant-definition-symbol) <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
context-directory <i>CONTEXT</i>	[Function]
Return <i>CONTEXT</i> 's main system directory.	
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [doc.lisp], page 24, (file)	
context-external-definitions <i>INSTANCE</i>	[Function]
(setf context-external-definitions) <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [doc.lisp], page 24, (file)	
context-hyperlinksp <i>INSTANCE</i>	[Function]
(setf context-hyperlinksp) <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [doc.lisp], page 24, (file)	
context-internal-definitions <i>INSTANCE</i>	[Function]
(setf context-internal-definitions) <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [doc.lisp], page 24, (file)	
context-p <i>OBJECT</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [doc.lisp], page 24, (file)	
context-packages <i>INSTANCE</i>	[Function]
(setf context-packages) <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [doc.lisp], page 24, (file)	

context-systems	<i>INSTANCE</i>	[Function]
(setf context-systems)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[doc.lisp], page 24, (file)	
copy-accessor-definition	<i>INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
copy-accessor-method-definition	<i>INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
copy-class-definition	<i>INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
copy-classoid-definition	<i>INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
copy-combination-definition	<i>INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
copy-compiler-macro-definition	<i>INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
copy-condition-definition	<i>INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
copy-constant-definition	<i>INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
copy-context	<i>INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[doc.lisp], page 24, (file)	
copy-definition	<i>INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
copy-funcoid-definition	<i>INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	

<code>copy-function-definition</code> <i>INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>copy-generic-accessor-definition</code> <i>INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>copy-generic-definition</code> <i>INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>copy-generic-writer-definition</code> <i>INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>copy-long-combination-definition</code> <i>INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>copy-macro-definition</code> <i>INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>copy-method-definition</code> <i>INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>copy-node</code> <i>INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [texi.lisp], page 10, (file)	
<code>copy-setf-expander-definition</code> <i>INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>copy-short-combination-definition</code> <i>INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>copy-slot-definition</code> <i>INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>copy-special-definition</code> <i>INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [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	<i>INSTANCE</i>	[Function]
(setf function-definition-function)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
function-definition-p	<i>OBJECT</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
function-definition-symbol	<i>INSTANCE</i>	[Function]
(setf function-definition-symbol)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
function-definition-update-expander	<i>INSTANCE</i>	[Function]
(setf function-definition-update-expander)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
generate-quickutils ()		[Function]
	Generate the offline quickutil file.	
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[meta.lisp], page 9, (file)	
generic-accessor-definition-access-expander	<i>INSTANCE</i>	[Function]
(setf generic-accessor-definition-access-expander)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
generic-accessor-definition-combination	<i>INSTANCE</i>	[Function]
(setf generic-accessor-definition-combination)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
generic-accessor-definition-foreignp	<i>INSTANCE</i>	[Function]
(setf generic-accessor-definition-foreignp)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
generic-accessor-definition-function	<i>INSTANCE</i>	[Function]
(setf generic-accessor-definition-function)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
generic-accessor-definition-methods	<i>INSTANCE</i>	[Function]
(setf generic-accessor-definition-methods)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	

generic-accessor-definition-p	<i>OBJECT</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
generic-accessor-definition-symbol	<i>INSTANCE</i>	[Function]
(setf generic-accessor-definition-symbol)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
generic-accessor-definition-update-expander	<i>INSTANCE</i>	[Function]
(setf generic-accessor-definition-update-expander)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
generic-accessor-definition-writer	<i>INSTANCE</i>	[Function]
(setf generic-accessor-definition-writer)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
generic-definition-combination	<i>INSTANCE</i>	[Function]
(setf generic-definition-combination)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
generic-definition-foreignp	<i>INSTANCE</i>	[Function]
(setf generic-definition-foreignp)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
generic-definition-function	<i>INSTANCE</i>	[Function]
(setf generic-definition-function)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
generic-definition-methods	<i>INSTANCE</i>	[Function]
(setf generic-definition-methods)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
generic-definition-p	<i>OBJECT</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
generic-definition-symbol	<i>INSTANCE</i>	[Function]
(setf generic-definition-symbol)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	

<code>generic-definition-update-expander</code> <i>INSTANCE</i>	[Function]
<code>(setf generic-definition-update-expander)</code> <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>generic-writer-definition-combination</code> <i>INSTANCE</i>	[Function]
<code>(setf generic-writer-definition-combination)</code> <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>generic-writer-definition-foreignp</code> <i>INSTANCE</i>	[Function]
<code>(setf generic-writer-definition-foreignp)</code> <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>generic-writer-definition-function</code> <i>INSTANCE</i>	[Function]
<code>(setf generic-writer-definition-function)</code> <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>generic-writer-definition-methods</code> <i>INSTANCE</i>	[Function]
<code>(setf generic-writer-definition-methods)</code> <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>generic-writer-definition-p</code> <i>OBJECT</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>generic-writer-definition-reader</code> <i>INSTANCE</i>	[Function]
<code>(setf generic-writer-definition-reader)</code> <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>generic-writer-definition-symbol</code> <i>INSTANCE</i>	[Function]
<code>(setf generic-writer-definition-symbol)</code> <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>generic-writer-definition-update-expander</code> <i>INSTANCE</i>	[Function]
<code>(setf generic-writer-definition-update-expander)</code> <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>i-reader</code> <i>STREAM SUBCHAR ARG</i>	[Function]
Read an argument list for the <code>DEFINDENT</code> macro.	
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [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* [Function]  
*INSTANCE*
- 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* [Function]  
**FOREIGNP**) (*FUNCTION FUNCTION*) (*UPDATE-EXPANDER*  
**UPDATE-EXPANDER**) (*WRITER WRITER*) (*ACCESS-EXPANDER*  
**ACCESS-EXPANDER**)  
**Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)
- make-accessor-method-definition **&key** (*SYMBOL SYMBOL*) [Function]  
 (*FOREIGNP FOREIGNP*) (*METHOD METHOD*) (*WRITER WRITER*)  
**Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)
- make-class-definition **&key** (*SYMBOL SYMBOL*) (*FOREIGNP* [Function]  
**FOREIGNP**) (*PARENTS PARENTS*) (*CHILDREN CHILDREN*)  
 (*METHODS METHODS*) (*SLOTS SLOTS*)  
**Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)
- make-classoid-definition **&key** (*SYMBOL SYMBOL*) (*FOREIGNP* [Function]  
**FOREIGNP**) (*PARENTS PARENTS*) (*CHILDREN CHILDREN*)  
 (*METHODS METHODS*) (*SLOTS SLOTS*)  
**Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)
- make-combination-definition **&key** (*SYMBOL SYMBOL*) [Function]  
 (*FOREIGNP FOREIGNP*) (*COMBINATION COMBINATION*) (*USERS*  
**USERS**)  
**Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)

- `make-compiler-macro-definition` **&key** (*SYMBOL SYMBOL*) [Function]  
(*FOREIGNP FOREIGNP*) (*FUNCTION FUNCTION*)
- Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)
- `make-condition-definition` **&key** (*SYMBOL SYMBOL*) (*FOREIGNP* [Function]  
**FOREIGNP**) (*PARENTS PARENTS*) (*CHILDREN CHILDREN*)  
(*METHODS METHODS*) (*SLOTS SLOTS*)
- Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)
- `make-constant-definition` **&key** (*SYMBOL SYMBOL*) (*FOREIGNP* [Function]  
**FOREIGNP**)
- Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)
- `make-context` **&key** (*SYSTEMS SYSTEMS*) (*PACKAGES PACKAGES*) [Function]  
(*EXTERNAL-DEFINITIONS EXTERNAL-DEFINITIONS*)  
(*INTERNAL-DEFINITIONS INTERNAL-DEFINITIONS*)  
(*HYPERLINKSP HYPERLINKSP*)
- Package** [net.didierverna.declt], page 29,  
**Source** [doc.lisp], page 24, (file)
- `make-definition` **&key** (*SYMBOL SYMBOL*) (*FOREIGNP* [Function]  
**FOREIGNP**)
- 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* [Function]  
**FOREIGNP**) (*FUNCTION FUNCTION*)
- Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)
- `make-function-definition` **&key** (*SYMBOL SYMBOL*) (*FOREIGNP* [Function]  
**FOREIGNP**) (*FUNCTION FUNCTION*) (*UPDATE-EXPANDER*  
**UPDATE-EXPANDER**)
- Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)

- `make-generic-accessor-definition` **&key** (*SYMBOL SYMBOL*) [Function]  
 (*FOREIGNP FOREIGNP*) (*FUNCTION FUNCTION*)  
 (*UPDATE-EXPANDER UPDATE-EXPANDER*) (*COMBINATION*  
**COMBINATION**) (*METHODS METHODS*) (*WRITER WRITER*)  
 (*ACCESS-EXPANDER ACCESS-EXPANDER*)
- Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)
- `make-generic-definition` **&key** (*SYMBOL SYMBOL*) (*FOREIGNP* [Function]  
**FOREIGNP**) (*FUNCTION FUNCTION*) (*UPDATE-EXPANDER*  
**UPDATE-EXPANDER**) (*COMBINATION COMBINATION*) (*METHODS*  
**METHODS**)
- Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)
- `make-generic-writer-definition` **&key** (*SYMBOL SYMBOL*) [Function]  
 (*FOREIGNP FOREIGNP*) (*FUNCTION FUNCTION*)  
 (*UPDATE-EXPANDER UPDATE-EXPANDER*) (*COMBINATION*  
**COMBINATION**) (*METHODS METHODS*) (*READER READER*)
- Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)
- `make-long-combination-definition` **&key** (*SYMBOL SYMBOL*) [Function]  
 (*FOREIGNP FOREIGNP*) (*COMBINATION COMBINATION*) (*USERS*  
**USERS**)
- Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)
- `make-macro-definition` **&key** (*SYMBOL SYMBOL*) (*FOREIGNP* [Function]  
**FOREIGNP**) (*FUNCTION FUNCTION*) (*ACCESS-EXPANDER*  
**ACCESS-EXPANDER**) (*UPDATE-EXPANDER UPDATE-EXPANDER*)
- Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)
- `make-method-definition` **&key** (*SYMBOL SYMBOL*) (*FOREIGNP* [Function]  
**FOREIGNP**) (*METHOD METHOD*)
- Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)
- `make-node` **&key** (*NAME NAME*) (*SYNOPSIS SYNOPSIS*) [Function]  
 (*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**)
- Package** [net.didierverna.declt], page 29,  
**Source** [texi.lisp], page 10, (file)

- `make-setf-expander-definition &key (SYMBOL SYMBOL)` [Function]  
 (*FOREIGNP FOREIGNP*) (*ACCESS ACCESS*) (*UPDATE UPDATE*)
- Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)
- `make-short-combination-definition &key (SYMBOL SYMBOL)` [Function]  
 (*FOREIGNP FOREIGNP*) (*COMBINATION COMBINATION*) (*USERS*  
*USERS*) (*OPERATOR OPERATOR*)
- Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)
- `make-slot-definition &key (SYMBOL SYMBOL) (FOREIGNP` [Function]  
**FOREIGNP**) (*SLOT SLOT*) (*READERS READERS*) (*WRITERS*  
*WRITERS*)
- 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` [Function]  
**FOREIGNP**)
- Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)
- `make-structure-definition &key (SYMBOL SYMBOL) (FOREIGNP` [Function]  
**FOREIGNP**) (*PARENTS PARENTS*) (*CHILDREN CHILDREN*)  
 (*METHODS METHODS*) (*SLOTS SLOTS*)
- Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)
- `make-symbol-macro-definition &key (SYMBOL SYMBOL)` [Function]  
 (*FOREIGNP FOREIGNP*)
- Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)
- `make-type-definition &key (SYMBOL SYMBOL) (FOREIGNP` [Function]  
**FOREIGNP**)
- Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)
- `make-writer-definition &key (SYMBOL SYMBOL) (FOREIGNP` [Function]  
**FOREIGNP**) (*FUNCTION FUNCTION*) (*UPDATE-EXPANDER*  
*UPDATE-EXPANDER*) (*READER READER*)
- Package** [net.didierverna.declt], page 29,  
**Source** [symbol.lisp], page 12, (file)



make-writer-method-definition <b>&amp;key</b> ( <i>SYMBOL SYMBOL</i> ) ( <i>FOREIGNP FOREIGNP</i> ) ( <i>METHOD METHOD</i> )	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
mapcan-definitions-pool <i>FUNCTION POOL</i>	[Function]
Like MAPCAN, but work on a definitions POOL.	
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
method-definition-foreignp <i>INSTANCE</i>	[Function]
(setf method-definition-foreignp) <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
method-definition-method <i>INSTANCE</i>	[Function]
(setf method-definition-method) <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
method-definition-p <i>OBJECT</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
method-definition-symbol <i>INSTANCE</i>	[Function]
(setf method-definition-symbol) <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
method-name <i>METHOD &amp;aux NAME</i>	[Function]
Return METHOD's name.	
Return a second value of T if METHOD is a writer method.	
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
module-components <i>MODULE</i>	[Function]
Return the list of all module components from ASDF MODULE.	
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [asdf.lisp], page 10, (file)	
module-node <i>MODULE CONTEXT</i>	[Function]
Create and return a MODULE node in CONTEXT.	
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [asdf.lisp], page 26, (file)	
node-after-menu-contents <i>INSTANCE</i>	[Function]
(setf node-after-menu-contents) <i>VALUE INSTANCE</i>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [texi.lisp], page 10, (file)	

node-before-menu-contents	<i>INSTANCE</i>	[Function]
(setf node-before-menu-contents)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[texi.lisp], page 10, (file)	
node-children	<i>INSTANCE</i>	[Function]
(setf node-children)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[texi.lisp], page 10, (file)	
node-name	<i>INSTANCE</i>	[Function]
(setf node-name)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[texi.lisp], page 10, (file)	
node-next	<i>INSTANCE</i>	[Function]
(setf node-next)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[texi.lisp], page 10, (file)	
node-p	<i>OBJECT</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[texi.lisp], page 10, (file)	
node-previous	<i>INSTANCE</i>	[Function]
(setf node-previous)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[texi.lisp], page 10, (file)	
node-section-name	<i>INSTANCE</i>	[Function]
(setf node-section-name)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[texi.lisp], page 10, (file)	
node-section-type	<i>INSTANCE</i>	[Function]
(setf node-section-type)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[texi.lisp], page 10, (file)	
node-synopsis	<i>INSTANCE</i>	[Function]
(setf node-synopsis)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[texi.lisp], page 10, (file)	
node-up	<i>INSTANCE</i>	[Function]
(setf node-up)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[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  
INTERNAL-SYMBOLS* [Function]  
 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 TEXT-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)

<code>sbcl-has-setf-inverse-meta-info ()</code>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>setf-expander-definition-access <i>INSTANCE</i></code>	[Function]
<code>(setf setf-expander-definition-access) <i>VALUE INSTANCE</i></code>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>setf-expander-definition-foreignp <i>INSTANCE</i></code>	[Function]
<code>(setf setf-expander-definition-foreignp) <i>VALUE INSTANCE</i></code>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>setf-expander-definition-p <i>OBJECT</i></code>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>setf-expander-definition-symbol <i>INSTANCE</i></code>	[Function]
<code>(setf setf-expander-definition-symbol) <i>VALUE INSTANCE</i></code>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>setf-expander-definition-update <i>INSTANCE</i></code>	[Function]
<code>(setf setf-expander-definition-update) <i>VALUE INSTANCE</i></code>	[Function]
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>setf-expander-p <i>SYMBOL</i></code>	[Function]
Return whether <i>SYMBOL</i> defines a setf-expander.	
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>short-combination-definition-combination <i>INSTANCE</i></code>	[Function]
<code>(setf short-combination-definition-combination) <i>VALUE</i></code>	[Function]
<i>INSTANCE</i>	
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>short-combination-definition-foreignp <i>INSTANCE</i></code>	[Function]
<code>(setf short-combination-definition-foreignp) <i>VALUE</i></code>	[Function]
<i>INSTANCE</i>	
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>short-combination-definition-operator <i>INSTANCE</i></code>	[Function]
<code>(setf short-combination-definition-operator) <i>VALUE</i></code>	[Function]
<i>INSTANCE</i>	
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [symbol.lisp], page 12, (file)	

short-combination-definition-p	<i>OBJECT</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
short-combination-definition-symbol	<i>INSTANCE</i>	[Function]
(setf short-combination-definition-symbol)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
short-combination-definition-users	<i>INSTANCE</i>	[Function]
(setf short-combination-definition-users)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
slot-definition-foreignp	<i>INSTANCE</i>	[Function]
(setf slot-definition-foreignp)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
slot-definition-p	<i>OBJECT</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
slot-definition-readers	<i>INSTANCE</i>	[Function]
(setf slot-definition-readers)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
slot-definition-slot	<i>INSTANCE</i>	[Function]
(setf slot-definition-slot)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
slot-definition-symbol	<i>INSTANCE</i>	[Function]
(setf slot-definition-symbol)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
slot-definition-writers	<i>INSTANCE</i>	[Function]
(setf slot-definition-writers)	<i>VALUE INSTANCE</i>	[Function]
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	
slot-property	<i>SLOT PROPERTY</i>	[Function]
Return SLOT definition's PROPERTY value.		
<b>Package</b>	[net.didierverna.declt], page 29,	
<b>Source</b>	[symbol.lisp], page 12, (file)	



special-definition-foreignp	<i>INSTANCE</i>	[Function]
(setf special-definition-foreignp)	<i>VALUE INSTANCE</i>	[Function]
Package	[net.didierverna.declt], page 29,	
Source	[symbol.lisp], page 12, (file)	
special-definition-p	<i>OBJECT</i>	[Function]
Package	[net.didierverna.declt], page 29,	
Source	[symbol.lisp], page 12, (file)	
special-definition-symbol	<i>INSTANCE</i>	[Function]
(setf special-definition-symbol)	<i>VALUE INSTANCE</i>	[Function]
Package	[net.didierverna.declt], page 29,	
Source	[symbol.lisp], page 12, (file)	
specializers	<i>METHOD</i>	[Function]
	Return METHOD's specializers.	
Package	[net.didierverna.declt], page 29,	
Source	[symbol.lisp], page 12, (file)	
structure-definition-children	<i>INSTANCE</i>	[Function]
(setf structure-definition-children)	<i>VALUE INSTANCE</i>	[Function]
Package	[net.didierverna.declt], page 29,	
Source	[symbol.lisp], page 12, (file)	
structure-definition-foreignp	<i>INSTANCE</i>	[Function]
(setf structure-definition-foreignp)	<i>VALUE INSTANCE</i>	[Function]
Package	[net.didierverna.declt], page 29,	
Source	[symbol.lisp], page 12, (file)	
structure-definition-methods	<i>INSTANCE</i>	[Function]
(setf structure-definition-methods)	<i>VALUE INSTANCE</i>	[Function]
Package	[net.didierverna.declt], page 29,	
Source	[symbol.lisp], page 12, (file)	
structure-definition-p	<i>OBJECT</i>	[Function]
Package	[net.didierverna.declt], page 29,	
Source	[symbol.lisp], page 12, (file)	
structure-definition-parents	<i>INSTANCE</i>	[Function]
(setf structure-definition-parents)	<i>VALUE INSTANCE</i>	[Function]
Package	[net.didierverna.declt], page 29,	
Source	[symbol.lisp], page 12, (file)	
structure-definition-slots	<i>INSTANCE</i>	[Function]
(setf structure-definition-slots)	<i>VALUE INSTANCE</i>	[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	<i>INSTANCE</i>	[Function]
(setf writer-definition-foreignp)	<i>VALUE INSTANCE</i>	[Function]
Package	[net.didierverna.declt], page 29,	
Source	[symbol.lisp], page 12, (file)	
writer-definition-function	<i>INSTANCE</i>	[Function]
(setf writer-definition-function)	<i>VALUE INSTANCE</i>	[Function]
Package	[net.didierverna.declt], page 29,	
Source	[symbol.lisp], page 12, (file)	
writer-definition-p	<i>OBJECT</i>	[Function]
Package	[net.didierverna.declt], page 29,	
Source	[symbol.lisp], page 12, (file)	
writer-definition-reader	<i>INSTANCE</i>	[Function]
(setf writer-definition-reader)	<i>VALUE INSTANCE</i>	[Function]
Package	[net.didierverna.declt], page 29,	
Source	[symbol.lisp], page 12, (file)	
writer-definition-symbol	<i>INSTANCE</i>	[Function]
(setf writer-definition-symbol)	<i>VALUE INSTANCE</i>	[Function]
Package	[net.didierverna.declt], page 29,	
Source	[symbol.lisp], page 12, (file)	
writer-definition-update-expander	<i>INSTANCE</i>	[Function]
(setf writer-definition-update-expander)	<i>VALUE INSTANCE</i>	[Function]
Package	[net.didierverna.declt], page 29,	
Source	[symbol.lisp], page 12, (file)	
writer-method-definition-foreignp	<i>INSTANCE</i>	[Function]
(setf writer-method-definition-foreignp)	<i>VALUE INSTANCE</i>	[Function]
Package	[net.didierverna.declt], page 29,	
Source	[symbol.lisp], page 12, (file)	
writer-method-definition-method	<i>INSTANCE</i>	[Function]
(setf writer-method-definition-method)	<i>VALUE INSTANCE</i>	[Function]
Package	[net.didierverna.declt], page 29,	
Source	[symbol.lisp], page 12, (file)	
writer-method-definition-p	<i>OBJECT</i>	[Function]
Package	[net.didierverna.declt], page 29,	
Source	[symbol.lisp], page 12, (file)	
writer-method-definition-symbol	<i>INSTANCE</i>	[Function]
(setf writer-method-definition-symbol)	<i>VALUE INSTANCE</i>	[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 *CATEGORY*s.
- 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* [Generic Function]  
*DOCUMENT-WRITERS GENERIC-SOURCE &allow-other-keys*

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* [Method]  
*PATHNAME*

Render lisp FILE's documentation in CONTEXT.

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

**document** (*COMPONENT component*) *CONTEXT &key* [Method]  
*&aux RELATIVE-TO*

Render COMPONENT's documentation in CONTEXT.

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

**document** (*COMPONENT component*) *CONTEXT &key* [Method]  
*around*

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*) [Method]  
*CONTEXT &key*

Render STRUCTURE's documentation in CONTEXT.

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

- document (*CONDITION* condition-definition) [Method]  
**CONTEXT &key**  
 Render *CONDITION*'s documentation in *CONTEXT*.  
**Source** [symbol.lisp], page 24, (file)
- document (*COMBINATION* long-combination-definition) [Method]  
**CONTEXT &key**  
 Render long method *COMBINATION*'s documentation in *CONTEXT*.  
**Source** [symbol.lisp], page 24, (file)
- document (*COMBINATION* short-combination-definition) [Method]  
**CONTEXT &key**  
 Render short method *COMBINATION*'s documentation in *CONTEXT*.  
**Source** [symbol.lisp], page 24, (file)
- document (*EXPANDER* setf-expander-definition) [Method]  
**CONTEXT &key**  
 Render setf *EXPANDER*'s documentation in *CONTEXT*.  
**Source** [symbol.lisp], page 24, (file)
- document (*ACCESSOR* generic-accessor-definition) [Method]  
**CONTEXT &key &aux ACCESS-EXPANDER  
 UPDATE-EXPANDER WRITER**  
 Render generic *ACCESSOR*'s documentation in *CONTEXT*.  
**Source** [symbol.lisp], page 24, (file)
- document (*WRITER* generic-writer-definition) [Method]  
**CONTEXT &key ADDITIONAL-METHODS**  
 Render generic *WRITER*'s documentation in *CONTEXT*.  
**Source** [symbol.lisp], page 24, (file)
- document (*GENERIC* generic-definition) [Method]  
**CONTEXT &key**  
 Render *GENERIC*'s documentation in *CONTEXT*.  
**Source** [symbol.lisp], page 24, (file)
- document (*METHOD* accessor-method-definition) [Method]  
**CONTEXT &key DOCUMENT-WRITERS  
 GENERIC-SOURCE**  
 Render accessor *METHOD*'s documentation in *CONTEXT*.  
**Source** [symbol.lisp], page 24, (file)
- document (*METHOD* method-definition) [Method]  
**CONTEXT &key GENERIC-SOURCE**  
 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* [Method]  
**&key &aux** *ACCESS-EXPANDER UPDATE-EXPANDER*  
*WRITER MERGE-EXPANDER MERGE-WRITER*  
*MERGE-SETTERS*

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* [Method]  
*ACCESS-EXPANDER UPDATE-EXPANDER MERGE*  
 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 CATEGORIES
- 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.

<code>lambda-list</code> ( <i>FUNCOID</i> funcoid-definition)	[Method]
Return FUNCOID's lambda-list.	
<code>lambda-list</code> ( <i>EXPANDER</i> setf-expander-definition) &aux <i>UPDATE</i>	[Method]
Return setf EXPANDER's lambda-list.	
<code>lambda-list</code> ( <i>METHOD</i> method-definition)	[Method]
Return METHOD's lambda-list.	
<code>lambda-list</code> ( <i>TYPE</i> type-definition)	[Method]
Return TYPE's lambda-list.	
<b>name</b> <i>OBJECT</i>	[Generic Function]
Return OBJECT's name as a string.	
<b>Package</b> [net.didierverna.declt], page 29,	
<b>Source</b> [texi.lisp], page 10, (file)	
<b>Methods</b>	
<code>name</code> ( <i>SOURCE-FILE</i> source-file) &aux <i>NAME</i> <i>EXTENSION</i>	[Method]
Return SOURCE-FILE's name, possibly adding its extension.	
<b>Source</b> [asdf.lisp], page 23, (file)	
<code>name</code> ( <i>COMPONENT</i> component)	[Method]
Return COMPONENT's name.	
<b>Source</b> [asdf.lisp], page 23, (file)	
<code>name</code> ( <i>PACKAGE</i> package)	[Method]
Return PACKAGE's name.	
<b>Source</b> [package.lisp], page 23, (file)	
<code>name</code> ( <i>EXPANDER</i> setf-expander-definition)	[Method]
Return setf EXPANDER's name, that is (setf <name>).	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>name</code> ( <i>GENERIC-WRITER</i> generic-writer-definition)	[Method]
Return GENERIC-WRITER's name, that is (setf <name>).	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>name</code> ( <i>WRITER-METHOD</i> writer-method-definition)	[Method]
Return WRITER-METHOD's name, that is (setf <name>).	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>name</code> ( <i>WRITER</i> writer-definition)	[Method]
Return WRITER's name, that is (setf <name>).	
<b>Source</b> [symbol.lisp], page 12, (file)	
<code>name</code> ( <i>DEFINITION</i> definition)	[Method]
Return DEFINITION's symbol name.	
<b>Source</b> [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]  
Default 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 RELATIVE-TO* [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* [Generic Function]  
*RELATIVE-TO*

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* [Method]  
*list*) *SYSTEM RELATIVE-TO*

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)

<p><code>type-name</code> (<i>PACKAGE</i> package)  Return "package".  <b>Source</b> [package.lisp], page 23, (file)</p>	[Method]
<p><code>type-name</code> (<i>TYPE</i> type-definition)  Return "type"  <b>Source</b> [symbol.lisp], page 12, (file)</p>	[Method]
<p><code>type-name</code> (<i>CLASS</i> class-definition)  Return "class"  <b>Source</b> [symbol.lisp], page 12, (file)</p>	[Method]
<p><code>type-name</code> (<i>STRUCTURE</i> structure-definition)  Return "structure"  <b>Source</b> [symbol.lisp], page 12, (file)</p>	[Method]
<p><code>type-name</code> (<i>CONDITION</i> condition-definition)  Return "condition"  <b>Source</b> [symbol.lisp], page 12, (file)</p>	[Method]
<p><code>type-name</code> (<i>COMBINATION</i>  long-combination-definition)  Return "long method combination".  <b>Source</b> [symbol.lisp], page 12, (file)</p>	[Method]
<p><code>type-name</code> (<i>COMBINATION</i>  short-combination-definition)  Return "short method combination".  <b>Source</b> [symbol.lisp], page 12, (file)</p>	[Method]
<p><code>type-name</code> (<i>EXPANDER</i> setf-expander-definition)  Return "setf expander"  <b>Source</b> [symbol.lisp], page 12, (file)</p>	[Method]
<p><code>type-name</code> (<i>METHOD</i> method-definition)  Return "method"  <b>Source</b> [symbol.lisp], page 12, (file)</p>	[Method]
<p><code>type-name</code> (<i>GENERIC</i> generic-definition)  Return "generic function"  <b>Source</b> [symbol.lisp], page 12, (file)</p>	[Method]
<p><code>type-name</code> (<i>FUNCTION</i> function-definition)  Return "function"  <b>Source</b> [symbol.lisp], page 12, (file)</p>	[Method]
<p><code>type-name</code> (<i>COMPILER-MACRO</i>  compiler-macro-definition)  Return "compiler macro"  <b>Source</b> [symbol.lisp], page 12, (file)</p>	[Method]

- `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 COMPONENT'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**  
 [funccoid-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]
<b>Readers</b>	[definition-symbol], page 67, (function)	
<b>Writers</b>	[(setf definition-symbol)], page 67, (function)	
foreignp		[Slot]
<b>Readers</b>	[definition-foreignp], page 66, (function)	
<b>Writers</b>	[(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]
<b>Readers</b>	[funcoid-definition-function], page 68, (function)	
<b>Writers</b>	[(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]
<b>Readers</b>	[method-definition-method], page 77, (function)	
<b>Writers</b>	[(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]
<b>Readers</b>	[node-name], page 78, (function)	
<b>Writers</b>	[(setf node-name)], page 78, (function)	
synopsis		[Slot]
<b>Readers</b>	[node-synopsis], page 78, (function)	
<b>Writers</b>	[(setf node-synopsis)], page 78, (function)	
section-type		[Slot]
<b>Initform</b>	:numbered	
<b>Readers</b>	[node-section-type], page 78, (function)	
<b>Writers</b>	[(setf node-section-type)], page 78, (function)	
section-name		[Slot]
<b>Readers</b>	[node-section-name], page 78, (function)	
<b>Writers</b>	[(setf node-section-name)], page 78, (function)	
next		[Slot]
<b>Readers</b>	[node-next], page 78, (function)	
<b>Writers</b>	[(setf node-next)], page 78, (function)	

<code>previous</code>		[Slot]
<b>Readers</b>	[ <code>node-previous</code> ], page 78, (function)	
<b>Writers</b>	[ <code>(setf node-previous)</code> ], page 78, (function)	
<code>up</code>		[Slot]
<b>Readers</b>	[ <code>node-up</code> ], page 78, (function)	
<b>Writers</b>	[ <code>(setf node-up)</code> ], page 78, (function)	
<code>children</code>		[Slot]
<b>Readers</b>	[ <code>node-children</code> ], page 78, (function)	
<b>Writers</b>	[ <code>(setf node-children)</code> ], page 78, (function)	
<code>before-menu-contents</code>		[Slot]
<b>Readers</b>	[ <code>node-before-menu-contents</code> ], page 78, (function)	
<b>Writers</b>	[ <code>(setf node-before-menu-contents)</code> ], page 78, (function)	
<code>after-menu-contents</code>		[Slot]
<b>Readers</b>	[ <code>node-after-menu-contents</code> ], page 77, (function)	
<b>Writers</b>	[ <code>(setf node-after-menu-contents)</code> ], page 77, (function)	
<code>setf-expander-definition ()</code>		[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.		
<b>Package</b>	[ <code>net.didierverna.declt</code> ], page 29,	
<b>Source</b>	[ <code>symbol.lisp</code> ], page 12, (file)	
<b>Direct superclasses</b>		
	[ <code>definition</code> ], page 109, (structure)	
<b>Direct methods</b>		
	<ul style="list-style-type: none"> <li>• [<code>document</code>], page 94, (method)</li> <li>• [<code>index</code>], page 98, (method)</li> <li>• [<code>headline-function</code>], page 96, (method)</li> <li>• [<code>type-name</code>], page 104, (method)</li> <li>• [<code>docstring</code>], page 92, (method)</li> <li>• [<code>source</code>], page 102, (method)</li> <li>• [<code>name</code>], page 99, (method)</li> <li>• [<code>lambda-list</code>], page 99, (method)</li> </ul>	
<b>Direct slots</b>		
<code>access</code>		[Slot]
<b>Readers</b>	[ <code>setf-expander-definition-access</code> ], page 83, (function)	
<b>Writers</b>	[ <code>(setf setf-expander-definition-access)</code> ], page 83, (function)	

<code>update</code>		[Slot]
<b>Readers</b>	[ <code>setf-expander-definition-update</code> ], page 83, (function)	
<b>Writers</b>	[( <code>setf setf-expander-definition-update</code> )], page 83, (function)	
<code>short-combination-definition ()</code>		[Structure]
Structure for short method combination definitions.		
This structure holds the operator definition.		
<b>Package</b>	[ <code>net.didierverna.declt</code> ], page 29,	
<b>Source</b>	[ <code>symbol.lisp</code> ], page 12, (file)	
<b>Direct superclasses</b>	[ <code>combination-definition</code> ], page 107, (structure)	
<b>Direct methods</b>		
	<ul style="list-style-type: none"> <li>• [<code>document</code>], page 94, (method)</li> <li>• [<code>index</code>], page 97, (method)</li> <li>• [<code>type-name</code>], page 104, (method)</li> </ul>	
<b>Direct slots</b>		
<code>operator</code>		[Slot]
<b>Readers</b>	[ <code>short-combination-definition-operator</code> ], page 83, (function)	
<b>Writers</b>	[( <code>setf short-combination-definition-operator</code> )], page 83, (function)	
<code>slot-definition ()</code>		[Structure]
Structure for slot definitions.		
This structure holds the slot object and the readers and writers definitions.		
<b>Package</b>	[ <code>net.didierverna.declt</code> ], page 29,	
<b>Source</b>	[ <code>symbol.lisp</code> ], page 12, (file)	
<b>Direct superclasses</b>	[ <code>definition</code> ], page 109, (structure)	
<b>Direct methods</b>		
	<ul style="list-style-type: none"> <li>• [<code>index</code>], page 98, (method)</li> <li>• [<code>docstring</code>], page 92, (method)</li> </ul>	
<b>Direct slots</b>		
<code>slot</code>		[Slot]
<b>Readers</b>	[ <code>slot-definition-slot</code> ], page 84, (function)	
<b>Writers</b>	[( <code>setf slot-definition-slot</code> )], page 84, (function)	
<code>readers</code>		[Slot]
<b>Readers</b>	[ <code>slot-definition-readers</code> ], page 84, (function)	
<b>Writers</b>	[( <code>setf slot-definition-readers</code> )], page 84, (function)	
<code>writers</code>		[Slot]
<b>Readers</b>	[ <code>slot-definition-writers</code> ], page 84, (function)	
<b>Writers</b>	[( <code>setf slot-definition-writers</code> )], 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)



## A.2 Functions

%

%version..... 55

(

(setf  
 accessor-definition-access-expander) ..... 56  
 (setf accessor-definition-foreignp)..... 56  
 (setf accessor-definition-function)..... 56  
 (setf accessor-definition-symbol)..... 56  
 (setf  
 accessor-definition-update-expander) ..... 57  
 (setf accessor-definition-writer)..... 57  
 (setf  
 accessor-method-definition-foreignp) ..... 57  
 (setf accessor-method-definition-method).... 57  
 (setf accessor-method-definition-symbol).... 57  
 (setf accessor-method-definition-writer).... 57  
 (setf class-definition-children) ..... 59  
 (setf class-definition-foreignp) ..... 59  
 (setf class-definition-methods) ..... 60  
 (setf class-definition-parents) ..... 60  
 (setf class-definition-slots)..... 60  
 (setf class-definition-symbol) ..... 60  
 (setf classoid-definition-children) ..... 60  
 (setf classoid-definition-foreignp)..... 60  
 (setf classoid-definition-methods)..... 60  
 (setf classoid-definition-parents)..... 60  
 (setf classoid-definition-slots) ..... 61  
 (setf classoid-definition-symbol)..... 61  
 (setf combination-definition-combination)... 61  
 (setf combination-definition-foreignp) ..... 61  
 (setf combination-definition-symbol)..... 61  
 (setf combination-definition-users)..... 61  
 (setf compiler-macro-definition-foreignp)... 61  
 (setf compiler-macro-definition-function)... 62  
 (setf compiler-macro-definition-symbol).... 62  
 (setf condition-definition-children)..... 62  
 (setf condition-definition-foreignp)..... 62  
 (setf condition-definition-methods)..... 62  
 (setf condition-definition-parents)..... 62  
 (setf condition-definition-slots)..... 62  
 (setf condition-definition-symbol)..... 63  
 (setf constant-definition-foreignp)..... 63  
 (setf constant-definition-symbol)..... 63  
 (setf context-external-definitions)..... 63  
 (setf context-hyperlinksp) ..... 63  
 (setf context-internal-definitions)..... 63  
 (setf context-packages) ..... 63  
 (setf context-systems)..... 64  
 (setf definition-foreignp) ..... 66  
 (setf definition-symbol) ..... 67  
 (setf funcoid-definition-foreignp)..... 68  
 (setf funcoid-definition-function)..... 68  
 (setf funcoid-definition-symbol) ..... 68  
 (setf function-definition-foreignp)..... 68  
 (setf function-definition-function)..... 69  
 (setf function-definition-symbol)..... 69  
 (setf  
 function-definition-update-expander) ..... 69

(setf generic-accessor-definition-  
 access-expander) ..... 69  
 (setf  
 generic-accessor-definition-combination).. 69  
 (setf  
 generic-accessor-definition-foreignp) ..... 69  
 (setf  
 generic-accessor-definition-function) ..... 69  
 (setf  
 generic-accessor-definition-methods) ..... 69  
 (setf generic-accessor-definition-symbol)... 70  
 (setf generic-accessor-definition-  
 update-expander) ..... 70  
 (setf generic-accessor-definition-writer)... 70  
 (setf generic-definition-combination) ..... 70  
 (setf generic-definition-foreignp) ..... 70  
 (setf generic-definition-function) ..... 70  
 (setf generic-definition-methods)..... 70  
 (setf generic-definition-symbol) ..... 70  
 (setf generic-definition-update-expander)... 71  
 (setf  
 generic-writer-definition-combination) .... 71  
 (setf generic-writer-definition-foreignp)... 71  
 (setf generic-writer-definition-function)... 71  
 (setf generic-writer-definition-methods).... 71  
 (setf generic-writer-definition-reader) ..... 71  
 (setf generic-writer-definition-symbol) ..... 71  
 (setf generic-writer-definition-  
 update-expander) ..... 71  
 (setf  
 long-combination-definition-combination).. 72  
 (setf  
 long-combination-definition-foreignp) ..... 72  
 (setf long-combination-definition-symbol)... 72  
 (setf long-combination-definition-users) .... 72  
 (setf macro-definition-access-expander) .... 72  
 (setf macro-definition-foreignp) ..... 72  
 (setf macro-definition-function) ..... 73  
 (setf macro-definition-symbol) ..... 73  
 (setf macro-definition-update-expander) ..... 73  
 (setf method-definition-foreignp)..... 77  
 (setf method-definition-method) ..... 77  
 (setf method-definition-symbol) ..... 77  
 (setf node-after-menu-contents) ..... 77  
 (setf node-before-menu-contents) ..... 78  
 (setf node-children) ..... 78  
 (setf node-name)..... 78  
 (setf node-next)..... 78  
 (setf node-previous) ..... 78  
 (setf node-section-name) ..... 78  
 (setf node-section-type) ..... 78  
 (setf node-synopsis) ..... 78  
 (setf node-up)..... 78  
 (setf setf-expander-definition-access) ..... 83  
 (setf setf-expander-definition-foreignp).... 83  
 (setf setf-expander-definition-symbol)..... 83  
 (setf setf-expander-definition-update) ..... 83  
 (setf short-combination-  
 definition-combination) ..... 83  
 (setf  
 short-combination-definition-foreignp) .... 83

(setf	
short-combination-definition-operator) . . . . .	83
(setf	
short-combination-definition-symbol) . . . . .	84
(setf short-combination-definition-users) . . . . .	84
(setf slot-definition-foreignp) . . . . .	84
(setf slot-definition-readers) . . . . .	84
(setf slot-definition-slot) . . . . .	84
(setf slot-definition-symbol) . . . . .	84
(setf slot-definition-writers) . . . . .	84
(setf special-definition-foreignp) . . . . .	85
(setf special-definition-symbol) . . . . .	85
(setf structure-definition-children) . . . . .	85
(setf structure-definition-foreignp) . . . . .	85
(setf structure-definition-methods) . . . . .	85
(setf structure-definition-parents) . . . . .	85
(setf structure-definition-slots) . . . . .	85
(setf structure-definition-symbol) . . . . .	86
(setf symbol-macro-definition-foreignp) . . . . .	86
(setf symbol-macro-definition-symbol) . . . . .	86
(setf type-definition-foreignp) . . . . .	87
(setf type-definition-symbol) . . . . .	87
(setf writer-definition-foreignp) . . . . .	88
(setf writer-definition-function) . . . . .	88
(setf writer-definition-reader) . . . . .	88
(setf writer-definition-symbol) . . . . .	88
(setf writer-definition-update-expander) . . . . .	88
(setf writer-method-definition-foreignp) . . . . .	88
(setf writer-method-definition-method) . . . . .	88
(setf writer-method-definition-symbol) . . . . .	88

**@**

@anchor . . . . .	55
@defclass . . . . .	50
@defcombination . . . . .	50
@defcompilermacro . . . . .	51
@defcond . . . . .	51
@defconstant . . . . .	51
@deffn . . . . .	51
@defnix . . . . .	55
@defgeneric . . . . .	51
@defgenericx . . . . .	55
@defmacro . . . . .	51
@defmethod . . . . .	51
@defmethodx . . . . .	55
@defsetf . . . . .	52
@defsetfx . . . . .	55
@defslot . . . . .	52
@defspecial . . . . .	52
@defstruct . . . . .	52
@defsymbolmacro . . . . .	52
@defstp . . . . .	52
@deftype . . . . .	53
@defun . . . . .	53
@defunx . . . . .	56
@defvr . . . . .	53
@item . . . . .	53
@itemize . . . . .	53
@itemize-list . . . . .	56
@multitable . . . . .	53
@ref . . . . .	56
@table . . . . .	53
@tableitem . . . . .	54

**A**

accessor-definition-access-expander . . . . .	56
accessor-definition-foreignp . . . . .	56
accessor-definition-function . . . . .	56
accessor-definition-p . . . . .	56
accessor-definition-symbol . . . . .	56
accessor-definition-update-expander . . . . .	57
accessor-definition-writer . . . . .	57
accessor-method-definition-foreignp . . . . .	57
accessor-method-definition-method . . . . .	57
accessor-method-definition-p . . . . .	57
accessor-method-definition-symbol . . . . .	57
accessor-method-definition-writer . . . . .	57
add-categories-node . . . . .	57
add-category-node . . . . .	57
add-child . . . . .	57
add-definition . . . . .	58
add-definitions . . . . .	58
add-definitions-node . . . . .	58
add-external-definitions . . . . .	58
add-files-node . . . . .	58
add-internal-definitions . . . . .	58
add-modules-node . . . . .	58
add-packages . . . . .	58
add-packages-node . . . . .	58
add-status-definitions-node . . . . .	59
add-symbol-definition . . . . .	59
add-symbol-definitions . . . . .	59
add-systems-node . . . . .	59
anchor . . . . .	59
anchor-and-index . . . . .	59
anchor-name . . . . .	89

**B**

boolean-to-feature-expression . . . . .	59
---	----

**C**

category-definitions . . . . .	89, 90
class-definition-children . . . . .	59
class-definition-foreignp . . . . .	59
class-definition-methods . . . . .	60
class-definition-p . . . . .	60
class-definition-parents . . . . .	60
class-definition-slots . . . . .	60
class-definition-symbol . . . . .	60
classoid-definition-children . . . . .	60
classoid-definition-foreignp . . . . .	60
classoid-definition-methods . . . . .	60
classoid-definition-p . . . . .	60
classoid-definition-parents . . . . .	60
classoid-definition-slots . . . . .	61
classoid-definition-symbol . . . . .	61
clindent . . . . .	61
combination-definition-combination . . . . .	61
combination-definition-foreignp . . . . .	61
combination-definition-p . . . . .	61
combination-definition-symbol . . . . .	61
combination-definition-users . . . . .	61
compiler-macro-definition-foreignp . . . . .	61
compiler-macro-definition-function . . . . .	62
compiler-macro-definition-p . . . . .	62
compiler-macro-definition-symbol . . . . .	62

components	62
condition-definition-children	62
condition-definition-foreignp	62
condition-definition-methods	62
condition-definition-p	62
condition-definition-parents	62
condition-definition-slots	62
condition-definition-symbol	63
configuration	48
configure	48
constant-definition-foreignp	63
constant-definition-p	63
constant-definition-symbol	63
context-directory	63
context-external-definitions	63
context-hyperlinksp	63
context-internal-definitions	63
context-p	63
context-packages	63
context-systems	64
copy-accessor-definition	64
copy-accessor-method-definition	64
copy-class-definition	64
copy-classoid-definition	64
copy-combination-definition	64
copy-compiler-macro-definition	64
copy-condition-definition	64
copy-constant-definition	64
copy-context	64
copy-definition	64
copy-funcoid-definition	64
copy-function-definition	65
copy-generic-accessor-definition	65
copy-generic-definition	65
copy-generic-writer-definition	65
copy-long-combination-definition	65
copy-macro-definition	65
copy-method-definition	65
copy-node	65
copy-setf-expander-definition	65
copy-short-combination-definition	65
copy-slot-definition	65
copy-special-definition	65
copy-structure-definition	66
copy-symbol-macro-definition	66
copy-type-definition	66
copy-writer-definition	66
copy-writer-method-definition	66
current-time-string	66

## D

declt	48
defindent	54
definition-combination-users	90
definition-file-definitions	90, 91
definition-foreignp	66
definition-p	66
definition-package	66
definition-package-definitions	91
definition-package-name	66
definition-source	66
definition-source-by-name	67
definition-symbol	67

definitions-pool-size	67
defsystem-dependencies	67
docstring	91, 92
document	93, 94, 95

## E

endpush	54
escape	67
escape-anchor	67

## F

file-definitions	67
file-node	67
file-packages	67
finalize-definitions	68
find-definition	95, 96
find-method-definition	68
first-word-length	68
funcoid-definition-foreignp	68
funcoid-definition-function	68
funcoid-definition-p	68
funcoid-definition-symbol	68
Function, %version	55
Function, (setf accessor-definition-access-expander)	56
Function, (setf accessor-definition-foreignp)	56
Function, (setf accessor-definition-function)	56
Function, (setf accessor-definition-symbol)	56
Function, (setf accessor-definition-update-expander)	57
Function, (setf accessor-definition-writer)	57
Function, (setf accessor-method-definition-foreignp)	57
Function, (setf accessor-method-definition-method)	57
Function, (setf accessor-method-definition-symbol)	57
Function, (setf accessor-method-definition-writer)	57
Function, (setf class-definition-children)	59
Function, (setf class-definition-foreignp)	59
Function, (setf class-definition-methods)	60
Function, (setf class-definition-parents)	60
Function, (setf class-definition-slots)	60
Function, (setf class-definition-symbol)	60
Function, (setf classoid-definition-children)	60
Function, (setf classoid-definition-foreignp)	60
Function, (setf classoid-definition-methods)	60
Function, (setf classoid-definition-parents)	60
Function, (setf classoid-definition-slots)	61
Function, (setf classoid-definition-symbol)	61
Function, (setf combination-definition-combination)	61
Function, (setf combination-definition-foreignp)	61

- Function, (setf combination-definition-symbol) ..... 61
- Function, (setf combination-definition-users) ..... 61
- Function, (setf compiler-macro-definition-foreignp) ..... 61
- Function, (setf compiler-macro-definition-function) ..... 62
- Function, (setf compiler-macro-definition-symbol) ..... 62
- Function, (setf condition-definition-children) ..... 62
- Function, (setf condition-definition-foreignp) ..... 62
- Function, (setf condition-definition-methods) ..... 62
- Function, (setf condition-definition-parents) ..... 62
- Function, (setf condition-definition-slots) .. 62
- Function, (setf condition-definition-symbol) ..... 63
- Function, (setf constant-definition-foreignp) ..... 63
- Function, (setf constant-definition-symbol) .. 63
- Function, (setf context-external-definitions) ..... 63
- Function, (setf context-hyperlinksp) ..... 63
- Function, (setf context-internal-definitions) ..... 63
- Function, (setf context-packages) ..... 63
- Function, (setf context-systems) ..... 64
- Function, (setf definition-foreignp) ..... 66
- Function, (setf definition-symbol) ..... 67
- Function, (setf funcoid-definition-foreignp) ..... 68
- Function, (setf funcoid-definition-function) ..... 68
- Function, (setf funcoid-definition-symbol) .... 68
- Function, (setf function-definition-foreignp) ..... 68
- Function, (setf function-definition-function) ..... 69
- Function, (setf function-definition-symbol) .. 69
- Function, (setf function-definition-update-expander) ..... 69
- Function, (setf generic-accessor-definition-access-expander) ..... 69
- Function, (setf generic-accessor-definition-combination) .. 69
- Function, (setf generic-accessor-definition-foreignp) .... 69
- Function, (setf generic-accessor-definition-function) .... 69
- Function, (setf generic-accessor-definition-methods) ..... 69
- Function, (setf generic-accessor-definition-symbol) ..... 70
- Function, (setf generic-accessor-definition-update-expander) ..... 70
- Function, (setf generic-accessor-definition-writer) ..... 70
- Function, (setf generic-definition-combination) ..... 70
- Function, (setf generic-definition-foreignp) ..... 70
- Function, (setf generic-definition-function) ..... 70
- Function, (setf generic-definition-methods) .. 70
- Function, (setf generic-definition-symbol) ... 70
- Function, (setf generic-definition-update-expander) ..... 71
- Function, (setf generic-writer-definition-combination) ... 71
- Function, (setf generic-writer-definition-foreignp) ..... 71
- Function, (setf generic-writer-definition-function) ..... 71
- Function, (setf generic-writer-definition-methods) ..... 71
- Function, (setf generic-writer-definition-reader) ..... 71
- Function, (setf generic-writer-definition-symbol) ..... 71
- Function, (setf generic-writer-definition-update-expander) ..... 71
- Function, (setf long-combination-definition-combination) .. 72
- Function, (setf long-combination-definition-foreignp) .... 72
- Function, (setf long-combination-definition-symbol) ..... 72
- Function, (setf long-combination-definition-users) ..... 72
- Function, (setf macro-definition-access-expander) ..... 72
- Function, (setf macro-definition-foreignp) ... 72
- Function, (setf macro-definition-function) ... 73
- Function, (setf macro-definition-symbol) ..... 73
- Function, (setf macro-definition-update-expander) ..... 73
- Function, (setf method-definition-foreignp) .. 77
- Function, (setf method-definition-method) .... 77
- Function, (setf method-definition-symbol) .... 77
- Function, (setf node-after-menu-contents) .... 77
- Function, (setf node-before-menu-contents) ... 78
- Function, (setf node-children) ..... 78
- Function, (setf node-name) ..... 78
- Function, (setf node-next) ..... 78
- Function, (setf node-previous) ..... 78
- Function, (setf node-section-name) ..... 78
- Function, (setf node-section-type) ..... 78
- Function, (setf node-synopsis) ..... 78
- Function, (setf node-up) ..... 78
- Function, (setf setf-expander-definition-access) ..... 83
- Function, (setf setf-expander-definition-foreignp) ..... 83
- Function, (setf setf-expander-definition-symbol) ..... 83
- Function, (setf setf-expander-definition-update) ..... 83
- Function, (setf short-combination-definition-combination) ..... 83
- Function, (setf short-combination-definition-foreignp) ... 83
- Function, (setf short-combination-definition-operator) ... 83
- Function, (setf short-combination-definition-symbol) ..... 84

Function, (setf		
short-combination-definition-users) .....	84	
Function, (setf slot-definition-foreignp).....	84	
Function, (setf slot-definition-readers).....	84	
Function, (setf slot-definition-slot) .....	84	
Function, (setf slot-definition-symbol) .....	84	
Function, (setf slot-definition-writers).....	84	
Function, (setf		
special-definition-foreignp) .....	85	
Function, (setf special-definition-symbol)....	85	
Function, (setf		
structure-definition-children) .....	85	
Function, (setf		
structure-definition-foreignp) .....	85	
Function, (setf		
structure-definition-methods) .....	85	
Function, (setf		
structure-definition-parents) .....	85	
Function, (setf structure-definition-slots)..	85	
Function, (setf		
structure-definition-symbol) .....	86	
Function, (setf		
symbol-macro-definition-foreignp) .....	86	
Function, (setf		
symbol-macro-definition-symbol) .....	86	
Function, (setf type-definition-foreignp).....	87	
Function, (setf type-definition-symbol) .....	87	
Function, (setf writer-definition-foreignp) ..	88	
Function, (setf writer-definition-function) ..	88	
Function, (setf writer-definition-reader).....	88	
Function, (setf writer-definition-symbol).....	88	
Function, (setf		
writer-definition-update-expander) .....	88	
Function, (setf		
writer-method-definition-foreignp) .....	88	
Function, (setf		
writer-method-definition-method) .....	88	
Function, (setf		
writer-method-definition-symbol) .....	88	
Function, @anchor .....	55	
Function, @deffnx .....	55	
Function, @defgenericx .....	55	
Function, @defmethodx .....	55	
Function, @defsetfx .....	55	
Function, @defunx .....	56	
Function, @itemize-list .....	56	
Function, @ref .....	56	
Function,		
accessor-definition-access-expander .....	56	
Function, accessor-definition-foreignp.....	56	
Function, accessor-definition-function.....	56	
Function, accessor-definition-p.....	56	
Function, accessor-definition-symbol .....	56	
Function,		
accessor-definition-update-expander .....	57	
Function, accessor-definition-writer .....	57	
Function,		
accessor-method-definition-foreignp .....	57	
Function, accessor-method-definition-method..	57	
Function, accessor-method-definition-p.....	57	
Function, accessor-method-definition-symbol..	57	
Function, accessor-method-definition-writer..	57	
Function, add-categories-node.....	57	
Function, add-category-node .....	57	
Function, add-child.....	57	
Function, add-definition.....	58	
Function, add-definitions.....	58	
Function, add-definitions-node.....	58	
Function, add-external-definitions .....	58	
Function, add-files-node.....	58	
Function, add-internal-definitions .....	58	
Function, add-modules-node .....	58	
Function, add-packages .....	58	
Function, add-packages-node .....	58	
Function, add-status-definitions-node.....	59	
Function, add-symbol-definition.....	59	
Function, add-symbol-definitions.....	59	
Function, add-systems-node .....	59	
Function, anchor .....	59	
Function, anchor-and-index .....	59	
Function, boolean-to-feature-expression.....	59	
Function, class-definition-children .....	59	
Function, class-definition-foreignp .....	59	
Function, class-definition-methods .....	60	
Function, class-definition-p.....	60	
Function, class-definition-parents .....	60	
Function, class-definition-slots.....	60	
Function, class-definition-symbol .....	60	
Function, classoid-definition-children.....	60	
Function, classoid-definition-foreignp.....	60	
Function, classoid-definition-methods.....	60	
Function, classoid-definition-p.....	60	
Function, classoid-definition-parents.....	60	
Function, classoid-definition-slots .....	61	
Function, classoid-definition-symbol.....	61	
Function, clindent.....	61	
Function,		
combination-definition-combination.....	61	
Function, combination-definition-foreignp ....	61	
Function, combination-definition-p .....	61	
Function, combination-definition-symbol.....	61	
Function, combination-definition-users.....	61	
Function,		
compiler-macro-definition-foreignp.....	61	
Function,		
compiler-macro-definition-function.....	62	
Function, compiler-macro-definition-p.....	62	
Function, compiler-macro-definition-symbol...	62	
Function, components .....	62	
Function, condition-definition-children.....	62	
Function, condition-definition-foreignp.....	62	
Function, condition-definition-methods.....	62	
Function, condition-definition-p.....	62	
Function, condition-definition-parents.....	62	
Function, condition-definition-slots .....	62	
Function, condition-definition-symbol.....	63	
Function, configuration.....	48	
Function, configure.....	48	
Function, constant-definition-foreignp.....	63	
Function, constant-definition-p.....	63	
Function, constant-definition-symbol.....	63	
Function, context-directory .....	63	
Function, context-external-definitions.....	63	
Function, context-hyperlinksp .....	63	
Function, context-internal-definitions.....	63	
Function, context-p.....	63	
Function, context-packages .....	63	
Function, context-systems.....	64	
Function, copy-accessor-definition .....	64	
Function, copy-accessor-method-definition ....	64	



- Function, copy-class-definition ..... 64
- Function, copy-classoid-definition ..... 64
- Function, copy-combination-definition ..... 64
- Function, copy-compiler-macro-definition ..... 64
- Function, copy-condition-definition ..... 64
- Function, copy-constant-definition ..... 64
- Function, copy-context ..... 64
- Function, copy-definition ..... 64
- Function, copy-funcoid-definition ..... 64
- Function, copy-function-definition ..... 65
- Function, copy-generic-accessor-definition ... 65
- Function, copy-generic-definition ..... 65
- Function, copy-generic-writer-definition ..... 65
- Function, copy-long-combination-definition ... 65
- Function, copy-macro-definition ..... 65
- Function, copy-method-definition ..... 65
- Function, copy-node ..... 65
- Function, copy-setf-expander-definition ..... 65
- Function, copy-short-combination-definition .. 65
- Function, copy-slot-definition ..... 65
- Function, copy-special-definition ..... 65
- Function, copy-structure-definition ..... 66
- Function, copy-symbol-macro-definition ..... 66
- Function, copy-type-definition ..... 66
- Function, copy-writer-definition ..... 66
- Function, copy-writer-method-definition ..... 66
- Function, current-time-string ..... 66
- Function, declt ..... 48
- Function, definition-foreignp ..... 66
- Function, definition-p ..... 66
- Function, definition-package ..... 66
- Function, definition-package-name ..... 66
- Function, definition-source ..... 66
- Function, definition-source-by-name ..... 67
- Function, definition-symbol ..... 67
- Function, definitions-pool-size ..... 67
- Function, defsystem-dependencies ..... 67
- Function, escape ..... 67
- Function, escape-anchor ..... 67
- Function, file-definitions ..... 67
- Function, file-node ..... 67
- Function, file-packages ..... 67
- Function, finalize-definitions ..... 68
- Function, find-method-definition ..... 68
- Function, first-word-length ..... 68
- Function, funcoid-definition-foreignp ..... 68
- Function, funcoid-definition-function ..... 68
- Function, funcoid-definition-p ..... 68
- Function, funcoid-definition-symbol ..... 68
- Function, function-definition-foreignp ..... 68
- Function, function-definition-function ..... 69
- Function, function-definition-p ..... 69
- Function, function-definition-symbol ..... 69
- Function,
  - function-definition-update-expander ..... 69
- Function, generate-quickutils ..... 69
- Function, generic-accessor-
  - definition-access-expander ..... 69
- Function,
  - generic-accessor-definition-combination ... 69
- Function,
  - generic-accessor-definition-foreignp ..... 69
- Function,
  - generic-accessor-definition-function ..... 69
- Function,
  - generic-accessor-definition-methods ..... 69
- Function, generic-accessor-definition-p ..... 70
- Function,
  - generic-accessor-definition-symbol ..... 70
- Function, generic-accessor-
  - definition-update-expander ..... 70
- Function,
  - generic-accessor-definition-writer ..... 70
- Function, generic-definition-combination ..... 70
- Function, generic-definition-foreignp ..... 70
- Function, generic-definition-function ..... 70
- Function, generic-definition-methods ..... 70
- Function, generic-definition-p ..... 70
- Function, generic-definition-symbol ..... 70
- Function,
  - generic-definition-update-expander ..... 71
- Function,
  - generic-writer-definition-combination ..... 71
- Function,
  - generic-writer-definition-foreignp ..... 71
- Function,
  - generic-writer-definition-function ..... 71
- Function, generic-writer-definition-methods .. 71
- Function, generic-writer-definition-p ..... 71
- Function, generic-writer-definition-reader ... 71
- Function, generic-writer-definition-symbol ... 71
- Function, generic-writer-definition-
  - update-expander ..... 71
- Function, i-reader ..... 71
- Function, lisp-components ..... 72
- Function, lisp-pathnames ..... 72
- Function,
  - long-combination-definition-combination ... 72
- Function,
  - long-combination-definition-foreignp ..... 72
- Function, long-combination-definition-p ..... 72
- Function,
  - long-combination-definition-symbol ..... 72
- Function, long-combination-definition-users .. 72
- Function, macro-definition-access-expander ... 72
- Function, macro-definition-foreignp ..... 72
- Function, macro-definition-function ..... 73
- Function, macro-definition-p ..... 73
- Function, macro-definition-symbol ..... 73
- Function, macro-definition-update-expander ... 73
- Function, make-accessor-definition ..... 73
- Function, make-accessor-method-definition .... 73
- Function, make-class-definition ..... 73
- Function, make-classoid-definition ..... 73
- Function, make-combination-definition ..... 73
- Function, make-compiler-macro-definition .... 74
- Function, make-condition-definition ..... 74
- Function, make-constant-definition ..... 74
- Function, make-context ..... 74
- Function, make-definition ..... 74
- Function, make-definitions-pool ..... 74
- Function, make-funcoid-definition ..... 74
- Function, make-function-definition ..... 74
- Function, make-generic-accessor-definition ... 75
- Function, make-generic-definition ..... 75
- Function, make-generic-writer-definition ..... 75
- Function, make-long-combination-definition ... 75
- Function, make-macro-definition ..... 75
- Function, make-method-definition ..... 75

- Function, make-node..... 75
- Function, make-setf-expander-definition..... 76
- Function, make-short-combination-definition.. 76
- Function, make-slot-definition..... 76
- Function, make-slot-definitions..... 76
- Function, make-special-definition..... 76
- Function, make-structure-definition..... 76
- Function, make-symbol-macro-definition..... 76
- Function, make-type-definition..... 76
- Function, make-writer-definition..... 76
- Function, make-writer-method-definition..... 77
- Function, mapcan-definitions-pool..... 77
- Function, method-definition-foreignp..... 77
- Function, method-definition-method..... 77
- Function, method-definition-p..... 77
- Function, method-definition-symbol..... 77
- Function, method-name..... 77
- Function, module-components..... 77
- Function, module-node..... 77
- Function, nickname-package..... 49
- Function, node-after-menu-contents..... 77
- Function, node-before-menu-contents..... 78
- Function, node-children..... 78
- Function, node-name..... 78
- Function, node-next..... 78
- Function, node-p..... 78
- Function, node-previous..... 78
- Function, node-section-name..... 78
- Function, node-section-type..... 78
- Function, node-synopsis..... 78
- Function, node-up..... 78
- Function, package-definitions..... 79
- Function, package-external-symbols..... 79
- Function, package-internal-symbols..... 79
- Function, parse-contact(s)..... 79
- Function, parse-contact-string..... 79
- Function, pool-combination-users..... 79
- Function, qualifiers..... 79
- Function, read-next-line..... 79
- Function, reference-component..... 80
- Function, relative-location..... 80
- Function, release-status-number..... 80
- Function, render-definition-core..... 80
- Function, render-dependencies..... 80
- Function, render-docstring..... 80
- Function,
  - render-external-definitions-references.... 80
- Function, render-header..... 80
- Function, render-headline..... 81
- Function, render-initargs..... 81
- Function,
  - render-internal-definitions-references.... 81
- Function, render-lambda-list..... 81
- Function, render-location..... 81
- Function, render-method-combination..... 81
- Function, render-node..... 81
- Function, render-packages-references..... 81
- Function, render-references..... 82
- Function, render-slot..... 82
- Function, render-slot-property..... 82
- Function, render-slots..... 82
- Function, render-source..... 82
- Function, render-text..... 82
- Function, render-top-node..... 82
- Function, render-use-list..... 82
- Function, sbcl-has-setf-inverse-meta-info.... 83
- Function, setf-expander-definition-access.... 83
- Function, setf-expander-definition-foreignp.. 83
- Function, setf-expander-definition-p..... 83
- Function, setf-expander-definition-symbol.... 83
- Function, setf-expander-definition-update.... 83
- Function, setf-expander-p..... 83
- Function,
  - short-combination-definition-combination.. 83
- Function,
  - short-combination-definition-foreignp.... 83
  - short-combination-definition-operator.... 83
- Function, short-combination-definition-p.... 84
- Function,
  - short-combination-definition-symbol..... 84
  - short-combination-definition-users..... 84
- Function, slot-definition-foreignp..... 84
- Function, slot-definition-p..... 84
- Function, slot-definition-readers..... 84
- Function, slot-definition-slot..... 84
- Function, slot-definition-symbol..... 84
- Function, slot-definition-writers..... 84
- Function, slot-property..... 84
- Function, special-definition-foreignp..... 85
- Function, special-definition-p..... 85
- Function, special-definition-symbol..... 85
- Function, specializers..... 85
- Function, structure-definition-children..... 85
- Function, structure-definition-foreignp..... 85
- Function, structure-definition-methods..... 85
- Function, structure-definition-p..... 85
- Function, structure-definition-parents..... 85
- Function, structure-definition-slots..... 85
- Function, structure-definition-symbol..... 86
- Function, sub-component-p..... 86
- Function, subsystems..... 86
- Function, symbol-macro-definition-foreignp... 86
- Function, symbol-macro-definition-p..... 86
- Function, symbol-macro-definition-symbol.... 86
- Function, system-base-name..... 86
- Function, system-dependencies..... 86
- Function, system-directory..... 86
- Function, system-external-symbols..... 87
- Function, system-file-name..... 87
- Function, system-file-type..... 87
- Function, system-internal-symbols..... 87
- Function, system-node..... 87
- Function, system-packages..... 87
- Function, tilde-reader..... 87
- Function, type-definition-foreignp..... 87
- Function, type-definition-p..... 87
- Function, type-definition-symbol..... 87
- Function, version..... 49
- Function, writer-definition-foreignp..... 88
- Function, writer-definition-function..... 88
- Function, writer-definition-p..... 88
- Function, writer-definition-reader..... 88
- Function, writer-definition-symbol..... 88
- Function, writer-definition-update-expander.. 88
- Function, writer-method-definition-foreignp.. 88
- Function, writer-method-definition-method.... 88
- Function, writer-method-definition-p..... 88
- Function, writer-method-definition-symbol.... 88

function-definition-foreignp ..... 68  
 function-definition-function ..... 69  
 function-definition-p ..... 69  
 function-definition-symbol ..... 69  
 function-definition-update-expander ..... 69

## G

generate-quickutils ..... 69  
 Generic Function, anchor-name ..... 89  
 Generic Function, category-definitions ..... 89  
 Generic Function,  
   definition-combination-users ..... 90  
 Generic Function,  
   definition-file-definitions ..... 90  
 Generic Function,  
   definition-package-definitions ..... 91  
 Generic Function, docstring ..... 91  
 Generic Function, document ..... 93  
 Generic Function, find-definition ..... 95  
 Generic Function, headline-function ..... 96  
 Generic Function, index ..... 96  
 Generic Function, lambda-list ..... 98  
 Generic Function, name ..... 99  
 Generic Function, pretty-specializer ..... 100  
 Generic Function, reader-definitions ..... 100  
 Generic Function, reference ..... 100  
 Generic Function, render-dependency ..... 101  
 Generic Function, reveal ..... 101  
 Generic Function, source ..... 101  
 Generic Function,  
   system-dependency-subsystem ..... 103  
 Generic Function, title ..... 103  
 Generic Function, type-name ..... 103  
 Generic Function, virtual-path ..... 105  
 Generic Function, writer-definitions ..... 105  
 generic-accessor-definition-  
   access-expander ..... 69  
 generic-accessor-definition-combination ..... 69  
 generic-accessor-definition-foreignp ..... 69  
 generic-accessor-definition-function ..... 69  
 generic-accessor-definition-methods ..... 69  
 generic-accessor-definition-p ..... 70  
 generic-accessor-definition-symbol ..... 70  
 generic-accessor-definition-  
   update-expander ..... 70  
 generic-accessor-definition-writer ..... 70  
 generic-definition-combination ..... 70  
 generic-definition-foreignp ..... 70  
 generic-definition-function ..... 70  
 generic-definition-methods ..... 70  
 generic-definition-p ..... 70  
 generic-definition-symbol ..... 70  
 generic-definition-update-expander ..... 71  
 generic-writer-definition-combination ..... 71  
 generic-writer-definition-foreignp ..... 71  
 generic-writer-definition-function ..... 71  
 generic-writer-definition-methods ..... 71  
 generic-writer-definition-p ..... 71  
 generic-writer-definition-reader ..... 71  
 generic-writer-definition-symbol ..... 71  
 generic-writer-definition-update-expander ..... 71

## H

headline-function ..... 96

## I

i-reader ..... 71  
 in-readtable ..... 54  
 index ..... 96, 97, 98

## L

lambda-list ..... 98, 99  
 lisp-components ..... 72  
 lisp-pathnames ..... 72  
 long-combination-definition-combination ..... 72  
 long-combination-definition-foreignp ..... 72  
 long-combination-definition-p ..... 72  
 long-combination-definition-symbol ..... 72  
 long-combination-definition-users ..... 72

## M

Macro, @defclass ..... 50  
 Macro, @defcombination ..... 50  
 Macro, @defcompilermacro ..... 51  
 Macro, @defcond ..... 51  
 Macro, @defconstant ..... 51  
 Macro, @deffn ..... 51  
 Macro, @defgeneric ..... 51  
 Macro, @defmacro ..... 51  
 Macro, @defmethod ..... 51  
 Macro, @defsetf ..... 52  
 Macro, @defslot ..... 52  
 Macro, @defspecial ..... 52  
 Macro, @defstruct ..... 52  
 Macro, @defsymbolmacro ..... 52  
 Macro, @deftp ..... 52  
 Macro, @deftype ..... 53  
 Macro, @defun ..... 53  
 Macro, @defvr ..... 53  
 Macro, @item ..... 53  
 Macro, @itemize ..... 53  
 Macro, @multitable ..... 53  
 Macro, @table ..... 53  
 Macro, @tableitem ..... 54  
 Macro, defindent ..... 54  
 Macro, endpush ..... 54  
 Macro, in-readtable ..... 54  
 Macro, render-classoid ..... 54  
 Macro, render-combination ..... 54  
 Macro, render-funcoid ..... 54  
 Macro, render-method ..... 54  
 Macro, render-to-string ..... 55  
 Macro, render-varoid ..... 55  
 Macro, when-let ..... 47  
 Macro, when-let\* ..... 48  
 macro-definition-access-expander ..... 72  
 macro-definition-foreignp ..... 72  
 macro-definition-function ..... 73  
 macro-definition-p ..... 73  
 macro-definition-symbol ..... 73  
 macro-definition-update-expander ..... 73  
 make-accessor-definition ..... 73  
 make-accessor-method-definition ..... 73

make-class-definition	73
make-classoid-definition	73
make-combination-definition	73
make-compiler-macro-definition	74
make-condition-definition	74
make-constant-definition	74
make-context	74
make-definition	74
make-definitions-pool	74
make-funcooid-definition	74
make-function-definition	74
make-generic-accessor-definition	75
make-generic-definition	75
make-generic-writer-definition	75
make-long-combination-definition	75
make-macro-definition	75
make-method-definition	75
make-node	75
make-setf-expander-definition	76
make-short-combination-definition	76
make-slot-definition	76
make-slot-definitions	76
make-special-definition	76
make-structure-definition	76
make-symbol-macro-definition	76
make-type-definition	76
make-writer-definition	76
make-writer-method-definition	77
mapcan-definitions-pool	77
Method, anchor-name	89
Method, category-definitions	89, 90
Method, definition-combination-users	90
Method, definition-file-definitions	90, 91
Method, definition-package-definitions	91
Method, docstring	91, 92
Method, document	93, 94, 95
Method, find-definition	96
Method, headline-function	96
Method, index	96, 97, 98
Method, lambda-list	98, 99
Method, name	99, 100
Method, pretty-specializer	100
Method, reader-definitions	100
Method, reference	100, 101
Method, render-dependency	101
Method, reveal	101
Method, source	102
Method, system-dependency-subsystem	103
Method, title	103
Method, type-name	103, 104, 105
Method, virtual-path	105
Method, writer-definitions	105
method-definition-foreignp	77
method-definition-method	77
method-definition-p	77
method-definition-symbol	77
method-name	77
module-components	77
module-node	77

## N

name	99, 100
nickname-package	49
node-after-menu-contents	77
node-before-menu-contents	78
node-children	78
node-name	78
node-next	78
node-p	78
node-previous	78
node-section-name	78
node-section-type	78
node-synopsis	78
node-up	78

## P

package-definitions	79
package-external-symbols	79
package-internal-symbols	79
parse-contact(s)	79
parse-contact-string	79
pool-combination-users	79
pretty-specializer	100

## Q

qualifiers	79
------------	----

## R

read-next-line	79
reader-definitions	100
reference	100, 101
reference-component	80
relative-location	80
release-status-number	80
render-classoid	54
render-combination	54
render-definition-core	80
render-dependencies	80
render-dependency	101
render-docstring	80
render-external-definitions-references	80
render-funcooid	54
render-header	80
render-headline	81
render-initargs	81
render-internal-definitions-references	81
render-lambda-list	81
render-location	81
render-method	54
render-method-combination	81
render-node	81
render-packages-references	81
render-references	82
render-slot	82
render-slot-property	82
render-slots	82
render-source	82
render-text	82
render-to-string	55
render-top-node	82
render-use-list	82

render-varoid ..... 55  
 reveal ..... 101

## S

sbcl-has-setf-inverse-meta-info ..... 83  
 setf-expander-definition-access ..... 83  
 setf-expander-definition-foreignp ..... 83  
 setf-expander-definition-p ..... 83  
 setf-expander-definition-symbol ..... 83  
 setf-expander-definition-update ..... 83  
 setf-expander-p ..... 83  
 short-combination-definition-combination .... 83  
 short-combination-definition-foreignp ..... 83  
 short-combination-definition-operator ..... 83  
 short-combination-definition-p ..... 84  
 short-combination-definition-symbol ..... 84  
 short-combination-definition-users ..... 84  
 slot-definition-foreignp ..... 84  
 slot-definition-p ..... 84  
 slot-definition-readers ..... 84  
 slot-definition-slot ..... 84  
 slot-definition-symbol ..... 84  
 slot-definition-writers ..... 84  
 slot-property ..... 84  
 source ..... 101, 102  
 special-definition-foreignp ..... 85  
 special-definition-p ..... 85  
 special-definition-symbol ..... 85  
 specializers ..... 85  
 structure-definition-children ..... 85  
 structure-definition-foreignp ..... 85  
 structure-definition-methods ..... 85  
 structure-definition-p ..... 85  
 structure-definition-parents ..... 85  
 structure-definition-slots ..... 85  
 structure-definition-symbol ..... 86  
 sub-component-p ..... 86  
 subsystems ..... 86  
 symbol-macro-definition-foreignp ..... 86  
 symbol-macro-definition-p ..... 86

symbol-macro-definition-symbol ..... 86  
 system-base-name ..... 86  
 system-dependencies ..... 86  
 system-dependency-subsystem ..... 103  
 system-directory ..... 86  
 system-external-symbols ..... 87  
 system-file-name ..... 87  
 system-file-type ..... 87  
 system-internal-symbols ..... 87  
 system-node ..... 87  
 system-packages ..... 87

## T

tilde-reader ..... 87  
 title ..... 103  
 type-definition-foreignp ..... 87  
 type-definition-p ..... 87  
 type-definition-symbol ..... 87  
 type-name ..... 103, 104, 105

## V

version ..... 49  
 virtual-path ..... 105

## W

when-let ..... 47  
 when-let\* ..... 48  
 writer-definition-foreignp ..... 88  
 writer-definition-function ..... 88  
 writer-definition-p ..... 88  
 writer-definition-reader ..... 88  
 writer-definition-symbol ..... 88  
 writer-definition-update-expander ..... 88  
 writer-definitions ..... 105  
 writer-method-definition-foreignp ..... 88  
 writer-method-definition-method ..... 88  
 writer-method-definition-p ..... 88  
 writer-method-definition-symbol ..... 88

## A.3 Variables

**\***

<code>*categories*</code> .....	50
<code>*configuration*</code> .....	50
<code>*licenses*</code> .....	50
<code>*readtable*</code> .....	50
<code>*release-major-level*</code> .....	47
<code>*release-minor-level*</code> .....	47
<code>*release-name*</code> .....	47
<code>*release-status*</code> .....	47
<code>*release-status-level*</code> .....	47
<code>*section-names*</code> .....	50

**A**

<code>access</code> .....	115
<code>access-expander</code> .....	106, 111, 113
<code>after-menu-contents</code> .....	115

**B**

<code>before-menu-contents</code> .....	115
---	-----

**C**

<code>children</code> .....	107, 115
<code>combination</code> .....	108, 112

**E**

<code>external-definitions</code> .....	109
---	-----

**F**

<code>foreignp</code> .....	110
<code>function</code> .....	110

**H**

<code>hyperlinksp</code> .....	109
--------------------------------	-----

**I**

<code>internal-definitions</code> .....	109
---	-----

**M**

<code>method</code> .....	114
<code>methods</code> .....	107, 112

**N**

<code>name</code> .....	114
<code>next</code> .....	114

**O**

<code>operator</code> .....	116
-----------------------------	-----

**P**

<code>packages</code> .....	109
<code>parents</code> .....	107
<code>previous</code> .....	115

**R**

<code>reader</code> .....	112, 118
<code>readers</code> .....	116

**S**

<code>section-name</code> .....	114
<code>section-type</code> .....	114
<code>slot</code> .....	116
Slot, <code>access</code> .....	115
Slot, <code>access-expander</code> .....	106, 111, 113
Slot, <code>after-menu-contents</code> .....	115
Slot, <code>before-menu-contents</code> .....	115
Slot, <code>children</code> .....	107, 115
Slot, <code>combination</code> .....	108, 112
Slot, <code>external-definitions</code> .....	109
Slot, <code>foreignp</code> .....	110
Slot, <code>function</code> .....	110
Slot, <code>hyperlinksp</code> .....	109
Slot, <code>internal-definitions</code> .....	109
Slot, <code>method</code> .....	114
Slot, <code>methods</code> .....	107, 112
Slot, <code>name</code> .....	114
Slot, <code>next</code> .....	114
Slot, <code>operator</code> .....	116
Slot, <code>packages</code> .....	109
Slot, <code>parents</code> .....	107
Slot, <code>previous</code> .....	115
Slot, <code>reader</code> .....	112, 118
Slot, <code>readers</code> .....	116
Slot, <code>section-name</code> .....	114
Slot, <code>section-type</code> .....	114
Slot, <code>slot</code> .....	116
Slot, <code>slots</code> .....	107
Slot, <code>symbol</code> .....	110
Slot, <code>synopsis</code> .....	114
Slot, <code>systems</code> .....	109
Slot, <code>up</code> .....	115
Slot, <code>update</code> .....	116
Slot, <code>update-expander</code> .....	111, 112, 113
Slot, <code>users</code> .....	108
Slot, <code>writer</code> .....	106, 111
Slot, <code>writers</code> .....	116
<code>slots</code> .....	107
Special Variable, <code>*categories*</code> .....	50
Special Variable, <code>*configuration*</code> .....	50
Special Variable, <code>*licenses*</code> .....	50
Special Variable, <code>*readtable*</code> .....	50
Special Variable, <code>*release-major-level*</code> .....	47
Special Variable, <code>*release-minor-level*</code> .....	47
Special Variable, <code>*release-name*</code> .....	47
Special Variable, <code>*release-status*</code> .....	47
Special Variable, <code>*release-status-level*</code> .....	47
Special Variable, <code>*section-names*</code> .....	50
<code>symbol</code> .....	110

synopsis..... 114  
systems..... 109

## U

up..... 115  
update..... 116  
update-expander..... 111, 112, 113  
users..... 108

## W

writer..... 106, 111  
writers..... 116

## A.4 Data types

### A

accessor-definition ..... 105  
 accessor-method-definition ..... 106

### C

class-definition ..... 106  
 classoid-definition ..... 107  
 combination-definition ..... 107  
 compiler-macro-definition ..... 108  
 condition-definition ..... 108  
 constant-definition ..... 108  
 context ..... 109

### D

definition ..... 109

### F

funcoid-definition ..... 110  
 function-definition ..... 110

### G

generic-accessor-definition ..... 111  
 generic-definition ..... 111  
 generic-writer-definition ..... 112

### L

long-combination-definition ..... 113

### M

macro-definition ..... 113  
 method-definition ..... 113

### N

net.didierverna.declt ..... 5, 29  
 net.didierverna.declt.core ..... 5  
 net.didierverna.declt.setup ..... 6, 46  
 node ..... 114

### P

Package, net.didierverna.declt ..... 29  
 Package, net.didierverna.declt.setup ..... 46  
 Package, quickutil ..... 29

### Q

quickutil ..... 29

### S

setf-expander-definition ..... 115  
 short-combination-definition ..... 116  
 slot-definition ..... 116  
 special-definition ..... 117  
 Structure, accessor-definition ..... 105  
 Structure, accessor-method-definition ..... 106  
 Structure, class-definition ..... 106  
 Structure, classoid-definition ..... 107  
 Structure, combination-definition ..... 107  
 Structure, compiler-macro-definition ..... 108  
 Structure, condition-definition ..... 108  
 Structure, constant-definition ..... 108  
 Structure, context ..... 109  
 Structure, definition ..... 109  
 Structure, funcoid-definition ..... 110  
 Structure, function-definition ..... 110  
 Structure, generic-accessor-definition ..... 111  
 Structure, generic-definition ..... 111  
 Structure, generic-writer-definition ..... 112  
 Structure, long-combination-definition ..... 113  
 Structure, macro-definition ..... 113  
 Structure, method-definition ..... 113  
 Structure, node ..... 114  
 Structure, setf-expander-definition ..... 115  
 Structure, short-combination-definition ..... 116  
 Structure, slot-definition ..... 116  
 Structure, special-definition ..... 117  
 Structure, structure-definition ..... 117  
 Structure, symbol-macro-definition ..... 117  
 Structure, type-definition ..... 117  
 Structure, writer-definition ..... 118  
 Structure, writer-method-definition ..... 118  
 structure-definition ..... 117  
 symbol-macro-definition ..... 117  
 System, net.didierverna.declt ..... 5  
 System, net.didierverna.declt.core ..... 5  
 System, net.didierverna.declt.setup ..... 6

### T

type-definition ..... 117

### W

writer-definition ..... 118  
 writer-method-definition ..... 118