The Declt Reference Manual

Documentation Extractor from Common Lisp to Texinfo, version 4.0 beta 3 "William Riker"

Didier Verna <didier@didierverna.net>

This manual was generated automatically by Declt 4.0 beta 3 "William Riker" on Sun Jun 15 22:31:57 2025 GMT+1.

Table of Contents

Copying

1 Introduction

`Declt` (pronounce "dec'let") is a reference manual generator for Common Lisp libraries. A `Declt` manual documents one specified ASDF system (considered as the "main" system), and all its local dependencies (subsystems found in the same distribution). This is what is collectively referred to as the *library*.

`Declt` doesn't perform any kind of static code analysis, but instead loads the library, and then introspects the Lisp environment to discover what "belongs" to it. The generated documentation includes the description of both programmatic and ASDF components. Every such component description is called a *definition*.

`Declt` manuals provide a detailed description of the library's infrastructure by including definitions for every relevant ASDF component (systems, modules, and files), and Lisp package.

Exported programmatic definitions are split from the internal ones, which allows to separately browse either the library's public interface or its implementation. Both sections of the manual include definitions for constants, special variables, symbol macros, macros, `setf` expanders, compiler macros, regular functions (including `setf` ones), generic functions and methods (including `setf` ones), method combinations, conditions, structures, classes, and types.

Programmatic definitions are as complete and exhaustive as introspection can make them. `Declt` collects documentation strings, lambda lists (including qualifiers and specializers where appropriate), slot definitions (including type information, allocation type, initialization arguments, *etc.*), definition sources, *etc.*

Every definition includes a full set of cross-references to related ones: ASDF component dependencies, parents, and children, classes direct methods, super- and sub-classes, slot readers and writers, `setf` expanders access and update functions, *etc.*

Finally, `Declt` produces exhaustive and multiple-entry indexes to all documented aspect of the library.

`Declt` manuals are generated in Texinfo format. From there it is possible to produce readable / printable output in Info, HTML, PDF, PostScript, *etc.*

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

A reference manual generator for Common Lisp libraries

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/typesetting.php#declt

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

License BSD

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.

If Feature :sbcl

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

Source [net.didierverna.declt.asd], page 11.

2.2 net.didierverna.declt.setup

Declt's preload 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/typesetting.php#declt

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

License BSD

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

Dependency
`named-readtables` (system).

Source [`net.didierverna.declt.setup.asd`], page 11.

Child Components
- [`package.lisp`], page 11 (file).
- [`src`], page 9 (module).

2.3 `net.didierverna.declt.core`

Declt's core functionality

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/typesetting.php#declt`

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

License BSD

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

If Feature `:sbcl`

Dependencies
- `sb-introspect` (system)., required, for feature `:sbcl`
- [`net.didierverna.declt.setup`], page 5 (system).
- [`net.didierverna.declt.assess`], page 6 (system).

Source [`net.didierverna.declt.core.asd`], page 11.

Child Components
- [`package.lisp`], page 13 (file).
- [`src`], page 9 (module).

2.4 `net.didierverna.declt.assess`

Declt's information gathering pipeline stage

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

Author Didier Verna

Contact `didier@didierverna.net`

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

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

License BSD

Long Description
The Declt assessment library collects information from ASDF systems by introspection, and produces an abstract representation, independent from both the final manual's organization and the output format. For a more complete description of Declt, see the 'net.didierverna.declt' system.

If Feature `:sbcl`

Dependencies
- `sb-introspect` (system)., required, for feature `:sbcl`
- [`net.didierverna.declt.setup`], page 5 (system).

Source [`net.didierverna.declt.assess.asd`], page 11.

Child Components
- [`package.lisp`], page 19 (file).
- [`src`], page 10 (module).

3 Modules

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

3.1 net.didierverna.declt.setup/src

Dependency
[package.lisp], page 11 (file).

Source [net.didierverna.declt.setup.asd], page 11.

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

Child Components
- [configuration.lisp], page 11 (file).
- [readtable.lisp], page 12 (file).
- [version.lisp], page 12 (file).
- [util.lisp], page 12 (file).

3.2 net.didierverna.declt.core/src

Dependency
[package.lisp], page 13 (file).

Source [net.didierverna.declt.core.asd], page 11.

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

Child Components
- [util], page 9 (module).
- [doc], page 9 (module).
- [declt.lisp], page 19 (file).

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

Source [net.didierverna.declt.core.asd], page 11.

Parent Component
[src], page 9 (module).

Child Component
[misc.lisp], page 13 (file).

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

Dependency
[util], page 9 (module).

Source [net.didierverna.declt.core.asd], page 11.

Parent Component
[src], page 9 (module).

Child Components
- [texi.lisp], page 13 (file).

- [doc.lisp], page 15 (file).
- [symbol.lisp], page 15 (file).
- [package.lisp], page 18 (file).
- [asdf.lisp], page 18 (file).

3.5 net.didierverna.declt.assess/src

Dependency
[package.lisp], page 19 (file).

Source [net.didierverna.declt.assess.asd], page 11.

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

Child Components

- [util.lisp], page 19 (file).
- [definition.lisp], page 19 (file).
- [license.lisp], page 20 (file).
- [symbol.lisp], page 20 (file).
- [package.lisp], page 24 (file).
- [asdf.lisp], page 25 (file).
- [finalize.lisp], page 27 (file).
- [assess.lisp], page 27 (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/net.didierverna.declt.asd

Source [net.didierverna.declt.asd], page 11.

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

ASDF Systems
[net.didierverna.declt], page 5.

4.1.2 net.didierverna.declt.setup/net.didierverna.declt.setup.asd

Source [net.didierverna.declt.setup.asd], page 11.

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

ASDF Systems
[net.didierverna.declt.setup], page 5.

4.1.3 net.didierverna.declt.core/net.didierverna.declt.core.asd

Source [net.didierverna.declt.core.asd], page 11.

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

ASDF Systems
[net.didierverna.declt.core], page 6.

4.1.4 net.didierverna.declt.assess/net.didierverna.declt.assess.asd

Source [net.didierverna.declt.assess.asd], page 11.

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

ASDF Systems
[net.didierverna.declt.assess], page 6.

4.1.5 net.didierverna.declt.setup/package.lisp

Source [net.didierverna.declt.setup.asd], page 11.

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

Packages [net.didierverna.declt.setup], page 31.

4.1.6 net.didierverna.declt.setup/src/configuration.lisp

Source [net.didierverna.declt.setup.asd], page 11.

Parent Component
[src], page 9 (module).

Public Interface

- [configuration], page 43 (function).
- [configure], page 43 (function).

Internals [*configuration*], page 105 (special variable).

4.1.7 net.didierverna.declt.setup/src/readtable.lisp

Dependency

[configuration.lisp], page 11 (file).

Source [net.didierverna.declt.setup.asd], page 11.

Parent Component

[src], page 9 (module).

Internals

- [clindent], page 110 (function).
- [defindent], page 106 (macro).
- [i-reader], page 112 (function).

4.1.8 net.didierverna.declt.setup/src/version.lisp

Dependency

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

Source [net.didierverna.declt.setup.asd], page 11.

Parent Component

[src], page 9 (module).

Public Interface

- [*copyright-years*], page 41 (special variable).
- [*release-major-level*], page 41 (special variable).
- [*release-minor-level*], page 41 (special variable).
- [*release-name*], page 41 (special variable).
- [*release-status*], page 41 (special variable).
- [*release-status-level*], page 41 (special variable).
- [version], page 48 (function).

Internals

- [%version], page 107 (function).
- [release-status-number], page 121 (function).

4.1.9 net.didierverna.declt.setup/src/util.lisp

Dependency

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

Source [net.didierverna.declt.setup.asd], page 11.

Parent Component

[src], page 9 (module).

Public Interface

- [abstract-class], page 74 (class).
- [declare-valid-superclass], page 41 (macro).

- [defabstract], page 42 (macro).
- [endpush], page 42 (macro).
- [find*], page 45 (function).
- [make-instance], page 73 (method).
- [mapcat], page 46 (function).
- [non-empty-string], page 104 (type).
- [non-empty-string-p], page 47 (function).
- [retain], page 47 (function).
- [validate-superclass], page 73 (method).
- [validate-superclass], page 73 (method).
- [when-let], page 42 (macro).
- [when-let*], page 42 (macro).
- [while], page 42 (macro).

4.1.10 net.didierverna.declt.core/package.lisp

Source [net.didierverna.declt.core.asd], page 11.

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

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

Public Interface
[nickname-package], page 46 (function).

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

Source [net.didierverna.declt.core.asd], page 11.

Parent Component
[util], page 9 (module).

Internals [current-time-string], page 110 (function).

4.1.12 net.didierverna.declt.core/src/doc/texi.lisp

Source [net.didierverna.declt.core.asd], page 11.

Parent Component
[doc], page 9 (module).

Internals

- [%deffn], page 107 (function).
- [*fragile-characters*], page 105 (special variable).
- [*section-names*], page 105 (special variable).
- [*special-characters*], page 105 (special variable).
- [@anchor], page 107 (function).
- [@deffn], page 107 (function).
- [@deffnx], page 108 (function).
- [@deftp], page 108 (function).
- [@defvr], page 108 (function).
- [@end], page 108 (function).

- [render-node], page 122 (function).
- [render-text], page 122 (function).
- [render-to-string], page 107 (macro).
- [render-top-node], page 122 (function).
- [table], page 107 (macro).

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

Dependency
[texi.lisp], page 13 (file).

Source [net.didierverna.declt.core.asd], page 11.

Parent Component
[doc], page 9 (module).

Internals

- [*blanks*], page 104 (special variable).
- [anchor], page 110 (function).
- [anchor-and-index], page 110 (function).
- [anchor-name], page 110 (function).
- [category-name], page 125 (generic function).
- [context], page 142 (class).
- [declt-notice], page 128 (reader method).
- [default-values], page 128 (reader method).
- [document], page 128 (generic function).
- [document], page 141 (method combination).
- [foreign-definitions], page 134 (reader method).
- [index], page 112 (function).
- [index-command-name], page 134 (generic function).
- [locations], page 137 (reader method).
- [long-title], page 113 (function).
- [make-context], page 115 (function).
- [reference], page 121 (function).
- [render-docstring], page 122 (function).
- [render-references], page 122 (function).
- [reveal], page 123 (function).
- [safe-name], page 137 (generic function).

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

Dependency
[doc.lisp], page 15 (file).

Source [net.didierverna.declt.core.asd], page 11.

Parent Component
[doc], page 9 (module).

Internals

- [*categories*], page 104 (special variable).

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

Dependency
[symbol.lisp], page 15 (file).

Source [net.didierverna.declt.core.asd], page 11.

Parent Component
[doc], page 9 (module).

Internals

- [add-packages-node], page 109 (function).
- [category-name], page 125 (method).
- [document], page 129 (method).
- [index-command-name], page 134 (method).

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

Dependency
[package.lisp], page 18 (file).

Source [net.didierverna.declt.core.asd], page 11.

Parent Component
[doc], page 9 (module).

Internals

- [add-files-node], page 109 (function).
- [add-modules-node], page 109 (function).
- [add-systems-node], page 110 (function).
- [category-name], page 125 (method).
- [category-name], page 125 (method).
- [category-name], page 125 (method).
- [document], page 128 (method).
- [document], page 128 (method).
- [document], page 128 (method).
- [document], page 128 (method).
- [document], page 129 (method).
- [file-node], page 112 (function).
- [index-command-name], page 134 (method).
- [index-command-name], page 134 (method).
- [index-command-name], page 134 (method).
- [render-dependencies], page 121 (function).
- [render-dependency], page 121 (function).
- [safe-name], page 137 (method).
- [safe-name], page 137 (method).

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

Dependency
[doc], page 9 (module).

Source [net.didierverna.declt.core.asd], page 11.

Parent Component
[src], page 9 (module).

Public Interface
[declt], page 44 (function).

Internals

- [declt-1], page 110 (function).
- [render-header], page 122 (function).
- [select-keys], page 124 (function).

4.1.18 net.didierverna.declt.assess/package.lisp

Source [net.didierverna.declt.assess.asd], page 11.

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

Packages [net.didierverna.declt.assess], page 32.

Public Interface
[nickname-package], page 46 (function).

4.1.19 net.didierverna.declt.assess/src/util.lisp

Source [net.didierverna.declt.assess.asd], page 11.

Parent Component
[src], page 10 (module).

Internals

- [one-liner-p], page 120 (function).
- [parse-contact(s)], page 120 (function).
- [parse-contact-string], page 121 (function).
- [reorder-dependency-def], page 123 (function).
- [reordered-dependency-def-system], page 123 (function).
- [source-by-name], page 124 (function).
- [source-by-object], page 124 (function).
- [validate-email], page 125 (function).

4.1.20 net.didierverna.declt.assess/src/definition.lisp

Dependency
[util.lisp], page 19 (file).

Source [net.didierverna.declt.assess.asd], page 11.

Parent Component
[src], page 10 (module).

Public Interface

- [definition], page 82 (class).

- [docstring], page 58 (generic function).
- [foreignp], page 61 (reader method).
- [(setf foreignp)], page 61 (writer method).
- [name], page 65 (generic function).
- [object], page 65 (reader method).
- [print-object], page 73 (method).
- [private-definitions], page 66 (generic function).
- [public-definitions], page 67 (generic function).
- [source-file], page 68 (reader method).
- [(setf source-file)], page 68 (writer method).
- [uid], page 71 (reader method).
- [(setf uid)], page 71 (writer method).

Internals

- [domesticp], page 111 (function).
- [find-definition], page 112 (function).
- [source-pathname], page 138 (generic function).

4.1.21 net.didierverna.declt.assess/src/license.lisp

Dependency
[definition.lisp], page 19 (file).

Source [net.didierverna.declt.assess.asd], page 11.

Parent Component
[src], page 10 (module).

Internals [*licenses*], page 105 (special variable).

4.1.22 net.didierverna.declt.assess/src/symbol.lisp

Dependency
[license.lisp], page 20 (file).

Source [net.didierverna.declt.assess.asd], page 11.

Parent Component
[src], page 10 (module).

Public Interface

- [accessor-method-definition], page 74 (class).
- [accessor-mixin], page 74 (class).
- [alias-definition], page 75 (class).
- [allocation], page 42 (function).
- [class-definition], page 76 (class).
- [classoid], page 49 (reader method).
- [classoid-definition], page 76 (class).
- [clients], page 49 (reader method).
- [(setf clients)], page 49 (writer method).
- [clos-classoid-mixin], page 77 (class).
- [clos-slot-definition], page 78 (class).

- [typed-structure-definition], page 102 (class).
- [typed-structure-slot-definition], page 103 (class).
- [value-type], page 72 (generic function).
- [variable-definition], page 103 (class).
- [varoid-definition], page 104 (class).
- [writer-method-definition], page 104 (class).
- [writer-method-definition-p], page 48 (function).
- [writers], page 72 (reader method).
- [(setf writers)], page 72 (writer method).

Internals

- [dd-element-type], page 106 (macro).
- [definition-source-by-name], page 111 (function).
- [make-classoid-definition], page 114 (function).
- [make-clos-slot-definition], page 114 (function).
- [make-combination-definition], page 114 (function).
- [make-compiler-macro-alias-definition], page 114 (function).
- [make-compiler-macro-definition], page 114 (function).
- [make-constant-definition], page 115 (function).
- [make-expander-definition], page 115 (function).
- [make-function-alias-definition], page 115 (function).
- [make-generic-function-definition], page 115 (function).
- [make-macro-alias-definition], page 115 (function).
- [make-macro-definition], page 115 (function).
- [make-method-definition], page 116 (function).
- [make-ordinary-function-definition], page 116 (function).
- [make-special-definition], page 116 (function).
- [make-symbol-macro-definition], page 116 (function).
- [make-type-definition], page 117 (function).
- [make-typed-structure-slot-definition], page 117 (function).
- [method-name], page 118 (function).
- [source-pathname], page 138 (method).
- [source-pathname], page 138 (method).
- [source-pathname], page 138 (method).
- [source-pathname], page 138 (method).
- [source-pathname], page 138 (method).
- [source-pathname], page 138 (method).
- [source-pathname], page 138 (method).
- [source-pathname], page 139 (method).

4.1.23 net.didierverna.declt.assess/src/package.lisp

Dependency

[symbol.lisp], page 20 (file).

Source [net.didierverna.declt.assess.asd], page 11.

Parent Component
[src], page 10 (module).

Public Interface
- [definition-package], page 52 (reader method).
- [definitions], page 53 (reader method).
- [(setf definitions)], page 53 (writer method).
- [initialize-instance], page 73 (method).
- [name], page 65 (method).
- [nicknames], page 46 (function).
- [package-definition], page 92 (class).
- [package-definition-p], page 47 (function).
- [private-definitions], page 66 (method).
- [public-definitions], page 67 (method).
- [use-list], page 71 (reader method).
- [(setf use-list)], page 71 (writer method).
- [used-by-list], page 72 (reader method).
- [(setf used-by-list)], page 72 (writer method).

Internals
- [external-symbols], page 134 (reader method).
- [(setf external-symbols)], page 134 (writer method).
- [internal-symbols], page 136 (reader method).
- [(setf internal-symbols)], page 136 (writer method).
- [make-package-definition], page 116 (function).
- [package-external-symbols], page 120 (function).
- [package-internal-symbols], page 120 (function).

4.1.24 net.didierverna.declt.assess/src/asdf.lisp

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

Source [net.didierverna.declt.assess.asd], page 11.

Parent Component
[src], page 10 (module).

Public Interface
- [authors], page 48 (reader method).
- [(setf authors)], page 48 (writer method).
- [bug-tracker], page 43 (function).
- [c-file-definition], page 75 (class).
- [children], page 49 (reader method).
- [(setf children)], page 49 (writer method).
- [cl-source-file.asd], page 75 (class).
- [component], page 50 (reader method).
- [component-definition], page 80 (class).
- [component-definition-p], page 43 (function).

- [system-file-definition], page 102 (class).

Internals

- [make-file-definition], page 115 (function).
- [make-module-definition], page 116 (function).
- [make-system-definition], page 117 (function).
- [make-system-file-definition], page 117 (function).
- [make-system-file-definitions], page 117 (function).
- [source-pathname], page 138 (method).

4.1.25 net.didierverna.declt.assess/src/finalize.lisp

Dependency
[asdf.lisp], page 25 (file).

Source [net.didierverna.declt.assess.asd], page 11.

Parent Component
[src], page 10 (module).

Internals

- [*stabilized*], page 105 (special variable).
- [destabilize], page 106 (macro).
- [finalize], page 112 (function).
- [freeze], page 112 (function).
- [new-funcoid-definition], page 118 (function).
- [new-generic-definition], page 118 (function).
- [resolve-dependency-specification], page 123 (function).
- [stabilize], page 139 (generic function).
- [stabilize-clos-classoid-slot], page 124 (function).
- [stabilize-clos-structure-slot], page 124 (function).

4.1.26 net.didierverna.declt.assess/src/assess.lisp

Dependency
[finalize.lisp], page 27 (file).

Source [net.didierverna.declt.assess.asd], page 11.

Parent Component
[src], page 10 (module).

Public Interface

- [assess], page 42 (function).
- [conclusion], page 50 (reader method).
- [(setf conclusion)], page 50 (writer method).
- [contacts], page 50 (reader method).
- [(setf contacts)], page 50 (writer method).
- [copyright-years], page 51 (reader method).
- [(setf copyright-years)], page 51 (writer method).
- [definitions], page 53 (reader method).
- [(setf definitions)], page 53 (writer method).

Internals

5 Packages

Packages are listed by definition order.

5.1 net.didierverna.declt

The Declt library's package.

Source [package.lisp], page 13.

Nickname declt

Use List

- common-lisp.
- [net.didierverna.declt.assess], page 32.
- [net.didierverna.declt.setup], page 31.

Public Interface

- [declt], page 44 (function).
- [nickname-package], page 46 (function).

Internals

- [%deffn], page 107 (function).
- [*blanks*], page 104 (special variable).
- [*categories*], page 104 (special variable).
- [*fragile-characters*], page 105 (special variable).
- [*section-names*], page 105 (special variable).
- [*special-characters*], page 105 (special variable).
- [@anchor], page 107 (function).
- [@deffn], page 107 (function).
- [@deffnx], page 108 (function).
- [@deftp], page 108 (function).
- [@defvr], page 108 (function).
- [@end], page 108 (function).
- [@item], page 108 (function).
- [@itemize], page 108 (function).
- [@multitable], page 108 (function).
- [@ref], page 109 (function).
- [@table], page 109 (function).
- [add-categories-node], page 109 (function).
- [add-category-node], page 109 (function).
- [add-child], page 109 (function).
- [add-definitions-node], page 109 (function).
- [add-files-node], page 109 (function).
- [add-modules-node], page 109 (function).
- [add-packages-node], page 109 (function).
- [add-systems-node], page 110 (function).
- [anchor], page 110 (function).

- [(setf node-name)], page 119 (writer).
- [node-next], page 119 (reader).
- [(setf node-next)], page 119 (writer).
- [node-p], page 119 (function).
- [node-previous], page 119 (reader).
- [(setf node-previous)], page 119 (writer).
- [node-section-name], page 119 (reader).
- [(setf node-section-name)], page 119 (writer).
- [node-section-type], page 120 (reader).
- [(setf node-section-type)], page 120 (writer).
- [node-synopsis], page 120 (reader).
- [(setf node-synopsis)], page 120 (writer).
- [node-up], page 120 (reader).
- [(setf node-up)], page 120 (writer).
- [read-next-line], page 121 (function).
- [reference], page 121 (function).
- [render-definition-core], page 121 (function).
- [render-dependencies], page 121 (function).
- [render-dependency], page 121 (function).
- [render-docstring], page 122 (function).
- [render-header], page 122 (function).
- [render-headline], page 122 (function).
- [render-node], page 122 (function).
- [render-package-reference], page 122 (function).
- [render-references], page 122 (function).
- [render-text], page 122 (function).
- [render-to-string], page 107 (macro).
- [render-top-node], page 122 (function).
- [reveal], page 123 (function).
- [safe-lambda-list], page 123 (function).
- [safe-name], page 137 (generic function).
- [safe-specializers], page 124 (function).
- [select-keys], page 124 (function).
- [table], page 107 (macro).

5.2 net.didierverna.declt.setup

The Declt setup library's package.

Source [package.lisp], page 11.

Use List common-lisp.

Used By List

- [net.didierverna.declt], page 29.
- [net.didierverna.declt.assess], page 32.

Public Interface

- [*copyright-years*], page 41 (special variable).
- [*release-major-level*], page 41 (special variable).
- [*release-minor-level*], page 41 (special variable).
- [*release-name*], page 41 (special variable).
- [*release-status*], page 41 (special variable).
- [*release-status-level*], page 41 (special variable).
- [abstract-class], page 74 (class).
- [configuration], page 43 (function).
- [configure], page 43 (function).
- [declare-valid-superclass], page 41 (macro).
- [defabstract], page 42 (macro).
- [endpush], page 42 (macro).
- [find*], page 45 (function).
- [mapcat], page 46 (function).
- [non-empty-string], page 104 (type).
- [non-empty-string-p], page 47 (function).
- [retain], page 47 (function).
- [version], page 48 (function).
- [when-let], page 42 (macro).
- [when-let*], page 42 (macro).
- [while], page 42 (macro).

Internals

- [%version], page 107 (function).
- [*configuration*], page 105 (special variable).
- [clindent], page 110 (function).
- [defindent], page 106 (macro).
- [i-reader], page 112 (function).
- [release-status-number], page 121 (function).

5.3 net.didierverna.declt.assess

The Declt assessment library's package.

Source [package.lisp], page 19.

Use List

- common-lisp.
- [net.didierverna.declt.setup], page 31.

Used By List

[net.didierverna.declt], page 29.

Public Interface

- [accessor-method-definition], page 74 (class).
- [accessor-mixin], page 74 (class).
- [alias-definition], page 75 (class).

Internals

6 Definitions

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

6.1 Public Interface

6.1.1 Special variables

`*copyright-years*` [Special Variable]
A string denoting the copyright years for the whole project.

Package [`net.didierverna.declt.setup`], page 31.

Source [`version.lisp`], page 12.

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

Package [`net.didierverna.declt.setup`], page 31.

Source [`version.lisp`], page 12.

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

Package [`net.didierverna.declt.setup`], page 31.

Source [`version.lisp`], page 12.

`*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 31.

Source [`version.lisp`], page 12.

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

Package [`net.didierverna.declt.setup`], page 31.

Source [`version.lisp`], page 12.

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

Package [`net.didierverna.declt.setup`], page 31.

Source [`version.lisp`], page 12.

6.1.2 Macros

`declare-valid-superclass` (*class superclass*) [Macro]
Validate SUPERCLASS classes for CLASS classes.

Package [`net.didierverna.declt.setup`], page 31.

Source [`util.lisp`], page 12.

`defabstract` (*class super-classes slots* &rest *options*) [Macro]
Like DEFCLASS, but define an abstract class.

Package [`net.didierverna.declt.setup`], page 31.

Source [`util.lisp`], page 12.

`endpush` (*object place*) [Macro]
Push OBJECT at the end of PLACE.

Package [`net.didierverna.declt.setup`], page 31.

Source [`util.lisp`], page 12.

`when-let` (*bindings* &body *body*) [Macro]
Execute BODY only when all BINDINGS are non-nil.
BINDINGS must be either a single binding of the form (VARIABLE VALUE), or a list of such. VALUEs are computed sequentially in the specified order, and then VARIABLEs are bound to the corresponding VALUEs. If all VALUEs are non-nil, BODY is executed.

Package [`net.didierverna.declt.setup`], page 31.

Source [`util.lisp`], page 12.

`when-let*` (*bindings* &body *body*) [Macro]
Execute BODY only when all BINDINGS are non-nil.
BINDINGS must be either a single binding of the form (VARIABLE VALUE),
or a list of such. VARIABLEs are bound to their respective VALUE sequentially, so that each VALUE expression may refer to a previously bound VARIABLE. Execution stops completely as soon as a null VALUE is encountered. Otherwise, BODY is executed as an implicit PROGN.

Package [`net.didierverna.declt.setup`], page 31.

Source [`util.lisp`], page 12.

`while` (*test* &body *body*) [Macro]
Execute BODY while TEST.

Package [`net.didierverna.declt.setup`], page 31.

Source [`util.lisp`], page 12.

6.1.3 Ordinary functions

`allocation` (*definition*) [Function]
Return CLOS slot DEFINITION's allocation type.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`assess` (*system-name* &key *introspection-level library-name tagline library-version contact copyright-years license introduction conclusion*) [Function]
Extract and return documentation information for ASDF SYSTEM-NAME.
The documentation information is returned in a REPORT structure, which see.

SYSTEM-NAME is an ASDF system designator. The following keyword parameters allow to specify or override some bits of information.
- INTROSPECTION-LEVEL: how hard to introspect the Lisp environment. At level 1 (the default), scan only the symbols from domestic packages. At level 2, scan all accessible symbols

in the Lisp environment. Some additional information may be discovered in the process, at the expense of a much higher computation time.
- LIBRARY-NAME: name of the library being documented. Defaults to the system name.
- TAGLINE: small text to be used as the manual's subtitle, or NIL. Defaults to the system long name or description.
- LIBRARY-VERSION: version information, or NIL.
Defaults to the system version.
- CONTACT: contact information, or NIL. The default value is computed from the system maintainer(s), author(s), and mailto information. Accepts a contact string, or a list of such. See 'parse-contact-string' for more information. - COPYRIGHT-YEARS: copyright years information or NIL. Defaults to the current year.
- LICENSE: license information. Defaults to NIL. Also accepts :mit, :boost, :bsd, :gpl, :lgpl, and :ms-pl.
- INTRODUCTION: introduction chapter contents in Texinfo format.
Defaults to NIL.
- CONCLUSION: conclusion chapter contents in Texinfo format.
Defaults to NIL.

Package [net.didierverna.declt.assess], page 32.

Source [assess.lisp], page 27.

`bug-tracker` (*definition*) [Function]
Return system DEFINITION's bug tracker, or NIL.

Package [net.didierverna.declt.assess], page 32.

Source [asdf.lisp], page 25.

`combination-options` (*definition*) [Function]
Return generic function DEFINITION's method combination options.

Package [net.didierverna.declt.assess], page 32.

Source [symbol.lisp], page 20.

`component-definition-p` (*definition*) [Function]
Return T if DEFINITION is a component definition.

Package [net.didierverna.declt.assess], page 32.

Source [asdf.lisp], page 25.

`configuration` (*key*) [Function]
Return KEY's value in the current Declt configuration.

Package [net.didierverna.declt.setup], page 31.

Source [configuration.lisp], page 11.

`configure` (*key value*) [Function]
Set KEY to VALUE in the current Declt configuration.

Package [net.didierverna.declt.setup], page 31.

Source [configuration.lisp], page 11.

`declt` (*system-name* &rest *keys* &key *introspection-level library-name tagline library-version contact copyright-years license introduction conclusion locations default-values foreign-definitions declt-notice output-directory file-name info-name info-category*) [Function]

Generate a reference manual for ASDF SYSTEM-NAME.
The reference manual is currently generated in Texinfo format.

For a description of SYSTEM-NAME, INTROSPECTION-LEVEL, LIBRARY-NAME, TAGLINE, LIBRARY-VERSION, CONTACT, COPYRIGHT-YEARS, LICENSE, INTRODUCTION, and CONCLUSION, see 'assess'.

For a description of LOCATIONS, DEFAULT-VALUES, FOREIGN-DEFINITIONS, and DECLT-NOTICE, see 'make-context'.

The following keyword parameters are also available.
- OUTPUT-DIRECTORY: output directory for the generated reference manual. Defaults to the current directory.
- FILE-NAME: base name for the generated reference manual, sans extension. Defaults to the system name.
- INFO-NAME: base name for the subsequent Info file, sans extension (this name appears in the Texinfo file). Defaults to FILE-NAME.
- INFO-CATEGORY: category under which to install the Info file (technically, this provides the value for Texinfo's @dircategory command).
Defaults to `"Common Lisp"`.

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

Source [`declt.lisp`], page 19.

`definition-version` (*definition*) [Function]

Return component DEFINITION's version string.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

`description` (*definition*) [Function]

Return component DEFINITION's description.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

`direct-default-initargs` (*definition*) [Function]

Return CLOS classoid mixin DEFINITION's direct default initargs.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`extension` (*definition*) [Function]

Return file DEFINITION's file extension, if any.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

`file-definition-p` (*definition*) [Function]
Return T if DEFINITION is a file definition.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

`find*` (*object list* &key *test key pre-test*) [Function]
Return the first finding of OBJECT in LIST, or NIL.
Each item in LIST is TESTed with EQ by default. TEST is performed on the item itself by default, or on the result of applying KEY to it. Optionally, only items satisfying PRE-TEST are considered.

Package [`net.didierverna.declt.setup`], page 31.

Source [`util.lisp`], page 12.

`homepage` (*definition*) [Function]
Return system DEFINITION's homepage, or NIL.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

`identity-with-one-argument` (*definition*) [Function]
Return short combination DEFINITION's :identity-with-one-argument option.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`if-feature` (*definition*) [Function]
Return component DEFINITION's if-feature.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

`initargs` (*definition*) [Function]
Return CLOS slot DEFINITION's initargs.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`initform` (*definition*) [Function]
Return CLOS slot DEFINITION's initform.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`license-name` (*definition*) [Function]
Return system DEFINITION's license name, or NIL.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

`lisp-file-definition-p` (*definition*) [Function]
Return T if DEFINITION is a Lisp file definition.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

`long-description` (*definition*) [Function]
Return component DEFINITION's long description.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

`long-name` (*definition*) [Function]
Return system DEFINITION's long name, or NIL.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

`mailto` (*definition*) [Function]
Return system DEFINITION's mailto, or NIL.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

`mapcat` (*function* &rest *lists*) [Function]
Short for "mapconcat": non destructive version of MAPCAN. That is, concatenate the results with APPEND rather than NCONC.

Package [`net.didierverna.declt.setup`], page 31.

Source [`util.lisp`], page 12.

`method-definition-p` (*definition*) [Function]
Return T if DEFINITION is a method definition.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`module-definition-p` (*definition*) [Function]
Return T if DEFINITION is a module definition.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

`nickname-package` (&optional *nickname*) [Function]
Add NICKNAME (:DECLT by default) to the :NET.DIDIERVERNA.DECLT package.

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

Source [`package.lisp`], page 13.

`nickname-package` (&optional *nickname*) [Function]
Add NICKNAME (:ASSESS by default) to the :NET.DIDIERVERNA.DECLT.ASSESS package.

Package [`net.didierverna.declt.assess`], page 32.

Source [`package.lisp`], page 19.

`nicknames` (*package-definition*) [Function]
Return the list of nicknames for PACKAGE-DEFINITION.

Package [`net.didierverna.declt.assess`], page 32.

Source [`package.lisp`], page 24.

`non-empty-string-p` (*object*) [Function]
Return T if OBJECT is a non-empty string.

Package [`net.didierverna.declt.setup`], page 31.

Source [`util.lisp`], page 12.

`package-definition-p` (*definition*) [Function]
Return T if DEFINITION is a package definition.

Package [`net.didierverna.declt.assess`], page 32.

Source [`package.lisp`], page 24.

`publicp` (*definition*) [Function]
Return T is DEFINITION is public.
A definition is public when the symbol naming it has a home package, and is exported from it.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`qualifiers` (*definition*) [Function]
Return method DEFINITION's method qualifiers.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`reader-method-definition-p` (*definition*) [Function]
Return T if DEFINITION is a reader method definition.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`retain` (*object list* &key *test key pre-test*) [Function]
Return a copy of LIST from which only OBJECT is retained.
Each item in LIST is TESTed with EQ by default. TEST is performed on the item itself by default, or on the result of applying KEY to it. Optionally, only items satisfying PRE-TEST are considered.

Package [`net.didierverna.declt.setup`], page 31.

Source [`util.lisp`], page 12.

`short-expander-definition-p` (*definition*) [Function]
Return T if DEFINITION is a short expander definition.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`source-control` (*definition*) [Function]
Return system DEFINITION's source control, or NIL.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

`symbol-definition-p` (*definition*) [Function]
Return T if DEFINITION is a symbol definition.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`system-definition-p` (*definition*) [Function]
Return T if DEFINITION is a system definition.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

`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 31.

Source [`version.lisp`], page 12.

`writer-method-definition-p` (*definition*) [Function]
Return T if DEFINITION is a writer method definition.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

6.1.4 Generic functions

`authors` (*object*) [Generic Reader]
`(setf authors)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`authors` ((*system-definition [`system-definition`], page 100*)) [Reader Method]
`(setf authors)` ((*system-definition [`system-definition`], page 100*)) [Writer Method]
The list of parsed author contacts.
See 'parse-contact-string' for more information.

Source [`asdf.lisp`], page 25.

Target Slot
[`authors`], page 101.

`children` (*object*) [Generic Reader]
`(setf children)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`children` ((*module-definition* [`module-definition`], page 90)) [Reader Method]
`(setf children)` ((*module-definition* [`module-definition`], page 90)) [Writer Method]

The list of child definitions for this definition's module.

Source [`asdf.lisp`], page 25.

Target Slot
[`children`], page 91.

`classoid` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`classoid` ((*classoid-definition* [`classoid-definition`], page 76)) [Reader Method]

automatically generated reader method

Source [`symbol.lisp`], page 20.

Target Slot
[`object`], page 77.

`clients` (*object*) [Generic Reader]
`(setf clients)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`clients` ((*combination-definition* [`combination-definition`], page 79)) [Reader Method]
`(setf clients)` ((*combination-definition* [`combination-definition`], page 79)) [Writer Method]

The list of client definitions for this definition's method combnination. These are generic functions using this combination.

Source [`symbol.lisp`], page 20.

Target Slot
[`clients`], page 79.

`combination` (*object*) [Generic Reader]
`(setf combination)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`combination` ((*generic-function-definition* [`generic-function-definition`], page 86)) [Reader Method]
`(setf combination)` ((*generic-function-definition* [`generic-function-definition`], page 86)) [Writer Method]

The method combination definition for this definition's generic function.

Source [`symbol.lisp`], page 20.

Target Slot
[combination], page 87.

`combination` ((*combination-definition* *[combination-definition], page 79*)) [Reader Method]
automatically generated reader method

Source [symbol.lisp], page 20.

Target Slot
[object], page 79.

`component` (*object*) [Generic Reader]

Package [net.didierverna.declt.assess], page 32.

Methods

`component` ((*component-definition* *[component-definition], page 80*)) [Reader Method]
automatically generated reader method

Source [asdf.lisp], page 25.

Target Slot
[object], page 81.

`conclusion` (*object*) [Generic Reader]
`(setf conclusion)` (*object*) [Generic Writer]

Package [net.didierverna.declt.assess], page 32.

Methods

`conclusion` ((*report [report], page 94*)) [Reader Method]
`(setf conclusion)` ((*report [report], page 94*)) [Writer Method]
Contents for a conclusion chapter.

Source [assess.lisp], page 27.

Target Slot
[conclusion], page 95.

`contacts` (*object*) [Generic Reader]
`(setf contacts)` (*object*) [Generic Writer]

Package [net.didierverna.declt.assess], page 32.

Methods

`contacts` ((*report [report], page 94*)) [Reader Method]
`(setf contacts)` ((*report [report], page 94*)) [Writer Method]
The list of contacts for the library.
Each element is of the form (NAME . EMAIL) where both NAME and EMAIL are strings or NIL, and cannot be null at the same time.

Source [assess.lisp], page 27.

Target Slot
[contacts], page 95.

`copyright-years` (*object*) [Generic Reader]
`(setf copyright-years)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`copyright-years` ((*report* [`report`], *page 94*)) [Reader Method]
`(setf copyright-years)` ((*report* [`report`], *page 94*)) [Writer Method]

A copyright years string.

Source [`assess.lisp`], page 27.

Target Slot
[`copyright-years`], page 95.

`definition-class` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`definition-class` ((*class-definition* [`class-definition`], *page 76*)) [Reader Method]

automatically generated reader method

Source [`symbol.lisp`], page 20.

Target Slot
[`object`], page 76.

`definition-compiler-macro` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`definition-compiler-macro` ((*compiler-macro-definition* [`compiler-macro-definition`], *page 80*)) [Reader Method]

automatically generated reader method

Source [`symbol.lisp`], page 20.

Target Slot
[`object`], page 80.

`definition-condition` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`definition-condition` ((*condition-definition* [`condition-definition`], *page 81*)) [Reader Method]

automatically generated reader method

Source [`symbol.lisp`], page 20.

Target Slot
[`object`], page 82.

`definition-function` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`definition-function` ((*function-definition* *[`function-definition`], page 86*)) [Reader Method]

automatically generated reader method

Source [`symbol.lisp`], page 20.

Target Slot

[`object`], page 86.

`definition-method` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`definition-method` ((*method-definition* *[`method-definition`], page 90*)) [Reader Method]

automatically generated reader method

Source [`symbol.lisp`], page 20.

Target Slot

[`object`], page 90.

`definition-package` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`definition-package` ((*package-definition* *[`package-definition`], page 92*)) [Reader Method]

automatically generated reader method

Source [`package.lisp`], page 24.

Target Slot

[`object`], page 93.

`definition-structure` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`definition-structure` ((*structure-definition* *[`structure-definition`], page 99*)) [Reader Method]

automatically generated reader method

Source [`symbol.lisp`], page 20.

Target Slot

[`object`], page 99.

`definition-symbol` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`definition-symbol` ((*symbol-definition* *[`symbol-definition`], page 99*)) [Reader Method]

The symbol naming this definition.

Source [`symbol.lisp`], page 20.

Target Slot
[`symbol`], page 100.

`definitions` (*object*) [Generic Reader]
`(setf definitions)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`definitions` ((*report* *[`report`], page 94*)) [Reader Method]
`(setf definitions)` ((*report* *[`report`], page 94*)) [Writer Method]

The list of definitions.

Source [`assess.lisp`], page 27.

Target Slot
[`definitions`], page 95.

`definitions` ((*lisp-file-definition* *[`lisp-file-definition`], page 88*)) [Reader Method]
`(setf definitions)` ((*lisp-file-definition* *[`lisp-file-definition`], page 88*)) [Writer Method]

The list of definitions for this definition's file.

Source [`asdf.lisp`], page 25.

Target Slot
[`definitions`], page 88.

`definitions` ((*package-definition* *[`package-definition`], page 92*)) [Reader Method]
`(setf definitions)` ((*package-definition* *[`package-definition`], page 92*)) [Writer Method]

The list of corresponding definitions.

Source [`package.lisp`], page 24.

Target Slot
[`definitions`], page 93.

`defsystem-dependencies` (*object*) [Generic Reader]
`(setf defsystem-dependencies)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`defsystem-dependencies` ((*system-definition* *[`system-definition`], page 100*)) [Reader Method]
`(setf defsystem-dependencies)` ((*system-definition* *[`system-definition`], page 100*)) [Writer Method]

The list of defsystem dependency definitions.

Source [`asdf.lisp`], page 25.

Target Slot
[`defsystem-dependencies`], page 101.

`dependencies` (*object*) [Generic Reader]
`(setf dependencies)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`dependencies` ((*component-definition* [`component-definition`], *page 80*)) [Reader Method]
`(setf dependencies)` ((*component-definition* [`component-definition`], *page 80*)) [Writer Method]

The list of dependency definitions for this definition's component.

Source [`asdf.lisp`], page 25.

Target Slot
[`dependencies`], page 81.

`direct-methods` (*object*) [Generic Reader]
`(setf direct-methods)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`direct-methods` ((*clos-classoid-mixin* [`clos-classoid-mixin`], *page 77*)) [Reader Method]
`(setf direct-methods)` ((*clos-classoid-mixin* [`clos-classoid-mixin`], *page 77*)) [Writer Method]

The list of direct method definitions for this definition's classoid.

Source [`symbol.lisp`], page 20.

Target Slot
[`direct-methods`], page 78.

`direct-slots` (*object*) [Generic Reader]
`(setf direct-slots)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`direct-slots` ((*classoid-definition* [`classoid-definition`], *page 76*)) [Reader Method]
`(setf direct-slots)` ((*classoid-definition* [`classoid-definition`], *page 76*)) [Writer Method]

The list of direct slot definitions for this definition's classoid.

Source [`symbol.lisp`], page 20.

Target Slot
[`direct-slots`], page 77.

`direct-subclasses` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`direct-subclasses` ((*class-definition* [`class-definition`], *page 76*)) [Reader Method]

automatically generated reader method

Source [`symbol.lisp`], page 20.

Target Slot
[direct-subclassoids], page 76.

`(setf direct-subclasses)` (*object*) [Generic Writer]

Package [net.didierverna.declt.assess], page 32.

Methods

`(setf direct-subclasses)` ((*class-definition* *[class-definition], page 76*)) [Writer Method]
automatically generated writer method

Source [symbol.lisp], page 20.

Target Slot
[direct-subclassoids], page 76.

`direct-subclassoids` (*object*) [Generic Reader]
`(setf direct-subclassoids)` (*object*) [Generic Writer]

Package [net.didierverna.declt.assess], page 32.

Methods

`direct-subclassoids` ((*clos-classoid-mixin* *[clos-classoid-mixin], page 77*)) [Reader Method]
`(setf direct-subclassoids)` ((*clos-classoid-mixin* *[clos-classoid-mixin], page 77*)) [Writer Method]
The list of direct subclassoid definitions for this definition's classoid.

Source [symbol.lisp], page 20.

Target Slot
[direct-subclassoids], page 78.

`direct-subconditions` (*object*) [Generic Reader]

Package [net.didierverna.declt.assess], page 32.

Methods

`direct-subconditions` ((*condition-definition* *[condition-definition], page 81*)) [Reader Method]
automatically generated reader method

Source [symbol.lisp], page 20.

Target Slot
[direct-subclassoids], page 82.

`(setf direct-subconditions)` (*object*) [Generic Writer]

Package [net.didierverna.declt.assess], page 32.

Methods

`(setf direct-subconditions)` ((*condition-definition* *[condition-definition], page 81*)) [Writer Method]
automatically generated writer method

Source [symbol.lisp], page 20.

Target Slot
[direct-subclassoids], page 82.

`direct-substructures` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`direct-substructures` ((*clos-structure-definition* *[`clos-structure-definition`], page 78*)) [Reader Method]

automatically generated reader method

Source [`symbol.lisp`], page 20.

Target Slot

[`direct-subclassoids`], page 79.

`(setf direct-substructures)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`(setf direct-substructures)` ((*clos-structure-definition* *[`clos-structure-definition`], page 78*)) [Writer Method]

automatically generated writer method

Source [`symbol.lisp`], page 20.

Target Slot

[`direct-subclassoids`], page 79.

`direct-superclasses` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`direct-superclasses` ((*class-definition* *[`class-definition`], page 76*)) [Reader Method]

automatically generated reader method

Source [`symbol.lisp`], page 20.

Target Slot

[`direct-superclassoids`], page 76.

`(setf direct-superclasses)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`(setf direct-superclasses)` ((*class-definition* *[`class-definition`], page 76*)) [Writer Method]

automatically generated writer method

Source [`symbol.lisp`], page 20.

Target Slot

[`direct-superclassoids`], page 76.

`direct-superclassoids` (*object*) [Generic Reader]
`(setf direct-superclassoids)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`direct-superclassoids` ((*clos-classoid-mixin* [`clos-classoid-mixin`], *page 77*)) [Reader Method]

`(setf direct-superclassoids)` ((*clos-classoid-mixin* [`clos-classoid-mixin`], *page 77*)) [Writer Method]

The list of direct superclassoid definitions for this definition's classoid.

Source [`symbol.lisp`], page 20.

Target Slot [`direct-superclassoids`], page 78.

`direct-superconditions` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`direct-superconditions` ((*condition-definition* [`condition-definition`], *page 81*)) [Reader Method]

automatically generated reader method

Source [`symbol.lisp`], page 20.

Target Slot [`direct-superclassoids`], page 82.

`(setf direct-superconditions)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`(setf direct-superconditions)` ((*condition-definition* [`condition-definition`], *page 81*)) [Writer Method]

automatically generated writer method

Source [`symbol.lisp`], page 20.

Target Slot [`direct-superclassoids`], page 82.

`direct-superstructures` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`direct-superstructures` ((*clos-structure-definition* [`clos-structure-definition`], *page 78*)) [Reader Method]

automatically generated reader method

Source [`symbol.lisp`], page 20.

Target Slot [`direct-superclassoids`], page 79.

`(setf direct-superstructures)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`(setf direct-superstructures)` [Writer Method]
((*clos-structure-definition* *[*`clos-structure-definition`*], page 78*))
automatically generated writer method

Source [`symbol.lisp`], page 20.

Target Slot
[`direct-superclassoids`], page 79.

`docstring` (*definition*) [Generic Function]
Return DEFINITION's docstring (Lisp documentation).

Package [`net.didierverna.declt.assess`], page 32.

Source [`definition.lisp`], page 19.

Methods

`docstring` ((*definition* *[*`component-definition`*], page 80*)) [Method]
Return component DEFINITION's description. This is the same as the 'description' function.

Source [`asdf.lisp`], page 25.

`docstring` ((*definition* *[*`function-alias-definition`*], page 86*)) [Method]
Return function alias DEFINITION's docstring. This is the docstring attached to DEFINITION's name, rather than the one attached to the function.

Source [`symbol.lisp`], page 20.

`docstring` ((*definition* *[*`compiler-macro-alias-definition`*], page 79*)) [Method]
Return compiler macro alias DEFINITION's docstring.
This is the docstring attached to DEFINITION's name, rather than the one attached to the compiler macro function.

Source [`symbol.lisp`], page 20.

`docstring` ((*definition* *[*`macro-alias-definition`*], page 89*)) [Method]
Return macro alias DEFINITION's docstring.
This is the docstring attached to DEFINITION's symbol, rather than the one attached to the macro function.

Source [`symbol.lisp`], page 20.

`docstring` ((*definition* *[*`typed-structure-slot-definition`*], page 103*)) [Method]
Return NIL.

Source [`symbol.lisp`], page 20.

`docstring` ((*definition* *[*`clos-slot-definition`*], page 78*)) [Method]
Return CLOS slot DEFINITION's docstring.

Source [`symbol.lisp`], page 20.

`docstring` ((*definition* *[*`typed-structure-definition`*]*, *page 102*)) [Method]
Return typed structure DEFINITION's docstring.

Source [`symbol.lisp`], page 20.

`docstring` ((*definition* *[*`combination-definition`*]*, *page 79*)) [Method]
Return method combination DEFINITION's docstring.

Source [`symbol.lisp`], page 20.

`docstring` ((*definition* *[*`expander-definition`*]*, *page 83*)) [Method]
Return setf expander DEFINITION's docstring.

Source [`symbol.lisp`], page 20.

`docstring` ((*definition* *[*`type-definition`*]*, *page 102*)) [Method]
Return type DEFINITION's docstring.

Source [`symbol.lisp`], page 20.

`docstring` ((*definition* *[*`symbol-macro-definition`*]*, *page 100*)) [Method]
Return NIL (symbol macros don't have a docstring).

Source [`symbol.lisp`], page 20.

`docstring` ((*definition* *[*`variable-definition`*]*, *page 103*)) [Method]
Return variable DEFINITION's docstring.

Source [`symbol.lisp`], page 20.

`docstring` (*definition*) [Method]
Return DEFINITION's object canonical documentation. This is the default method.

`element-type` (*object*) [Generic Reader]
`(setf element-type)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`element-type` ((*typed-structure-definition* *[*`typed-structure-definition`*]*, *page 102*)) [Reader Method]
`(setf element-type)` ((*typed-structure-definition* *[*`typed-structure-definition`*]*, *page 102*)) [Writer Method]
The structure's element type.
It is T for list structures, but may be something else for vector ones.

Source [`symbol.lisp`], page 20.

Target Slot
[`element-type`], page 103.

`expander` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`expander` ((*expander-definition* *[`expander-definition`], page 83*)) [Reader Method]
automatically generated reader method

Source [`symbol.lisp`], page 20.

Target Slot
[`object`], page 84.

`expander` ((*type-definition* *[`type-definition`], page 102*)) [Reader Method]
automatically generated reader method

Source [`symbol.lisp`], page 20.

Target Slot
[`object`], page 102.

`expander-for` (*object*) [Generic Reader]
`(setf expander-for)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`expander-for` ((*setfable-funcoid-definition* *[`setfable-funcoid-definition`], page 95*)) [Reader Method]
`(setf expander-for)` ((*setfable-funcoid-definition* *[`setfable-funcoid-definition`], page 95*)) [Writer Method]
A setf expander definition for this funcoid, or NIL.
This is the definition of a setf expander that expands forms identical to this funcoid's signature. There can be only one. Note that the Common Lisp standard does not impose any actual relation between the setf expander and its access-fn. In fact, the access-fn may not even exist at all. However, if it does, it is very likely that it is a reader for the place updated by this setf expander.

Source [`symbol.lisp`], page 20.

Target Slot
[`expander-for`], page 96.

`expanders-to` (*object*) [Generic Reader]
`(setf expanders-to)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`expanders-to` ((*setfable-funcoid-definition* *[`setfable-funcoid-definition`], page 95*)) [Reader Method]
`(setf expanders-to)` ((*setfable-funcoid-definition* *[`setfable-funcoid-definition`], page 95*)) [Writer Method]
The list of setf expander definitions to this funcoid.
This is a list of definitions for short form setf expanders that have this funcoid as their update-fn. There might be more than one.

Source [`symbol.lisp`], page 20.

Target Slot
[`expanders-to`], page 96.

`file` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`file` ((*file-definition* [`file-definition`], *page 84*)) [Reader Method]
automatically generated reader method

Source [`asdf.lisp`], page 25.

Target Slot
[`object`], page 85.

`foreignp` (*object*) [Generic Reader]
`(setf foreignp)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`foreignp` ((*definition* [`definition`], *page 82*)) [Reader Method]
`(setf foreignp)` ((*definition* [`definition`], *page 82*)) [Writer Method]
Whether this definition is foreign.

Source [`definition.lisp`], page 19.

Target Slot
[`foreign`], page 83.

`funcoid` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`funcoid` ((*funcoid-definition* [`funcoid-definition`], *page 85*)) [Reader Method]
automatically generated reader method

Source [`symbol.lisp`], page 20.

Target Slot
[`object`], page 85.

`generic` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`generic` ((*generic-function-definition* [`generic-function-definition`], *page 86*)) [Reader Method]
automatically generated reader method

Source [`symbol.lisp`], page 20.

Target Slot
[`object`], page 87.

`home-package` (*object*) [Generic Reader]
`(setf home-package)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`home-package` ((*symbol-definition* *[`symbol-definition`], page 99*)) [Reader Method]
`(setf home-package)` ((*symbol-definition* *[`symbol-definition`], page 99*)) [Writer Method]
The home package definition for this definition's symbol.
Every definition gets a home package, even foreign ones. A home package can only be null when the definition's symbol is uninterned.

Source [`symbol.lisp`], page 20.

Target Slot
[`home-package`], page 100.

`introduction` (*object*) [Generic Reader]
`(setf introduction)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`introduction` ((*report* *[`report`], page 94*)) [Reader Method]
`(setf introduction)` ((*report* *[`report`], page 94*)) [Writer Method]
Contents for an introduction chapter.

Source [`assess.lisp`], page 27.

Target Slot
[`introduction`], page 95.

`lambda-list` (*definition*) [Generic Function]
Return funcoid DEFINITION's lambda-list.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Methods

`lambda-list` ((*definition* *[`alias-definition`], page 75*)) [Method]
Return the lambda-list of alias DEFINITION's referee.

`lambda-list` ((*definition* *[`method-definition`], page 90*)) [Method]
Return method DEFINITION's method lambda-list.

`lambda-list` ((*definition* *[`combination-definition`], page 79*)) [Method]
Return method combination DEFINITION's lambda-list.

`lambda-list` ((*definition* *[`long-expander-definition`], page 89*)) [Method]
Return long setf expander DEFINITION's expander function's lambda-list.

`lambda-list` ((*definition* *[`short-expander-definition`], page 97*)) [Method]
Return short setf expander DEFINITION's lambda-list.
This lambda-list is computed as the shortened version of DEFINITION's update-fn lambda-list, because setf expanders pass the new value as the last argument to their operator.
If the expander's update-fn is not defined, return two values: NIL and T.

`lambda-list` ((*definition* [`type-definition`], *page 102*)) [Method]
Return type DEFINITION's type lambda-list.

`lambda-list` ((*definition* [`funcoid-definition`], *page 85*)) [Method]
Return funcoid DEFINITION's function lambda-list. This is the default method.

`lambda-list` *:around* ((*definition* [`funcoid-definition`], *page 85*)) [Method]
Return only the lambda-list's CDR for setf definitions.
This only applies to compiler macros and functions (to filter out the parameter corresponding to the new value) but does nothing on setf expanders because their primary methods already do the filtering (differently).

`library-name` (*object*) [Generic Reader]
`(setf library-name)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`library-name` ((*report* [`report`], *page 94*)) [Reader Method]
`(setf library-name)` ((*report* [`report`], *page 94*)) [Writer Method]
The library's name.

Source [`assess.lisp`], page 27.

Target Slot
[`library-name`], page 94.

`library-version` (*object*) [Generic Reader]
`(setf library-version)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`library-version` ((*report* [`report`], *page 94*)) [Reader Method]
`(setf library-version)` ((*report* [`report`], *page 94*)) [Writer Method]
The library's version.

Source [`assess.lisp`], page 27.

Target Slot
[`library-version`], page 95.

`license` (*object*) [Generic Reader]
`(setf license)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`license` ((*report* [`report`], *page 94*)) [Reader Method]
`(setf license)` ((*report* [`report`], *page 94*)) [Writer Method]
The library's license.

Source [`assess.lisp`], page 27.

Target Slot
[`license`], page 95.

`location` (*object*) [Generic Reader]
`(setf location)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`location` ((*component-definition* *[`component-definition`], page 80*)) [Reader Method]
`(setf location)` ((*component-definition* *[`component-definition`], page 80*)) [Writer Method]

The component's location (a namestring).

Source [`asdf.lisp`], page 25.

Target Slot
[`location`], page 81.

`macro` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`macro` ((*macro-definition* *[`macro-definition`], page 89*)) [Reader Method]

automatically generated reader method

Source [`symbol.lisp`], page 20.

Target Slot
[`object`], page 89.

`maintainers` (*object*) [Generic Reader]
`(setf maintainers)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`maintainers` ((*system-definition* *[`system-definition`], page 100*)) [Reader Method]
`(setf maintainers)` ((*system-definition* *[`system-definition`], page 100*)) [Writer Method]

The list of parsed maintainer contacts.
See 'parse-contact-string' for more information.

Source [`asdf.lisp`], page 25.

Target Slot
[`maintainers`], page 101.

`methods` (*object*) [Generic Reader]
`(setf methods)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`methods` ((*generic-function-definition* *[`generic-function-definition`], page 86*)) [Reader Method]
`(setf methods)` ((*generic-function-definition* *[`generic-function-definition`], page 86*)) [Writer Method]

The list of method definitions for this definition's generic function.

Source [`symbol.lisp`], page 20.

Target Slot
[methods], page 87.

`module` (*object*) [Generic Reader]

Package [net.didierverna.declt.assess], page 32.

Methods

`module` ((*module-definition* [module-definition], page 90)) [Reader Method]
automatically generated reader method

Source [asdf.lisp], page 25.

Target Slot
[object], page 91.

`name` (*definition*) [Generic Function]
The definition's name.
This is the native Lisp name for the definition's corresponding object. It's either a string (for ASDF components and packages), a symbol, or a list of the form (setf symbol).

Package [net.didierverna.declt.assess], page 32.

Source [definition.lisp], page 19.

Methods

`name` ((*definition* [component-definition], page 80)) [Method]
Return component DEFINITION's component name.

Source [asdf.lisp], page 25.

`name` ((*definition* [package-definition], page 92)) [Method]
Return package DEFINITION's package name.

Source [package.lisp], page 24.

`name` *:around* ((*definition* [alias-definition], page 75)) [Method]
Wrap alias DEFINITION's name in a SETF list when appropriate.

Source [symbol.lisp], page 20.

`name` *:around* ((*definition* [funcoid-definition], page 85)) [Method]
Wrap funcoid DEFINITION's name in a SETF list when appropriate.

Source [symbol.lisp], page 20.

`name` ((*definition* [symbol-definition], page 99)) [Method]
Return symbol DEFINITION's symbol.

Source [symbol.lisp], page 20.

`object` (*object*) [Generic Reader]

Package [net.didierverna.declt.assess], page 32.

Methods

`object` ((*definition* [definition], page 82)) [Reader Method]
The corresponding Lisp object, or NIL.
Only constants, special variables, symbol macros, and aliases lack such an object.

Source [definition.lisp], page 19.

Target Slot
[object], page 83.

`owner` (*object*) [Generic Reader]
`(setf owner)` (*object*) [Generic Writer]

Package [net.didierverna.declt.assess], page 32.

Methods

`owner` ((*slot-definition* [slot-definition], page 97)) [Reader Method]
`(setf owner)` ((*slot-definition* [slot-definition], page 97)) [Writer Method]
The definition for the owner of this definition's slot.

Source [symbol.lisp], page 20.

Target Slot
[owner], page 98.

`owner` ((*method-definition* [method-definition], page 90)) [Reader Method]
`(setf owner)` ((*method-definition* [method-definition], page 90)) [Writer Method]
The generic function definition for this definition's method.

Source [symbol.lisp], page 20.

Target Slot
[owner], page 90.

`parent` (*object*) [Generic Reader]
`(setf parent)` (*object*) [Generic Writer]

Package [net.didierverna.declt.assess], page 32.

Methods

`parent` ((*component-definition* [component-definition], page 80)) [Reader Method]
`(setf parent)` ((*component-definition* [component-definition], page 80)) [Writer Method]
The parent definition for this definition's component.

Source [asdf.lisp], page 25.

Target Slot
[parent], page 81.

`private-definitions` (*object*) [Generic Function]
Return OBJECT's private definitions.

Package [net.didierverna.declt.assess], page 32.

Source [definition.lisp], page 19.

Methods

`private-definitions` ((*definition* [package-definition], page 92)) [Method]
Return package DEFINITION's private definitions.

Source [package.lisp], page 24.

`private-definitions` (*object*) [Method]
Return OBJECT's private definitions from its definitions list. This is the default method for heterogeneous definitions lists.

`public-definitions` (*object*) [Generic Function]
Return OBJECT's public definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`definition.lisp`], page 19.

Methods

`public-definitions` ((*definition* [`package-definition`], *page 92*)) [Method]
Return package DEFINITION's public definitions.

Source [`package.lisp`], page 24.

`public-definitions` (*object*) [Method]
Return OBJECT's public definitions from its definitions list. This is the default method for heterogeneous definitions lists.

`readers` (*object*) [Generic Reader]
`(setf readers)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`readers` ((*slot-definition* [`slot-definition`], *page 97*)) [Reader Method]
`(setf readers)` ((*slot-definition* [`slot-definition`], *page 97*)) [Writer Method]
The list of definitions for this definition's slot readers.

Source [`symbol.lisp`], page 20.

Target Slot
[`readers`], page 98.

`referee` (*object*) [Generic Reader]
`(setf referee)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`referee` ((*alias-definition* [`alias-definition`], *page 75*)) [Reader Method]
`(setf referee)` ((*alias-definition* [`alias-definition`], *page 75*)) [Writer Method]
The original definition this definition aliases.

Source [`symbol.lisp`], page 20.

Target Slot
[`referee`], page 75.

`setfp` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`setfp` ((*alias-definition* [`alias-definition`], *page 75*)) [Reader Method]
Whether this is a setf alias definition.

Source [`symbol.lisp`], page 20.

Target Slot
[`setf`], page 75.

`setfp` ((*funcoid-definition* [`funcoid-definition`], *page 85*)) [Reader Method]
Whether this is a setf definition.

Source [`symbol.lisp`], page 20.

Target Slot
[`setf`], page 85.

`slot` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`slot` ((*slot-definition* [`slot-definition`], *page 97*)) [Reader Method]
automatically generated reader method

Source [`symbol.lisp`], page 20.

Target Slot
[`object`], page 98.

`source-file` (*object*) [Generic Reader]
`(setf source-file)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`source-file` ((*definition* [`definition`], *page 82*)) [Reader Method]
`(setf source-file)` ((*definition* [`definition`], *page 82*)) [Writer Method]
The source file definition for this definition's object.

Source [`definition.lisp`], page 19.

Target Slot
[`source-file`], page 83.

`specializers` (*object*) [Generic Reader]
`(setf specializers)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`specializers` ((*method-definition* [`method-definition`], *page 90*)) [Reader Method]
`(setf specializers)` ((*method-definition* [`method-definition`], *page 90*)) [Writer Method]
The specializers of this definition's method.
This is a list of either class definitions (for regular specializers), or raw EQL specializers.

Source [`symbol.lisp`], page 20.

Target Slot
[specializers], page 90.

`standalone-combinator` (*object*) [Generic Reader]
`(setf standalone-combinator)` (*object*) [Generic Writer]

Package [net.didierverna.declt.assess], page 32.

Methods

`standalone-combinator` [Reader Method]
((*short-combination-definition* *[short-combination-definition], page 96*))
`(setf standalone-combinator)` [Writer Method]
((*short-combination-definition* *[short-combination-definition], page 96*))
The corresponding standalone combinator definition, or NIL.
This is a function or macro definition. Note that if this definition is unavailable, it means that the method combination itself cannot be used (yet).

Source [symbol.lisp], page 20.

Target Slot
[standalone-combinator], page 97.

`standalone-reader` (*object*) [Generic Reader]
`(setf standalone-reader)` (*object*) [Generic Writer]

Package [net.didierverna.declt.assess], page 32.

Methods

`standalone-reader` ((*expander-definition* [Reader Method]
[expander-definition], page 83))
`(setf standalone-reader)` ((*expander-definition* [Writer Method]
[expander-definition], page 83))
A standalone reader definition for this definition's expander, or NIL.
If it exists, it's a definition for a function or macro with the same signature as that of the expander's access-fn. Note that the Common Lisp standard does not impose any actual relation between the setf expander and its access-fn. In fact, the access-fn may not even exist at all. However, if it does, it is very likely that it is a reader for the place updated by this setf expander.

Source [symbol.lisp], page 20.

Target Slot
[standalone-reader], page 84.

`standalone-writer` (*object*) [Generic Reader]
`(setf standalone-writer)` (*object*) [Generic Writer]

Package [net.didierverna.declt.assess], page 32.

Methods

`standalone-writer` ((*short-expander-definition [*`short-expander-definition`*], page 97*)) [Reader Method]
`(setf standalone-writer)` ((*short-expander-definition [*`short-expander-definition`*], page 97*)) [Writer Method]

A standalone writer definition for this definition's expander, or NIL. This is a function or macro definition. Note that if this definition is unavailable, it means that the expander itself cannot be used (yet).

Source [`symbol.lisp`], page 20.

Target Slot
[`standalone-writer`], page 97.

`structure-type` (*object*) [Generic Reader]
`(setf structure-type)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`structure-type` ((*typed-structure-definition [*`typed-structure-definition`*], page 102*)) [Reader Method]
`(setf structure-type)` ((*typed-structure-definition [*`typed-structure-definition`*], page 102*)) [Writer Method]

The structure type, either LIST or VECTOR.

Source [`symbol.lisp`], page 20.

Target Slot
[`type`], page 103.

`system` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`system` ((*system-definition [*`system-definition`*], page 100*)) [Reader Method]

automatically generated reader method

Source [`asdf.lisp`], page 25.

Target Slot
[`object`], page 101.

`system-name` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`system-name` ((*report [*`report`*], page 94*)) [Reader Method]

The main system's name, coerced to a string.

Source [`assess.lisp`], page 27.

Target Slot
[`system-name`], page 94.

`tagline` (*object*) [Generic Reader]
`(setf tagline)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`tagline` ((*report* [`report`], *page 94*)) [Reader Method]
`(setf tagline)` ((*report* [`report`], *page 94*)) [Writer Method]
The reference manual's tagline.

Source [`assess.lisp`], page 27.

Target Slot
[`tagline`], page 95.

`target-slot` (*object*) [Generic Reader]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`target-slot` ((*accessor-mixin* [`accessor-mixin`], *page 74*)) [Reader Method]
The target slot definition for this definition's accessor.

Source [`symbol.lisp`], page 20.

Target Slot
[`target-slot`], page 74.

`uid` (*object*) [Generic Reader]
`(setf uid)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`uid` ((*definition* [`definition`], *page 82*)) [Reader Method]
`(setf uid)` ((*definition* [`definition`], *page 82*)) [Writer Method]
This definition's UID.

Source [`definition.lisp`], page 19.

Target Slot
[`uid`], page 83.

`use-list` (*object*) [Generic Reader]
`(setf use-list)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`use-list` ((*package-definition* [`package-definition`], *page 92*)) [Reader Method]
`(setf use-list)` ((*package-definition* [`package-definition`], *page 92*)) [Writer Method]
The definitions use-list for this definition's package.

Source [`package.lisp`], page 24.

Target Slot
[`use-list`], page 93.

`used-by-list` (*object*) [Generic Reader]
`(setf used-by-list)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`used-by-list` ((*package-definition* *[`package-definition`], page 92*)) [Reader Method]
`(setf used-by-list)` ((*package-definition* *[`package-definition`], page 92*)) [Writer Method]
The definitions used-by-list for this definition's package.

Source [`package.lisp`], page 24.

Target Slot
[`used-by-list`], page 93.

`value-type` (*definition*) [Generic Function]
Return slot DEFINITION's value type.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Methods

`value-type` ((*definition* *[`typed-structure-slot-definition`], page 103*)) [Method]
Return typed structure slot DEFINITION's value type.

`value-type` ((*definition [`clos-slot-definition`], page 78*)) [Method]
Return CLOS slot DEFINITION's value type.

`writers` (*object*) [Generic Reader]
`(setf writers)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`writers` ((*slot-definition [`slot-definition`], page 97*)) [Reader Method]
`(setf writers)` ((*slot-definition [`slot-definition`], page 97*)) [Writer Method]
The list of definitions for this definition's slot writers.

Source [`symbol.lisp`], page 20.

Target Slot
[`writers`], page 98.

6.1.5 Standalone methods

`initialize-instance` *:after* ((*definition [`clos-classoid-mixin`], page 77*) &key *packages pathnames*) [Method]
Compute CLOS classoid DEFINITION's slot definitions.

Source [`symbol.lisp`], page 20.

`initialize-instance` *:after* ((*definition* *[*`package-definition`*]*, *page 92*) &key) [Method]
Compute DEFINITION's package lists of external and internal symbols.

Source [`package.lisp`], page 24.

`initialize-instance` *:before* ((*definition* *[*`system-file-definition`*]*, *page 102*) &key *system*) [Method]
Create and store a fake ASDF comoponent representing the system file.

Source [`asdf.lisp`], page 25.

`initialize-instance` *:after* ((*definition* *[*`classoid-definition`*]*, *page 76*) &key *packages pathnames*) [Method]
Compute classoid DEFINITION's foreign status.

Source [`symbol.lisp`], page 20.

`initialize-instance` *:after* ((*definition* *[*`component-definition`*]*, *page 80*) &key) [Method]
Compute component DEFINITION's location.

Source [`asdf.lisp`], page 25.

`initialize-instance` *:after* ((*definition* *[*`typed-structure-definition`*]*, *page 102*) &key *packages pathnames*) [Method]
Compute typed structure DEFINITION's type, element type, and slots.

Source [`symbol.lisp`], page 20.

`initialize-instance` *:after* ((*definition* *[*`system-definition`*]*, *page 100*) &key) [Method]
Extract author and maintainer contacts.

Source [`asdf.lisp`], page 25.

`make-instance` ((*class* *[*`abstract-class`*]*, *page 74*) &rest *initargs*) [Method]

Source [`util.lisp`], page 12.

`print-object` ((*report* *[*`report`*]*, *page 94*) *stream*) [Method]
Show REPORT's library name.

Source [`assess.lisp`], page 27.

`print-object` ((*definition* *[*`definition`*]*, *page 82*) *stream*) [Method]
Show DEFINITION's name.

Source [`definition.lisp`], page 19.

`validate-superclass` ((*class* *[*`abstract-class`*]*, *page 74*) (*superclass* `standard-class`)) [Method]

Package `sb-mop`.

Source [`util.lisp`], page 12.

`validate-superclass` ((*class* `standard-class`) (*superclass* *[*`abstract-class`*]*, *page 74*)) [Method]

Package `sb-mop`.

Source [`util.lisp`], page 12.

6.1.6 Classes

`abstract-class` [Class]
The Abstract Class meta-class.

Package [`net.didierverna.declt.setup`], page 31.

Source [`util.lisp`], page 12.

Direct superclasses
`standard-class`.

Direct methods
- [`make-instance`], page 73.
- [`validate-superclass`], page 73.
- [`validate-superclass`], page 73.

`accessor-method-definition` [Class]
Abstract root class for accessor methods.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
- [`accessor-mixin`], page 74.
- [`method-definition`], page 90.

Direct subclasses
- [`reader-method-definition`], page 93.
- [`writer-method-definition`], page 104.

`accessor-mixin` [Class]
Mixin class for accessor definitions.
An accessor is a funcoid which reads or writes a target slot in a classoid. More specifically, these are ordinary functions for structure slots, and methods for classes or conditions slots.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct subclasses
- [`accessor-method-definition`], page 74.
- [`ordinary-accessor-definition`], page 91.

Direct methods
- [`document`], page 132.
- [`target-slot`], page 71.

Direct slots

`target-slot` [Slot]
The target slot definition for this definition's accessor.

Initargs `:target-slot`

Readers [`target-slot`], page 71.

Writers *This slot is read-only.*

`alias-definition` [Class]
Abstract root class for alias definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
[`symbol-definition`], page 99.

Direct subclasses
- [`compiler-macro-alias-definition`], page 79.
- [`function-alias-definition`], page 86.
- [`macro-alias-definition`], page 89.

Direct methods
- [`category-name`], page 126.
- [`document`], page 129.
- [`index-command-name`], page 135.
- [`lambda-list`], page 62.
- [`name`], page 65.
- [`referee`], page 67.
- [`(setf referee)`], page 67.
- [`setfp`], page 68.
- [`source-pathname`], page 138.

Direct slots

`setf` [Slot]
Whether this is a setf alias definition.

Package `common-lisp`.

Initargs `:setf`

Readers [`setfp`], page 68.

Writers *This slot is read-only.*

`referee` [Slot]
The original definition this definition aliases.

Readers [`referee`], page 67.

Writers [`(setf referee)`], page 67.

`c-file-definition` [Class]
The class of ASDF C file definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

Direct superclasses
[`source-file-definition`], page 98.

`cl-source-file.asd` [Class]
A fake ASDF Lisp file component class for system files.

Package [`net.didierverna.declt.assess`], page 32.

Source [asdf.lisp], page 25.

Direct superclasses
cl-source-file.

Direct slots

type [Slot]

Package common-lisp.

Initform "asd"

class-definition [Class]

The class for class definitions.

Package [net.didierverna.declt.assess], page 32.

Source [symbol.lisp], page 20.

Direct superclasses
- [classoid-definition], page 76.
- [clos-classoid-mixin], page 77.

Direct methods
- [category-name], page 126.
- [definition-class], page 51.
- [direct-subclasses], page 54.
- [(setf direct-subclasses)], page 55.
- [direct-superclasses], page 56.
- [(setf direct-superclasses)], page 56.
- [index-command-name], page 135.

Direct slots

object [Slot]

Readers [definition-class], page 51.

Writers *This slot is read-only.*

direct-superclassoids [Slot]

Readers [direct-superclasses], page 56.

Writers [(setf direct-superclasses)], page 56.

direct-subclassoids [Slot]

Readers [direct-subclasses], page 54.

Writers [(setf direct-subclasses)], page 55.

classoid-definition [Class]

Abstract root class for classoid definitions. These are conditions, structures, and classes.

Package [net.didierverna.declt.assess], page 32.

Source [symbol.lisp], page 20.

Direct superclasses
[symbol-definition], page 99.

Direct subclasses
- [class-definition], page 76.

- [condition-definition], page 81.
- [structure-definition], page 99.

Direct methods

- [classoid], page 49.
- [direct-slots], page 54.
- [(setf direct-slots)], page 54.
- [document], page 130.
- [document], page 130.
- [document], page 129.
- [initialize-instance], page 73.

Direct slots

`object` [Slot]

Initargs :classoid

Readers [classoid], page 49.

Writers *This slot is read-only.*

`direct-slots` [Slot]

The list of direct slot definitions for this definition's classoid.

Readers [direct-slots], page 54.

Writers [(setf direct-slots)], page 54.

`clos-classoid-mixin` [Class]

Mixin for CLOS-based classoids.
These are conditions, ordinary structures, and classes.
All CLOS classoid mixin definitions respond to the following public protocols, which see:
- 'direct-default-initargs'.

Package [net.didierverna.declt.assess], page 32.

Source [symbol.lisp], page 20.

Direct subclasses

- [class-definition], page 76.
- [clos-structure-definition], page 78.
- [condition-definition], page 81.

Direct methods

- [direct-methods], page 54.
- [(setf direct-methods)], page 54.
- [direct-subclassoids], page 55.
- [(setf direct-subclassoids)], page 55.
- [direct-superclassoids], page 57.
- [(setf direct-superclassoids)], page 57.
- [document], page 129.
- [initialize-instance], page 72.
- [stabilize], page 140.

Direct slots

`direct-superclassoids` [Slot]
The list of direct superclassoid definitions for this definition's classoid.

Readers [`direct-superclassoids`], page 57.

Writers [`(setf direct-superclassoids)`], page 57.

`direct-subclassoids` [Slot]
The list of direct subclassoid definitions for this definition's classoid.

Readers [`direct-subclassoids`], page 55.

Writers [`(setf direct-subclassoids)`], page 55.

`direct-methods` [Slot]
The list of direct method definitions for this definition's classoid.

Readers [`direct-methods`], page 54.

Writers [`(setf direct-methods)`], page 54.

`clos-slot-definition` [Class]
The class of CLOS slot definitions.
All CLOS slot definitions respond to the following public protocols, which see:
- 'allocation',
- 'initform',
- 'initargs'.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
[`slot-definition`], page 97.

Direct methods
- [`docstring`], page 58.
- [`document`], page 129.
- [`stabilize`], page 140.
- [`value-type`], page 72.

`clos-structure-definition` [Class]
The class of CLOS structure definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
- [`clos-classoid-mixin`], page 77.
- [`structure-definition`], page 99.

Direct methods
- [`direct-substructures`], page 56.
- [`(setf direct-substructures)`], page 56.
- [`direct-superstructures`], page 57.
- [`(setf direct-superstructures)`], page 58.

Direct slots

`direct-superclassoids` [Slot]

Readers [`direct-superstructures`], page 57.

Writers [`(setf direct-superstructures)`], page 58.

`direct-subclassoids` [Slot]

Readers [`direct-substructures`], page 56.

Writers [`(setf direct-substructures)`], page 56.

`combination-definition` [Class]

Root class for method combination definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses

[`funcoid-definition`], page 85.

Direct subclasses

- [`long-combination-definition`], page 89.
- [`short-combination-definition`], page 96.

Direct methods

- [`category-name`], page 126.
- [`clients`], page 49.
- [`(setf clients)`], page 49.
- [`combination`], page 50.
- [`docstring`], page 59.
- [`document`], page 130.
- [`index-command-name`], page 135.
- [`lambda-list`], page 62.
- [`source-pathname`], page 138.
- [`stabilize`], page 140.

Direct slots

`object` [Slot]

Initargs `:combination`

Readers [`combination`], page 50.

Writers *This slot is read-only.*

`clients` [Slot]

The list of client definitions for this definition's method combnination. These are generic functions using this combination.

Readers [`clients`], page 49.

Writers [`(setf clients)`], page 49.

`compiler-macro-alias-definition` [Class]

The class of compiler macro alias definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [symbol.lisp], page 20.

Direct superclasses
[alias-definition], page 75.

Direct methods
- [docstring], page 58.
- [stabilize], page 139.

compiler-macro-definition [Class]
The class of compiler macro definitions.

Package [net.didierverna.declt.assess], page 32.

Source [symbol.lisp], page 20.

Direct superclasses
[funcoid-definition], page 85.

Direct methods
- [category-name], page 127.
- [definition-compiler-macro], page 51.
- [index-command-name], page 136.

Direct slots

object [Slot]
Initargs :compiler-macro
Readers [definition-compiler-macro], page 51.
Writers *This slot is read-only.*

component-definition [Class]
Abstract root class for ASDF components.
All component definitions respond to the following public protocols, which see:
- 'description'
- 'long-description'
- 'definition-version'
- 'if-feature'.

Package [net.didierverna.declt.assess], page 32.

Source [asdf.lisp], page 25.

Direct superclasses
[definition], page 82.

Direct subclasses
- [file-definition], page 84.
- [module-definition], page 90.

Direct methods
- [component], page 50.
- [dependencies], page 54.
- [(setf dependencies)], page 54.
- [docstring], page 58.
- [document], page 129.
- [document], page 128.

- [initialize-instance], page 73.
- [location], page 64.
- [(setf location)], page 64.
- [name], page 65.
- [parent], page 66.
- [(setf parent)], page 66.
- [safe-name], page 137.
- [source-pathname], page 138.
- [stabilize], page 139.

Direct slots

`object` [Slot]

Readers [component], page 50.

Writers *This slot is read-only.*

`location` [Slot]

The component's location (a namestring).

Readers [location], page 64.

Writers [(setf location)], page 64.

`parent` [Slot]

The parent definition for this definition's component.

Readers [parent], page 66.

Writers [(setf parent)], page 66.

`dependencies` [Slot]

The list of dependency definitions for this definition's component.

Readers [dependencies], page 54.

Writers [(setf dependencies)], page 54.

`condition-definition` [Class]

The class of condition definitions.

Package [net.didierverna.declt.assess], page 32.

Source [symbol.lisp], page 20.

Direct superclasses

- [classoid-definition], page 76.
- [clos-classoid-mixin], page 77.

Direct methods

- [category-name], page 126.
- [definition-condition], page 51.
- [direct-subconditions], page 55.
- [(setf direct-subconditions)], page 55.
- [direct-superconditions], page 57.
- [(setf direct-superconditions)], page 57.
- [index-command-name], page 135.

Direct slots

`object` [Slot]

Readers [`definition-condition`], page 51.

Writers *This slot is read-only.*

`direct-superclassoids` [Slot]

Readers [`direct-superconditions`], page 57.

Writers [`(setf direct-superconditions)`], page 57.

`direct-subclassoids` [Slot]

Readers [`direct-subconditions`], page 55.

Writers [`(setf direct-subconditions)`], page 55.

`constant-definition` [Class]

The class of constant definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses

[`variable-definition`], page 103.

Direct methods

- [`category-name`], page 127.
- [`index-command-name`], page 136.
- [`source-pathname`], page 139.

`definition` [Class]

Abstract root class for all definitions.
All definitions respond to the following public protocols, which see: - 'name', - 'docstring'.

Package [`net.didierverna.declt.assess`], page 32.

Source [`definition.lisp`], page 19.

Direct subclasses

- [`component-definition`], page 80.
- [`package-definition`], page 92.
- [`symbol-definition`], page 99.

Direct methods

- [`foreignp`], page 61.
- [`(setf foreignp)`], page 61.
- [`object`], page 65.
- [`print-object`], page 73.
- [`safe-name`], page 137.
- [`source-file`], page 68.
- [`(setf source-file)`], page 68.
- [`stabilize`], page 141.
- [`uid`], page 71.

- [(setf uid)], page 71.

Direct slots

`object` [Slot]
The corresponding Lisp object, or NIL.
Only constants, special variables, symbol macros, and aliases lack such an object.

Initargs `:object`

Readers [object], page 65.

Writers *This slot is read-only.*

`uid` [Slot]
This definition's UID.

Readers [uid], page 71.

Writers [(setf uid)], page 71.

`source-file` [Slot]
The source file definition for this definition's object.

Readers [source-file], page 68.

Writers [(setf source-file)], page 68.

`foreign` [Slot]
Whether this definition is foreign.

Initargs `:foreign`

Readers [foreignp], page 61.

Writers [(setf foreignp)], page 61.

`doc-file-definition` [Class]
The class of ASDF doc file definitions.

Package [net.didierverna.declt.assess], page 32.

Source [asdf.lisp], page 25.

Direct superclasses
[static-file-definition], page 99.

Direct subclasses
[html-file-definition], page 88.

`expander-definition` [Class]
Abstract root class for setf expander definitions.

Package [net.didierverna.declt.assess], page 32.

Source [symbol.lisp], page 20.

Direct superclasses
[funcoid-definition], page 85.

Direct subclasses

- [long-expander-definition], page 89.
- [short-expander-definition], page 97.

Direct methods

- [`category-name`], page 127.
- [`docstring`], page 59.
- [`document`], page 131.
- [`expander`], page 60.
- [`index-command-name`], page 135.
- [`source-pathname`], page 138.
- [`stabilize`], page 140.
- [`standalone-reader`], page 69.
- [`(setf standalone-reader)`], page 69.

Direct slots

`object` [Slot]

Initargs `:expander`

Readers [`expander`], page 60.

Writers *This slot is read-only.*

`setf` [Slot]

Package `common-lisp`.

Initform `t`

`standalone-reader` [Slot]

A standalone reader definition for this definition's expander, or NIL.
If it exists, it's a definition for a function or macro with the same signature as that of the expander's access-fn. Note that the Common Lisp standard does not impose any actual relation between the setf expander and its access-fn. In fact, the access-fn may not even exist at all. However, if it does, it is very likely that it is a reader for the place updated by this setf expander.

Readers [`standalone-reader`], page 69.

Writers [`(setf standalone-reader)`], page 69.

`file-definition` [Class]

The class of ASDF file definitions.
All file definitions respond to the following public protocols, which see: - 'extension'.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

Direct superclasses

[`component-definition`], page 80.

Direct subclasses

[`source-file-definition`], page 98.

Direct methods

- [`category-name`], page 125.
- [`file`], page 61.
- [`index-command-name`], page 134.
- [`safe-name`], page 137.

Direct slots

`object` [Slot]

Initargs `:file`

Readers [`file`], page 61.

Writers *This slot is read-only.*

`funcoid-definition` [Class]

Abstract root class for functional definitions.
These are (compiler) macros, (generic) functions, methods, setf expanders, method combinations, and types.
All funcoid definitions respond to the following public protocols, which see: - 'lambda-list'.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses

[`symbol-definition`], page 99.

Direct subclasses

- [`combination-definition`], page 79.
- [`compiler-macro-definition`], page 80.
- [`expander-definition`], page 83.
- [`method-definition`], page 90.
- [`setfable-funcoid-definition`], page 95.
- [`type-definition`], page 102.

Direct methods

- [`document`], page 133.
- [`document`], page 133.
- [`funcoid`], page 61.
- [`lambda-list`], page 63.
- [`lambda-list`], page 63.
- [`name`], page 65.
- [`setfp`], page 68.

Direct slots

`object` [Slot]

Readers [`funcoid`], page 61.

Writers *This slot is read-only.*

`setf` [Slot]

Whether this is a setf definition.

Package `common-lisp`.

Initargs `:setf`

Readers [`setfp`], page 68.

Writers *This slot is read-only.*

`function-alias-definition` [Class]
The class of non-setf function alias definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
[`alias-definition`], page 75.

Direct methods
- [`docstring`], page 58.
- [`stabilize`], page 139.

`function-definition` [Class]
Abstract root class for functions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
[`setfable-funcoid-definition`], page 95.

Direct subclasses
- [`generic-function-definition`], page 86.
- [`ordinary-function-definition`], page 91.

Direct methods
[`definition-function`], page 52.

Direct slots

`object` [Slot]

Initargs `:function`

Readers [`definition-function`], page 52.

Writers *This slot is read-only.*

`generic-accessor-definition` [Class]
Abstract root class for generic accessor functions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
[`generic-function-definition`], page 86.

Direct subclasses
- [`generic-reader-definition`], page 87.
- [`generic-writer-definition`], page 88.

`generic-function-definition` [Class]
The class of generic function definitions.
All generic function definitions respond to the following public protocols, which see:
- 'combination-options'.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
[function-definition], page 86.

Direct subclasses
[generic-accessor-definition], page 86.

Direct methods
- [category-name], page 127.
- [combination], page 49.
- [(setf combination)], page 49.
- [document], page 131.
- [document], page 131.
- [generic], page 61.
- [index-command-name], page 135.
- [methods], page 64.
- [(setf methods)], page 64.
- [stabilize], page 140.

Direct slots

object [Slot]

Initargs :generic

Readers [generic], page 61.

Writers *This slot is read-only.*

methods [Slot]
The list of method definitions for this definition's generic function.

Readers [methods], page 64.

Writers [(setf methods)], page 64.

combination [Slot]
The method combination definition for this definition's generic function.

Readers [combination], page 49.

Writers [(setf combination)], page 49.

generic-reader-definition [Class]
The class of generic reader function definitions.
A generic function is considered to be a reader function when all its mehtods are reader methods.

Package [net.didierverna.declt.assess], page 32.

Source [symbol.lisp], page 20.

Direct superclasses
[generic-accessor-definition], page 86.

Direct methods
- [category-name], page 126.
- [document], page 131.

`generic-writer-definition` [Class]

The class of generic writer function definitions.
A generic function is considered to be a writer function when all its mehtods are writer methods.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
[`generic-accessor-definition`], page 86.

Direct methods
- [`category-name`], page 126.
- [`document`], page 130.

`html-file-definition` [Class]

The class of ASDF HTML file definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

Direct superclasses
[`doc-file-definition`], page 83.

`java-file-definition` [Class]

The class of ASDF Java file definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

Direct superclasses
[`source-file-definition`], page 98.

`lisp-file-definition` [Class]

The class of ASDF Lisp file definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

Direct superclasses
[`source-file-definition`], page 98.

Direct subclasses
[`system-file-definition`], page 102.

Direct methods
- [`definitions`], page 53.
- [`(setf definitions)`], page 53.
- [`document`], page 128.
- [`stabilize`], page 139.

Direct slots

`definitions` [Slot]

The list of definitions for this definition's file.

Readers [`definitions`], page 53.

Writers [`(setf definitions)`], page 53.

`long-combination-definition` [Class]
Class for long method combination definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
[`combination-definition`], page 79.

`long-expander-definition` [Class]
The class of long form setf expanders definitions.
This class is shared by expanders created with either the long form of DEFSETF, or DEFINE-SETF-EXPANDER.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
[`expander-definition`], page 83.

Direct methods
- [`document`], page 131.
- [`lambda-list`], page 62.

`macro-alias-definition` [Class]
The class of macro alias definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
[`alias-definition`], page 75.

Direct methods
- [`docstring`], page 58.
- [`stabilize`], page 140.

`macro-definition` [Class]
The class of macro definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
[`setfable-funcoid-definition`], page 95.

Direct methods
- [`category-name`], page 127.
- [`index-command-name`], page 136.
- [`macro`], page 64.

Direct slots

`object` [Slot]

Initargs `:macro`

Readers [`macro`], page 64.

Writers *This slot is read-only.*

`method-definition` [Class]

Abstract root class for method definitions.
All method definitions respond to the following public protocols, which see: - 'qualifiers'.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
[`funcoid-definition`], page 85.

Direct subclasses
[`accessor-method-definition`], page 74.

Direct methods

- [`category-name`], page 126.
- [`definition-method`], page 52.
- [`document`], page 130.
- [`index-command-name`], page 135.
- [`lambda-list`], page 62.
- [`owner`], page 66.
- [`(setf owner)`], page 66.
- [`safe-name`], page 137.
- [`specializers`], page 68.
- [`(setf specializers)`], page 68.
- [`stabilize`], page 140.

Direct slots

`object` [Slot]

Initargs `:method`

Readers [`definition-method`], page 52.

Writers *This slot is read-only.*

`owner` [Slot]

The generic function definition for this definition's method.

Readers [`owner`], page 66.

Writers [`(setf owner)`], page 66.

`specializers` [Slot]

The specializers of this definition's method.
This is a list of either class definitions (for regular specializers), or raw EQL specializers.

Readers [`specializers`], page 68.

Writers [`(setf specializers)`], page 68.

`module-definition` [Class]

The class of ASDF module definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

Direct superclasses
[component-definition], page 80.

Direct subclasses
[system-definition], page 100.

Direct methods
- [category-name], page 125.
- [children], page 49.
- [(setf children)], page 49.
- [document], page 128.
- [index-command-name], page 134.
- [module], page 65.
- [stabilize], page 139.

Direct slots

`object` [Slot]

Initargs :module

Readers [module], page 65.

Writers *This slot is read-only.*

`children` [Slot]
The list of child definitions for this definition's module.

Readers [children], page 49.

Writers [(setf children)], page 49.

`ordinary-accessor-definition` [Class]
Abstract root class for ordinary accessor functions.

Package [net.didierverna.declt.assess], page 32.

Source [symbol.lisp], page 20.

Direct superclasses
- [accessor-mixin], page 74.
- [ordinary-function-definition], page 91.

Direct subclasses
- [ordinary-reader-definition], page 92.
- [ordinary-writer-definition], page 92.

`ordinary-function-definition` [Class]
The class of ordinary functions.

Package [net.didierverna.declt.assess], page 32.

Source [symbol.lisp], page 20.

Direct superclasses
[function-definition], page 86.

Direct subclasses
[ordinary-accessor-definition], page 91.

Direct methods
- [category-name], page 127.
- [index-command-name], page 135.

`ordinary-reader-definition` [Class]

The class of ordinary reader definitions.
An ordinary reader is an ordinary function that reads a slot in a structure.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
[`ordinary-accessor-definition`], page 91.

Direct methods
- [`category-name`], page 127.
- [`document`], page 131.

`ordinary-writer-definition` [Class]

The class of ordinary writer definitions.
An ordinary writer is an ordinary function that writes a slot in a structure.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
[`ordinary-accessor-definition`], page 91.

Direct methods
- [`category-name`], page 127.
- [`document`], page 131.

Direct slots

`setf` [Slot]

Package `common-lisp`.

Initform `t`

`package-definition` [Class]

The class of package definitions.
All package definitions respond to the following public protocols, which see:
- 'nicknames'.

Package [`net.didierverna.declt.assess`], page 32.

Source [`package.lisp`], page 24.

Direct superclasses
[`definition`], page 82.

Direct methods
- [`category-name`], page 125.
- [`definition-package`], page 52.
- [`definitions`], page 53.
- [`(setf definitions)`], page 53.
- [`document`], page 129.
- [`external-symbols`], page 134.
- [`(setf external-symbols)`], page 134.
- [`index-command-name`], page 134.

- [initialize-instance], page 73.
- [internal-symbols], page 136.
- [(setf internal-symbols)], page 136.
- [name], page 65.
- [private-definitions], page 66.
- [public-definitions], page 67.
- [stabilize], page 139.
- [use-list], page 71.
- [(setf use-list)], page 71.
- [used-by-list], page 72.
- [(setf used-by-list)], page 72.

Direct slots

`object` [Slot]

Initargs `:package`

Readers [definition-package], page 52.

Writers *This slot is read-only.*

`external-symbols` [Slot]

The list of corresponding external symbols.

Readers [external-symbols], page 134.

Writers [(setf external-symbols)], page 134.

`internal-symbols` [Slot]

The list of corresponding internal symbols.

Readers [internal-symbols], page 136.

Writers [(setf internal-symbols)], page 136.

`use-list` [Slot]

The definitions use-list for this definition's package.

Readers [use-list], page 71.

Writers [(setf use-list)], page 71.

`used-by-list` [Slot]

The definitions used-by-list for this definition's package.

Readers [used-by-list], page 72.

Writers [(setf used-by-list)], page 72.

`definitions` [Slot]

The list of corresponding definitions.

Readers [definitions], page 53.

Writers [(setf definitions)], page 53.

`reader-method-definition` [Class]

The class of reader method definitions.
A reader method is a method that reads a slot in a class or condition.

Package [net.didierverna.declt.assess], page 32.

Source [symbol.lisp], page 20.

Direct superclasses
[accessor-method-definition], page 74.

Direct methods
[category-name], page 126.

`report` [Class]

The Report class.
This is the class holding all extracted documentation information.

Package [net.didierverna.declt.assess], page 32.

Source [assess.lisp], page 27.

Direct methods

- [conclusion], page 50.
- [(setf conclusion)], page 50.
- [contacts], page 50.
- [(setf contacts)], page 50.
- [copyright-years], page 51.
- [(setf copyright-years)], page 51.
- [definitions], page 53.
- [(setf definitions)], page 53.
- [introduction], page 62.
- [(setf introduction)], page 62.
- [library-name], page 63.
- [(setf library-name)], page 63.
- [library-version], page 63.
- [(setf library-version)], page 63.
- [license], page 63.
- [(setf license)], page 63.
- [print-object], page 73.
- [system-name], page 70.
- [tagline], page 71.
- [(setf tagline)], page 71.

Direct slots

`system-name` [Slot]

The main system's name, coerced to a string.

Initargs `:system-name`

Readers [system-name], page 70.

Writers *This slot is read-only.*

`library-name` [Slot]

The library's name.

Readers [library-name], page 63.

Writers [(setf library-name)], page 63.

`tagline` [Slot]
The reference manual's tagline.

Readers [`tagline`], page 71.

Writers [`(setf tagline)`], page 71.

`library-version` [Slot]
The library's version.

Readers [`library-version`], page 63.

Writers [`(setf library-version)`], page 63.

`contacts` [Slot]
The list of contacts for the library.
Each element is of the form (NAME . EMAIL) where both NAME and EMAIL are strings or NIL, and cannot be null at the same time.

Readers [`contacts`], page 50.

Writers [`(setf contacts)`], page 50.

`copyright-years` [Slot]
A copyright years string.

Readers [`copyright-years`], page 51.

Writers [`(setf copyright-years)`], page 51.

`license` [Slot]
The library's license.

Readers [`license`], page 63.

Writers [`(setf license)`], page 63.

`introduction` [Slot]
Contents for an introduction chapter.

Readers [`introduction`], page 62.

Writers [`(setf introduction)`], page 62.

`conclusion` [Slot]
Contents for a conclusion chapter.

Readers [`conclusion`], page 50.

Writers [`(setf conclusion)`], page 50.

`definitions` [Slot]
The list of definitions.

Readers [`definitions`], page 53.

Writers [`(setf definitions)`], page 53.

`setfable-funcoid-definition` [Class]
Abstract root class for setfable funcoids.
These are (generic) functions and macros. A funcoid is setfable when it may be related to one or more setf expanders. There are two kinds of relation to a setf expander: 1. the funcoid' signature is the same as that of an expander's access-fn, and 2. a short form setf expander expands to it (i.e., it has this funcoid as its update-fn).

Package [`net.didierverna.declt.assess`], page 32.

Source [symbol.lisp], page 20.

Direct superclasses
[funcoid-definition], page 85.

Direct subclasses
- [function-definition], page 86.
- [macro-definition], page 89.

Direct methods
- [document], page 132.
- [document], page 132.
- [document], page 132.
- [expander-for], page 60.
- [(setf expander-for)], page 60.
- [expanders-to], page 60.
- [(setf expanders-to)], page 60.
- [stabilize], page 140.

Direct slots

expander-for [Slot]
A setf expander definition for this funcoid, or NIL.
This is the definition of a setf expander that expands forms identical to this funcoid's signature. There can be only one. Note that the Common Lisp standard does not impose any actual relation between the setf expander and its access-fn. In fact, the access-fn may not even exist at all. However, if it does, it is very likely that it is a reader for the place updated by this setf expander.

Readers [expander-for], page 60.

Writers [(setf expander-for)], page 60.

expanders-to [Slot]
The list of setf expander definitions to this funcoid.
This is a list of definitions for short form setf expanders that have this funcoid as their update-fn. There might be more than one.

Readers [expanders-to], page 60.

Writers [(setf expanders-to)], page 60.

short-combination-definition [Class]
The class of short method combination definitions.
All short method combination definitions respond to the following public protocols, which see:
- 'identity-with-one-argument'.

Package [net.didierverna.declt.assess], page 32.

Source [symbol.lisp], page 20.

Direct superclasses
[combination-definition], page 79.

Direct methods
- [document], page 130.

- [stabilize], page 140.
- [standalone-combinator], page 69.
- [(setf standalone-combinator)], page 69.

Direct slots

`standalone-combinator` [Slot]
The corresponding standalone combinator definition, or NIL.
This is a function or macro definition. Note that if this definition is unavailable, it means that the method combination itself cannot be used (yet).

Readers [standalone-combinator], page 69.

Writers [(setf standalone-combinator)], page 69.

`short-expander-definition` [Class]
The class of short form setf expanders definitions.
Short form setf expanders simply expand to a globally defined function or macro.

Package [net.didierverna.declt.assess], page 32.

Source [symbol.lisp], page 20.

Direct superclasses
[expander-definition], page 83.

Direct methods
- [document], page 131.
- [lambda-list], page 62.
- [stabilize], page 140.
- [standalone-writer], page 70.
- [(setf standalone-writer)], page 70.

Direct slots

`standalone-writer` [Slot]
A standalone writer definition for this definition's expander, or NIL. This is a function or macro definition. Note that if this definition is unavailable, it means that the expander itself cannot be used (yet).

Readers [standalone-writer], page 70.

Writers [(setf standalone-writer)], page 70.

`slot-definition` [Class]
Abstract root class for slots.
All slot definitions respond to the following public protocols, which see: - 'value-type'.

Package [net.didierverna.declt.assess], page 32.

Source [symbol.lisp], page 20.

Direct superclasses
[varoid-definition], page 104.

Direct subclasses
- [clos-slot-definition], page 78.
- [typed-structure-slot-definition], page 103.

Direct methods
- [category-name], page 126.

- [document], page 129.
- [index-command-name], page 135.
- [owner], page 66.
- [(setf owner)], page 66.
- [readers], page 67.
- [(setf readers)], page 67.
- [safe-name], page 137.
- [slot], page 68.
- [source-pathname], page 138.
- [writers], page 72.
- [(setf writers)], page 72.

Direct slots

`object` [Slot]

Initargs `:slot`

Readers [slot], page 68.

Writers *This slot is read-only.*

`owner` [Slot]

The definition for the owner of this definition's slot.

Readers [owner], page 66.

Writers [(setf owner)], page 66.

`readers` [Slot]

The list of definitions for this definition's slot readers.

Readers [readers], page 67.

Writers [(setf readers)], page 67.

`writers` [Slot]

The list of definitions for this definition's slot writers.

Readers [writers], page 72.

Writers [(setf writers)], page 72.

`source-file-definition` [Class]

The class of ASDF source file definitions.

Package [net.didierverna.declt.assess], page 32.

Source [asdf.lisp], page 25.

Direct superclasses

[file-definition], page 84.

Direct subclasses

- [c-file-definition], page 75.
- [java-file-definition], page 88.
- [lisp-file-definition], page 88.
- [static-file-definition], page 99.

`special-definition` [Class]
The class of special variable definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
[`variable-definition`], page 103.

Direct methods
- [`category-name`], page 127.
- [`index-command-name`], page 136.
- [`source-pathname`], page 138.

`static-file-definition` [Class]
The class of ASDF static file definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

Direct superclasses
[`source-file-definition`], page 98.

Direct subclasses
[`doc-file-definition`], page 83.

`structure-definition` [Class]
Abstract root class for structures.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
[`classoid-definition`], page 76.

Direct subclasses
- [`clos-structure-definition`], page 78.
- [`typed-structure-definition`], page 102.

Direct methods
- [`category-name`], page 126.
- [`definition-structure`], page 52.
- [`index-command-name`], page 135.

Direct slots

`object` [Slot]

Readers [`definition-structure`], page 52.

Writers *This slot is read-only.*

`symbol-definition` [Class]
Abstract root class for definitions named by symbols.
All symbol definitions respond to the following public protocols, which see: - 'publicp'.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses

[definition], page 82.

Direct subclasses

- [alias-definition], page 75.
- [classoid-definition], page 76.
- [funcoid-definition], page 85.
- [varoid-definition], page 104.

Direct methods

- [definition-symbol], page 53.
- [home-package], page 62.
- [(setf home-package)], page 62.
- [name], page 65.
- [safe-name], page 137.
- [stabilize], page 140.

Direct slots

symbol [Slot]

The symbol naming this definition.

Package common-lisp.

Initargs :symbol

Readers [definition-symbol], page 53.

Writers *This slot is read-only.*

home-package [Slot]

The home package definition for this definition's symbol.
Every definition gets a home package, even foreign ones. A home package can only be null when the definition's symbol is uninterned.

Readers [home-package], page 62.

Writers [(setf home-package)], page 62.

symbol-macro-definition [Class]

The class of symbol macro definitions.

Package [net.didierverna.declt.assess], page 32.

Source [symbol.lisp], page 20.

Direct superclasses

[varoid-definition], page 104.

Direct methods

- [category-name], page 127.
- [docstring], page 59.
- [index-command-name], page 136.
- [source-pathname], page 138.

system-definition [Class]

The class of ASDF system definitions.
All system definitions respond to the following public protocols, which see: - 'long-name',

- ‘mailto’,
- ‘homepage’,
- ‘source-control’,
- ‘bug-tracker’,
- ‘license-name’.

Package [net.didierverna.declt.assess], page 32.

Source [asdf.lisp], page 25.

Direct superclasses

[module-definition], page 90.

Direct methods

- [authors], page 48.
- [(setf authors)], page 48.
- [category-name], page 125.
- [defsystem-dependencies], page 53.
- [(setf defsystem-dependencies)], page 53.
- [document], page 128.
- [index-command-name], page 134.
- [initialize-instance], page 73.
- [maintainers], page 64.
- [(setf maintainers)], page 64.
- [stabilize], page 139.
- [system], page 70.

Direct slots

`object` [Slot]

Initargs `:system`

Readers [system], page 70.

Writers *This slot is read-only.*

`parent` [Slot]

`maintainers` [Slot]

The list of parsed maintainer contacts.
See ‘parse-contact-string’ for more information.

Readers [maintainers], page 64.

Writers [(setf maintainers)], page 64.

`authors` [Slot]

The list of parsed author contacts.
See ‘parse-contact-string’ for more information.

Readers [authors], page 48.

Writers [(setf authors)], page 48.

`defsystem-dependencies` [Slot]

The list of defsystem dependency definitions.

Readers [defsystem-dependencies], page 53.

Writers [(setf defsystem-dependencies)], page 53.

`system-file-definition` [Class]

The class of ASDF system file definitions.
This class represents ASDF system files as Lisp files. Because system files are not components, we use an ad-hoc fake component class for them, 'cl-source-file.asd', which see.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

Direct superclasses
[`lisp-file-definition`], page 88.

Direct methods
[`initialize-instance`], page 73.

`type-definition` [Class]

The class of type definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
[`funcoid-definition`], page 85.

Direct methods

- [`category-name`], page 127.
- [`docstring`], page 59.
- [`document`], page 132.
- [`document`], page 132.
- [`expander`], page 60.
- [`index-command-name`], page 136.
- [`lambda-list`], page 63.
- [`source-pathname`], page 138.

Direct slots

`object` [Slot]

Initargs `:expander`

Readers [`expander`], page 60.

Writers *This slot is read-only.*

`typed-structure-definition` [Class]

The class of typed structure definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
[`structure-definition`], page 99.

Direct methods

- [`docstring`], page 59.
- [`document`], page 129.
- [`element-type`], page 59.
- [`(setf element-type)`], page 59.

- [initialize-instance], page 73.
- [structure-type], page 70.
- [(setf structure-type)], page 70.

Direct slots

`type` [Slot]
The structure type, either LIST or VECTOR.

Package common-lisp.

Initargs :type

Readers [structure-type], page 70.

Writers [(setf structure-type)], page 70.

`element-type` [Slot]
The structure's element type.
It is T for list structures, but may be something else for vector ones.

Initargs :element-type

Readers [element-type], page 59.

Writers [(setf element-type)], page 59.

`typed-structure-slot-definition` [Class]
The class of typed structure slot definitions.

Package [net.didierverna.declt.assess], page 32.

Source [symbol.lisp], page 20.

Direct superclasses
[slot-definition], page 97.

Direct methods
- [docstring], page 58.
- [document], page 129.
- [stabilize], page 140.
- [value-type], page 72.

`variable-definition` [Class]
Abstract root class for constant and special variables.

Package [net.didierverna.declt.assess], page 32.

Source [symbol.lisp], page 20.

Direct superclasses
[varoid-definition], page 104.

Direct subclasses
- [constant-definition], page 82.
- [special-definition], page 99.

Direct methods
[docstring], page 59.

`varoid-definition` [Class]
Abstract root class for simply valued symbol definitions. These are constants, special variables, and symbol macros.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
[`symbol-definition`], page 99.

Direct subclasses
- [`slot-definition`], page 97.
- [`symbol-macro-definition`], page 100.
- [`variable-definition`], page 103.

Direct methods
- [`document`], page 133.
- [`document`], page 133.
- [`document`], page 133.

`writer-method-definition` [Class]
The class of writer method definitions.
A writer method is a method that writes a slot in a class or condition.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

Direct superclasses
[`accessor-method-definition`], page 74.

Direct methods
[`category-name`], page 126.

6.1.7 Types

`non-empty-string` () [Type]

Package [`net.didierverna.declt.setup`], page 31.

Source [`util.lisp`], page 12.

6.2 Internals

6.2.1 Special variables

`*blanks*` [Special Variable]
A list of blank characters and their associated revealed representation. Each element in this list is of the form (#BLANK . #REPLACEMENT).

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

Source [`doc.lisp`], page 15.

`*categories*` [Special Variable]
The list of definition categories.
Each category is of the form (TITLE FILTER).
- TITLE (a string) serves as the section title.

- FILTER can be either a definition type (symbol), in which case definitions of that type a retained, or a predicate function of one (definition) argument, which should return T if the definition is to be retained.

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

Source [symbol.lisp], page 15.

`*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 4.1 of the user manual for more information.

Package [net.didierverna.declt.setup], page 31.

Source [configuration.lisp], page 11.

`*fragile-characters*` [Special Variable]

An association list of Texinfo fragile (anchor) characters. Elements are the form (CHAR . ALT) where CHAR is the fragile (anchor) character and ALT is an alternative Unicode character.

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

Source [texi.lisp], page 13.

`*licenses*` [Special Variable]

Package [net.didierverna.declt.assess], page 32.

Source [license.lisp], page 20.

`*section-names*` [Special Variable]

The numbered, unumbered and appendix section names sorted by level.

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

Source [texi.lisp], page 13.

`*special-characters*` [Special Variable]

An association list of Texinfo special characters.
Elements are the form (CHAR . COMMAND) where CHAR is the special character and COMMAND is the name of the corresponding Texinfo alphabetic command.

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

Source [texi.lisp], page 13.

`*stabilized*` [Special Variable]

Whether the stabilization process is over.
This variable is set to NIL whenever new definitions are created during the process. Stabilization is run over and over again until nothing moves anymore.

Package [net.didierverna.declt.assess], page 32.

Source [finalize.lisp], page 27.

6.2.2 Macros

`dd-element-type` (*dd*) [Macro]
Duplication of SBCL's macro.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`deffn` ((*category name lambda-list* &optional *qualifiers*) &body *body*) [Macro]
Execute BODY as part of a @deffn {CATEGORY} {NAME} [QUALIFIERS] LAMBDA-LIST. CATEGORY, NAME, QUALIFIERS, and LAMBDA-LIST are escaped for Texinfo prior to rendering. LAMBDA-LIST should be provided by 'safe-lambda-list' or 'safe-specializers', which see. BODY should render on *standard-output*.

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

Source [`texi.lisp`], page 13.

`defindent` (*symbol indent*) [Macro]
Wrapper around 'clindent' to avoid quoting SYMBOL and INDENT.

Package [`net.didierverna.declt.setup`], page 31.

Source [`readtable.lisp`], page 12.

`deftp` ((*category name* &optional *lambda-list*) &body *body*) [Macro]
Execute BODY as part of a @deftp {CATEGORY} {NAME} [LAMBDA-LIST] environment. CATEGORY, NAME, and LAMBDA-LIST are escaped for Texinfo prior to rendering. LAMBDA-LIST should be provided by 'safe-lambda-list', which see.
BODY should render on *standard-output*.

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

Source [`texi.lisp`], page 13.

`defvr` (*category name* &body *body*) [Macro]
Execute BODY as part of 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 13.

`destabilize` (*definitions expression*) [Macro]
Invalidate the stabilization process by adding a new definition. EXPRESSION should evaluate to a new definition. ENDPUSH that definition to DEFINITIONS (a symbol), mark the stabilization process as dirty, and return that definition.

Package [`net.didierverna.declt.assess`], page 32.

Source [`finalize.lisp`], page 27.

`item` ((&optional *title*) &body *body*) [Macro]
Execute BODY as part of an @item [TITLE]. BODY should render on *standard-output*.

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

Source [`texi.lisp`], page 13.

`itemize` ((&optional *kind*) &body *body*) [Macro]
Execute BODY as part of an @itemize KIND environment. KIND should be a string designator. It defaults to @bullet. BODY should render on *standard-output*.

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

Source [`texi.lisp`], page 13.

`multitable` ((&rest *fractions*) &body *body*) [Macro]
Execute BODY as part of a @multitable @columnFRACTIONS environment. BODY should render on *standard-output*.

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

Source [`texi.lisp`], page 13.

`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 13.

`table` ((&optional *kind*) &body *body*) [Macro]
Execute BODY as part of a @table KIND environment.
KIND should be a string designator. It defaults to @strong. BODY should render on *standard-output*.

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

Source [`texi.lisp`], page 13.

6.2.3 Ordinary functions

`%deffn` (*x category name lambda-list* &optional *qualifiers*) [Function]
Render a @deffn[x] {CATEGORY} {NAME} [QUALIFIERS] LAMBDA-LIST line. Rendering is done on *standard-output*. CATEGORY, NAME, QUALIFIERS, and LAMBDA-LIST are escaped for Texinfo prior to rendering. LAMBDA-LIST should be provided by 'safe-lambda-list' or 'safe-specializers', which see.

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

Source [`texi.lisp`], page 13.

`%version` (*type major minor status level name*) [Function]

Package [`net.didierverna.declt.setup`], page 31.

Source [`version.lisp`], page 12.

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

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

Source [`texi.lisp`], page 13.

`@deffn` (*category name lambda-list* &optional *qualifiers*) [Function]
Render a @deffn {CATEGORY} {NAME} [QUALIFIERS] LAMBDA-LIST line. Rendering is done on *standard-output*. CATEGORY, NAME, QUALIFIERS, and LAMBDA-LIST are escaped for Texinfo prior to rendering. LAMBDA-LIST should be provided by 'safe-lambda-list' or 'safe-specializers', which see.

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

Source [texi.lisp], page 13.

`@deffnx` (*category name lambda-list* &optional *qualifiers*) [Function]
Render a @deffnx {CATEGORY} {NAME} [QUALIFIERS] LAMBDA-LIST line. Rendering is done on *standard-output*. CATEGORY, NAME, QUALIFIERS, and LAMBDA-LIST are escaped for Texinfo prior to rendering. LAMBDA-LIST should be provided by 'safe-lambda-list' or 'safe-specializers', which see.

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

Source [texi.lisp], page 13.

`@deftp` (*category name* &optional *lambda-list*) [Function]
Render a @deftp {CATEGORY} {NAME} [LAMBDA-LIST] line on *standard-output*. CATEGORY, NAME, and LAMBDA-LIST are escaped for Texinfo prior to rendering. LAMBDA-LIST should be provided by 'safe-lambda-list', which see.

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

Source [texi.lisp], page 13.

`@defvr` (*category name*) [Function]
Render a @defvr {CATEGORY} {NAME} line on *standard-output*. CATEGORY and NAME are escaped for Texinfo prior to rendering.

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

Source [texi.lisp], page 13.

`@end` (*environment*) [Function]
Render and @end ENVIRONMENT line on *standard-output*. ENVIRONMENT should be a string designator.

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

Source [texi.lisp], page 13.

`@item` (&optional *title*) [Function]
Render an @item [TITLE] line on *standard-output*.

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

Source [texi.lisp], page 13.

`@itemize` (&optional *kind*) [Function]
Render an @itemize KIND line on *standard-output*.
KIND should be a string designator. It defaults to @bullet.

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

Source [texi.lisp], page 13.

`@multitable` (&rest *fractions*) [Function]
Render a @multitable @columnFRACTIONS line on *standard-output*.

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

Source [texi.lisp], page 13.

`@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 13.

`@table` (&optional *kind*) [Function]
Render a @table KIND line on *standard-output*.
KIND should be a string designator. It defaults to @strong.

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

Source [`texi.lisp`], page 13.

`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 15.

`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 15.

`add-child` (*parent child*) [Function]
Add CHILD node to PARENT node and return CHILD.

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

Source [`texi.lisp`], page 13.

`add-definitions-node` (*parent report context*) [Function]
Add REPORT's definitions node to PARENT in CONTEXT.

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

Source [`symbol.lisp`], page 15.

`add-files-node` (*parent report context*) [Function]
Add REPORT's files node to PARENT in CONTEXT.

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

Source [`asdf.lisp`], page 18.

`add-modules-node` (*parent report context*) [Function]
Add REPORT's modules node to PARENT in CONTEXT.

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

Source [`asdf.lisp`], page 18.

`add-packages-node` (*parent report context*) [Function]
Add REPORT's packages node to PARENT in CONTEXT.

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

Source [`package.lisp`], page 18.

`add-systems-node` (*parent report context*) [Function]
Add REPORT's systems node to PARENT in CONTEXT.

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

Source [`asdf.lisp`], page 18.

`anchor` (*definition*) [Function]
Render DEFINITION's anchoring command on *STANDARD-OUTPUT*.

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

Source [`doc.lisp`], page 15.

`anchor-and-index` (*definition*) [Function]
Render DEFINITION's anchoring and indexing commands on *STANDARD-OUTPUT*.

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

Source [`doc.lisp`], page 15.

`anchor-name` (*definition*) [Function]
Return DEFINITION's anchor name, that is, `"(<UID>)"`.

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

Source [`doc.lisp`], page 15.

`clindent` (*symbol indent*) [Function]
Send SYMBOL's INDENTation information to Emacs.
Emacs will set the 'common-lisp-indent-function property.
If INDENT is a symbol, use its indentation definition. Otherwise, INDENT is considered as an indentation definition.

Package [`net.didierverna.declt.setup`], page 31.

Source [`readtable.lisp`], page 12.

`components` (*module type*) [Function]
Return the list of all (sub)TYPE components found in MODULE's tree.

Package [`net.didierverna.declt.assess`], page 32.

Source [`assess.lisp`], page 27.

`copy-node` (*instance*) [Function]

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

Source [`texi.lisp`], page 13.

`current-time-string` () [Function]
Return the current time as a string.

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

Source [`misc.lisp`], page 13.

`declt-1` (*report* &rest *keys* &key *locations default-values foreign-definitions declt-notice output-directory file-name info-name info-category*) [Function]

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

Source [`declt.lisp`], page 19.

`definition-source-by-name` (*definition type*) [Function]
Return DEFINITION's source for TYPE.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`domesticp` (*symbol pathname packages pathnames*) [Function]
Return T if a definition for SYMBOL originating in PATHNAME is domestic.
A definition is considered domestic under the following conditions:
- its originating PATHNAME is known (non NIL) and one of domestic PATHNAMES, - its originating PATHNAME is unknown, but the SYMBOL's home package is one of domestic PACKAGES.
Note that a definition for a domestic symbol, but originating in a foreign source file is considered foreign.

Package [`net.didierverna.declt.assess`], page 32.

Source [`definition.lisp`], page 19.

`escape` (*string*) [Function]
When STRING, escape it for Texinfo.

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

Source [`texi.lisp`], page 13.

`escape-anchor` (*string*) [Function]
Escape STRING for use as a Texinfo anchor name.
In addition to regular escaping, periods, commas, colons, and parenthesis are replaced with alternative Unicode characters.

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

Source [`texi.lisp`], page 13.

`escape-label` (*string*) [Function]
Escape STRING for use as a Texinfo anchor label.
In addition to regular escaping, colons are replaced with alternative Unicode characters.

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

Source [`texi.lisp`], page 13.

`escape-lambda-list` (*lambda-list*) [Function]
Escape safe LAMBDA-LIST for Texinfo.
This function expects a value from 'safe-lambda-list', which see. It returns a string properly escaped for Texinfo, apart from &-constructs which retain their original form, and @ref's and @t's which are already properly set.

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

Source [`texi.lisp`], page 13.

`file-components` (*module*) [Function]
Return the list of all file components found in MODULE's tree.

Package [`net.didierverna.declt.assess`], page 32.

Source [`assess.lisp`], page 27.

`file-node` (*definition context*) [Function]
Create and return a file DEFINITION node in CONTEXT.

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

Source [`asdf.lisp`], page 18.

`finalize` (*definitions packages pathnames*) [Function]
Finalize DEFINITIONS in domestic PACKAGES and PATHNAMES. For more information, see 'stabilize' and 'freeze'.

Package [`net.didierverna.declt.assess`], page 32.

Source [`finalize.lisp`], page 27.

`find-definition` (*object definitions*) [Function]
Find a definition for OBJECT in DEFINITIONS.

Package [`net.didierverna.declt.assess`], page 32.

Source [`definition.lisp`], page 19.

`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 13.

`freeze` (*definitions*) [Function]
Freeze DEFINITIONS.
Currently, this means:
- computing the definitions UIDs,
- potentially upgrading generic definitions to reader or writer definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`finalize.lisp`], page 27.

`funcoid-name` (*funcoid*) [Function]
Return FUNCOID's name, or NIL.
FUNCOID may be a function, a macro function, or a compiler macro function. Lambda expression are not considered as proper names, so NIL is returned.

Package [`net.didierverna.declt.assess`], page 32.

Source [`assess.lisp`], page 27.

`i-reader` (*stream subchar arg*) [Function]
Construct a call to 'defindent' by reading an argument list from STREAM. This dispatch macro character function is installed on #i in the NET.DIDIERVERNA.DECLT named readtable.

Package [`net.didierverna.declt.setup`], page 31.

Source [`readtable.lisp`], page 12.

`index` (*definition*) [Function]
Render DEFINITION's indexing command on *STANDARD-OUTPUT*.

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

Source [`doc.lisp`], page 15.

`itemize-list` (*list* &key *renderer kind format key*) [Function]
Render a LIST of items as part of an @itemize KIND environment.
KIND should be a string designator. It defaults to @bullet.
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. It defaults to `"~A"`.
- 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 13.

`load-system` (*system-name*) [Function]
Load ASDF SYSTEM-NAME in a manner suitable to extract documentation. Return the corresponding ASDF system.
SYSTEM-NAME is an ASDF system designator.

Package [`net.didierverna.declt.assess`], page 32.

Source [`assess.lisp`], page 27.

`long-title` (*definition*) [Function]
Return a long title for DEFINITION.
It is of the form "The <qualified safe name> <type name>".

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

Source [`doc.lisp`], page 15.

`make-all-file-definitions` (*definitions*) [Function]
Return a list of all file definitions for system DEFINITIONS.

Package [`net.didierverna.declt.assess`], page 32.

Source [`assess.lisp`], page 27.

`make-all-module-definitions` (*definitions*) [Function]
Return a list of all module definitions for system DEFINITIONS.

Package [`net.didierverna.declt.assess`], page 32.

Source [`assess.lisp`], page 27.

`make-all-package-definitions` (*file-definitions system-definitions*) [Function]
Return a list of all package definitions for FILE- and SYSTEM-DEFINITIONS. This list contains definitions for packages defined in the corresponding files, or for which the source is not found, but the name is of the form SYSTEM/... (case insensitive) for one of the corresponding systems.

Package [`net.didierverna.declt.assess`], page 32.

Source [`assess.lisp`], page 27.

`make-all-symbol-definitions` (*packages pathnames all-symbols-p*) [Function]
Return a list of all domestic symbol definitions.
If ALL-SYMBOLS-P, introspect all accessible symbols in the current Lisp environment. Otherwise (the default), limit introspection to the symbols from domestic PACKAGES.
Domesticity is defined in relation to domestic PACKAGES and PATHNAMES; see 'domesticp'.

Package [`net.didierverna.declt.assess`], page 32.

Source [`assess.lisp`], page 27.

`make-all-system-definitions` (*system*) [Function]
Return a list of all system definitions for SYSTEM.
The only guarantee is that the definition for SYSTEM comes first. The other considered systems are those found recursively in SYSTEM's dependencies, and located under SYSTEM's directory.
See 'subsystems' for more information.

Package [`net.didierverna.declt.assess`], page 32.

Source [`assess.lisp`], page 27.

`make-classoid-definition` (*symbol classoid packages pathnames*) [Function]
Make a new CLASSOID definition for SYMBOL.
Also create all slots definitions. The foreign status of the new classoid and its slots is computed from domestic PACKAGES and PATHNAMES.
The concrete class of the new definition (structure, class, or condition) depends on the kind of CLASSOID.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`make-clos-slot-definition` (*slot* &optional *foreign*) [Function]
Make a new CLOS SLOT definition, possibly FOREIGN.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`make-combination-definition` (*symbol combination* &optional *foreign*) [Function]
Make a new method COMBINATION definition for SYMBOL, possibly FOREIGN. The concrete class of the new definition depends on the COMBINATION type.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`make-compiler-macro-alias-definition` (*symbol* &optional *setf*) [Function]
Make a new compiler macro alias definition for (possibly SETF) SYMBOL.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`make-compiler-macro-definition` (*symbol compiler-macro* &rest *keys* &key *setf foreign*) [Function]
Make a new COMPILER-MACRO definition for SYMBOL.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`make-constant-definition` (*symbol*) [Function]
Make a new constant definition for SYMBOL.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`make-context` (&rest *keys* &key *locations default-values foreign-definitions declt-notice*) [Function]
Make a new rendering context.
The following keys are available.
- LOCATIONS: whether to hyperlink definitions to their locations. Currently supported values are NIL (the default), and :file-system. - DEFAULT-VALUES: whether to render default / standard values. Defaults to NIL.
- FOREIGN-DEFINITIONS: whether to render foreign definitions. Defaults to NIL.
- DECLT-NOTICE: whether to add a small credit paragraph to Declt. Possible values are NIL, :short, or :long (the default).

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

Source [`doc.lisp`], page 15.

`make-expander-definition` (*symbol expander* &optional *foreign*) [Function]
Make a new setf EXPANDER definition for SYMBOL, possibly foreign.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`make-file-definition` (*file* &optional *foreign*) [Function]
Make a new FILE definition.
The concrete class of the new definition depends on the kind of FILE.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

`make-function-alias-definition` (*symbol* &optional *setf*) [Function]
make a new function alias definition for (possibly SETF) SYMBOL.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`make-generic-function-definition` (*symbol generic* &rest *keys* &key *setf foreign*) [Function]
Make a new GENERIC function definition for (SETF) SYMBOL, possibly FOREIGN.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`make-macro-alias-definition` (*symbol*) [Function]
Make a new macro alias definition for SYMBOL.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`make-macro-definition` (*symbol macro* &optional *foreign*) [Function]
Make a new MACRO definition for SYMBOL.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`make-method-definition` (*method* &optional *foreign*) [Function]
Make a new METHOD definition, possibly FOREIGN.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`make-module-definition` (*module* &optional *foreign*) [Function]
Make a new MODULE definition.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

`make-node` (&key *name synopsis section-type section-name next previous up children before-menu-contents after-menu-contents*) [Function]

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

Source [`texi.lisp`], page 13.

`make-ordinary-function-definition` (*symbol function* &rest *keys* &key *setf foreign*) [Function]
Make a new ordinary FUNCTION definition for (SETF) SYMBOL, possibly FOREIGN.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`make-package-definition` (*package* &optional *foreign*) [Function]
Make a new PACKAGE definition, possibly FOREIGN.

Package [`net.didierverna.declt.assess`], page 32.

Source [`package.lisp`], page 24.

`make-report` (*system-name*) [Function]
Make a new report for SYSTEM-NAME (an ASDF system designator).

Package [`net.didierverna.declt.assess`], page 32.

Source [`assess.lisp`], page 27.

`make-special-definition` (*symbol*) [Function]
Make a new special variable definition for SYMBOL.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`make-symbol-definitions` (*symbol packages pathnames*) [Function]
Make and return a list of all existing domestic definitions for SYMBOL. Domesticity is defined in relation to domestic PACKAGES and PATHNAMES; see 'domesticp'.

Package [`net.didierverna.declt.assess`], page 32.

Source [`assess.lisp`], page 27.

`make-symbol-macro-definition` (*symbol*) [Function]
Make a new symbol macro definition for SYMBOL.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`make-system-definition` (*system* &optional *foreign*) [Function]
Make a new SYSTEM definition.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

`make-system-file-definition` (*system*) [Function]
Make a new system file definition for SYSTEM.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

`make-system-file-definitions` (*systems*) [Function]
Make a list of system file definitions for SYSTEMS.
Multiple systems may be defined in the same file. There is however only one definition for each file.

Package [`net.didierverna.declt.assess`], page 32.

Source [`asdf.lisp`], page 25.

`make-type-definition` (*symbol expander*) [Function]
Make a new type definition for SYMBOL.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`make-typed-structure-slot-definition` (*slot* &optional *foreign*) [Function]
Make a new typed structure SLOT definition, possibly FOREIGN.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`merge-expander-p` (*definition expander*) [Function]
Return T if function DEFINITION and setf EXPANDER can be documented jointly.

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

Source [`symbol.lisp`], page 15.

`merge-generic-writer` (*reader writer*) [Function]
Check if WRITER generic definition can be documented jointly with READER. If so, return the generalized Boolean value of 'merge-methods', which see.

Merging is only attempted on generic functions defined exclusively via slot :accessor keywords. For merging to actually occur, there must not exist any property specific to only one definition, or different between the two (no related expander information, same method combination, same docstring, etc.). The only exception is their lambda lists.

The same conditions apply to methods, which definitions are also merged. Only unqualified methods must exist. Standalone reader and writer methods are still permitted.

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

Source [`symbol.lisp`], page 15.

`merge-methods` (*reader writer*) [Function]
Attempt to merge READER and WRITER generic definitions methods. See 'merge-generic-writer' for the exact conditions under which merging may occur. If merging is possible, return a list of three values:
1. a list of the form ((READER-METHOD . WRITER-METHOD) ...) for associated reader and writer methods,
2. a list of standalone readers,
3. a list of standalone writers,
Otherwise, return NIL.

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

Source [`symbol.lisp`], page 15.

`merge-ordinary-writer` (*reader writer*) [Function]
Return WRITER if it can be documented jointly with READER.

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

Source [`symbol.lisp`], page 15.

`method-name` (*method*) [Function]
Return METHOD's canonical name.
Return a second value of T if METHOD is in fact a SETF one.

Package [`net.didierverna.declt.assess`], page 32.

Source [`symbol.lisp`], page 20.

`module-components` (*module*) [Function]
Return the list of all module components found in MODULE's tree.

Package [`net.didierverna.declt.assess`], page 32.

Source [`assess.lisp`], page 27.

`new-funcoid-definition` (*name packages pathnames*) [Function]
Return a new macro or function definition for NAME, or NIL. PACKAGES and PATHNAMES are used to determine domesticity.

Package [`net.didierverna.declt.assess`], page 32.

Source [`finalize.lisp`], page 27.

`new-generic-definition` (*generic packages pathnames*) [Function]
Make a new foreign GENERIC function definition. PACKAGES and PATHNAMES are used to determine domesticity.

Package [`net.didierverna.declt.assess`], page 32.

Source [`finalize.lisp`], page 27.

`node-after-menu-contents` (*instance*) [Reader]
`(setf node-after-menu-contents)` (*instance*) [Writer]

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

Source [`texi.lisp`], page 13.

Target Slot
[`after-menu-contents`], page 142.

`node-before-menu-contents` (*instance*) [Reader]
`(setf node-before-menu-contents)` (*instance*) [Writer]

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

Source [`texi.lisp`], page 13.

Target Slot
[`before-menu-contents`], page 142.

`node-children` (*instance*) [Reader]
`(setf node-children)` (*instance*) [Writer]

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

Source [`texi.lisp`], page 13.

Target Slot
[`children`], page 142.

`node-name` (*instance*) [Reader]
`(setf node-name)` (*instance*) [Writer]

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

Source [`texi.lisp`], page 13.

Target Slot
[`name`], page 141.

`node-next` (*instance*) [Reader]
`(setf node-next)` (*instance*) [Writer]

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

Source [`texi.lisp`], page 13.

Target Slot
[`next`], page 142.

`node-p` (*object*) [Function]

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

Source [`texi.lisp`], page 13.

`node-previous` (*instance*) [Reader]
`(setf node-previous)` (*instance*) [Writer]

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

Source [`texi.lisp`], page 13.

Target Slot
[`previous`], page 142.

`node-section-name` (*instance*) [Reader]
`(setf node-section-name)` (*instance*) [Writer]

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

Source [`texi.lisp`], page 13.

Target Slot
[`section-name`], page 142.

`node-section-type` (*instance*) [Reader]
`(setf node-section-type)` (*instance*) [Writer]

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

Source [`texi.lisp`], page 13.

Target Slot
[`section-type`], page 142.

`node-synopsis` (*instance*) [Reader]
`(setf node-synopsis)` (*instance*) [Writer]

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

Source [`texi.lisp`], page 13.

Target Slot
[`synopsis`], page 141.

`node-up` (*instance*) [Reader]
`(setf node-up)` (*instance*) [Writer]

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

Source [`texi.lisp`], page 13.

Target Slot
[`up`], page 142.

`one-liner-p` (*string*) [Function]
Return T if STRING is non empty and does not span multiple lines.

Package [`net.didierverna.declt.assess`], page 32.

Source [`util.lisp`], page 19.

`package-external-symbols` (*package*) [Function]
Return the list of PACKAGE's external symbols.

Package [`net.didierverna.declt.assess`], page 32.

Source [`package.lisp`], page 24.

`package-internal-symbols` (*package*) [Function]
Return the lists of PACKAGE's internal and external symbols as two values.

Package [`net.didierverna.declt.assess`], page 32.

Source [`package.lisp`], page 24.

`package-symbols` (*package*) [Function]
Return the list of symbols from home PACKAGE.

Package [`net.didierverna.declt.assess`], page 32.

Source [`assess.lisp`], page 27.

`parse-contact(s)` (*contact(s)*) [Function]
Parse CONTACT(S) as either a contact string, or a list of such. Return a list of parsed contacts. See 'parse-contact-string' for more information.

Package [`net.didierverna.declt.assess`], page 32.

Source [`util.lisp`], page 19.

`parse-contact-string` (*string*) [Function]
Parse STRING of the form "My Name <my@address>".
Both name and address are optional. If only an address is provided, the angle brackets may be omitted.
If neither a name nor an address can be extracted, return NIL. Otherwise, return the list ("My Name" . "my@address"). In such a case, either the CAR or the CDR may be null, but not both.

Package [`net.didierverna.declt.assess`], page 32.

Source [`util.lisp`], page 19.

`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 13.

`reference` (*definition context* &optional *short punctuation*) [Function]
Render a possibly SHORT DEFINITION's reference in CONTEXT.
Rendering is done on *STANDARD-OUTPUT*.
When DEFINITION is foreign and CONTEXT disables their rendering, the produced reference is just text. Otherwise, an actual link is created.
Unless SHORT, the DEFINITION type is advertised after the reference itself. When SHORT, the reference is followed by a PUNCTUATION character (a dot by default) or NIL.

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

Source [`doc.lisp`], page 15.

`release-status-number` (*release-status*) [Function]

Package [`net.didierverna.declt.setup`], page 31.

Source [`version.lisp`], page 12.

`render-definition-core` (*definition context*) [Function]
Render DEFINITION's documentation core in CONTEXT.
More specifically, render DEFINITION's package and source file references.

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

Source [`symbol.lisp`], page 15.

`render-dependencies` (*dependencies context* &optional *prefix*) [Function]
Render COMPONENT's DEPENDENCIES in CONTEXT, optionally PREFIXing the title.

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

Source [`asdf.lisp`], page 18.

`render-dependency` (*dependency context*) [Function]
Render a resolved DEPENDENCY specification in CONTEXT. See 'resolve-dependency-specification' for more information.

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

Source [`asdf.lisp`], page 18.

`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 15.

`render-header` (*report file-name info-name info-category declt-notice current-time-string*) [Function]
Render the header of the Texinfo file.

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

Source [`declt.lisp`], page 19.

`render-headline` (*definition*) [Function]
Render a headline for DEFINITION. Also anchor and index it.

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

Source [`symbol.lisp`], page 15.

`render-node` (*node level*) [Function]
Render NODE at LEVEL and all its children at LEVEL+1.

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

Source [`texi.lisp`], page 13.

`render-package-reference` (*definition context* &optional *force*) [Function]
Render a reference to DEFINITION's home package definition in CONTEXT. When FORCE, render a reference to the Common Lisp package, even if CONTEXT says otherwise.
Possibly render an "uninterned" mention instead of an actual reference, when there is no home package to reference.

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

Source [`symbol.lisp`], page 15.

`render-references` (*title definitions context* &optional *short*) [Function]
Render an enTITLEd list of [SHORT] references to DEFINITIONS in CONTEXT. See 'reference' for the meaning of SHORT. The list is rendered in an itemized table item, unless there is only one definition in which case it appears directly as the table item's contents.
Rendering is done on *standard-output*.

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

Source [`doc.lisp`], page 15.

`render-text` (*text*) [Function]
Render TEXT for Texinfo.
Rendering is done on *standard-output*.
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 13.

`render-top-node` (*node*) [Function]
Render the whole nodes hierarchy starting at toplevel NODE.

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

Source [`texi.lisp`], page 13.

`reorder-dependency-def` (*dependency-def*) [Function]
Reorder information in DEPENDENCY-DEF so that the system is always first. More specifically:
- simple component names are returned as-is,
- :version expressions are returned as (system :version version-specifier), - :feature expressions are returned as (... :feature feature-expression), - :require expressions are returned as (system :require).

Note that because a feature expression is defined recursively, the first element in the reordered list may be another reordered sub-list rather than a simple component name directly. In any case, the system name will always be in the deepest first position.

Package [`net.didierverna.declt.assess`], page 32.

Source [`util.lisp`], page 19.

`reordered-dependency-def-system` (*reordered-dependency-def*) [Function]
Extract the system name from REORDERED-DEPENDENCY-DEF. See 'reorder-dependency-def' for more information.

Package [`net.didierverna.declt.assess`], page 32.

Source [`util.lisp`], page 19.

`resolve-dependency-specification` (*specification component definitions foreign*) [Function]
Resolve dependency SPECIFICATION for (FOREIGN) COMPONENT in DEFINITIONS. SPECIFICATION must already be reordered (see 'reorder-dependency-def' for more information). The specification's component name is replaced with its corresponding definition. A foreign definition may be created in the process.

If such a definition is neither found, nor created, return NIL. Otherwise, return a list of the updated specification (suitable to MAPCAN).

Package [`net.didierverna.declt.assess`], page 32.

Source [`finalize.lisp`], page 27.

`reveal` (*string*) [Function]
Return a copy of STRING with blanks revealed.
If STRING is empty or null, use the empty set symbol. Otherwise, each blank character is replaced with a visible Unicode representation. See '*blanks*' for more information.

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

Source [`doc.lisp`], page 15.

`safe-lambda-list` (*lambda-list* &optional *safe-specializers*) [Function]
Return a safe LAMBDA-LIST, suitable to pass to Texinfo.
The original lambda-list's structure is preserved, but all symbols are converted to revealed strings, and initform / supplied-p data is removed. &whole, &environment, and &aux parts are removed as they don't provide any information on the funcoid's usage. SAFE-SPECIALIZERS is provided for method LAMBDA-LISTs. See 'safe-specializers' for more information. Mandatory arguments associated with a non-nil safe specializer are listed together.

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

Source [`symbol.lisp`], page 15.

`safe-specializers` (*definition context*) [Function]
Return a list of safe specializers for method DEFINITION in CONTEXT.
A safe specializer is the printed form of either a reference to a class definition, or an EQL specializer's type name.

Unless the CONTEXT specifies otherwise, T specializers are replaced by NIL to indicate that they are not to be advertized.

For setf and writer definitions, only the specializers rest is used, as these methods get the new value as their first argument.

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

Source [`symbol.lisp`], page 15.

`select-keys` (*keys* &rest *selected*) [Function]
Return a new property list from KEYS with only SELECTED ones.

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

Source [`declt.lisp`], page 19.

`source-by-name` (*name type*) [Function]
Return source pathname for NAMEd object of TYPE.

Package [`net.didierverna.declt.assess`], page 32.

Source [`util.lisp`], page 19.

`source-by-object` (*object*) [Function]
Return OBJECT's source pathname.

Package [`net.didierverna.declt.assess`], page 32.

Source [`util.lisp`], page 19.

`stabilize-clos-classoid-slot` (*definition definitions packages pathnames*) [Function]
Compute CLOS classoid slot DEFINITION's reader and writer definitions. This function is used for regular class and condition slots.

Package [`net.didierverna.declt.assess`], page 32.

Source [`finalize.lisp`], page 27.

`stabilize-clos-structure-slot` (*definition definitions packages pathnames*) [Function]
Compute CLOS structure slot DEFINITION's reader and writer definitions.

Package [`net.didierverna.declt.assess`], page 32.

Source [`finalize.lisp`], page 27.

`sub-component-p` (*component directory*) [Function]
Return T if COMPONENT can be found under DIRECTORY.

Package [`net.didierverna.declt.assess`], page 32.

Source [`assess.lisp`], page 27.

`subsystem` (*name system directory*) [Function]
Return NAME'd SYSTEM dependency if found under DIRECTORY, or nil.

Package [`net.didierverna.declt.assess`], page 32.

Source [`assess.lisp`], page 27.

`subsystems` (*system directory*) [Function]
Return the list of SYSTEM and all its dependencies found under DIRECTORY. All dependencies are descended recursively. Both :defsystem-depends-on and :depends-on are included. Potential duplicates are removed.

Package [`net.didierverna.declt.assess`], page 32.

Source [`assess.lisp`], page 27.

`system-dependencies` (*system*) [Function]
Return all system names from SYSTEM dependencies. This includes both :defsystem-depends-on and :depends-on.

Package [`net.didierverna.declt.assess`], page 32.

Source [`assess.lisp`], page 27.

`validate-email` (*string*) [Function]
Check that STRING is of the form nonblank@nonblank, after trimming. Return that string, or issue a warning and return NIL.

Package [`net.didierverna.declt.assess`], page 32.

Source [`util.lisp`], page 19.

6.2.4 Generic functions

`category-name` (*definition*) [Generic Function]
Return DEFINITION's category name.

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

Source [`doc.lisp`], page 15.

Methods

`category-name` ((*definition* [`system-definition`], page 100)) [Method]
Return "system"

Source [`asdf.lisp`], page 18.

`category-name` ((*definition* [`module-definition`], page 90)) [Method]
Return "module"

Source [`asdf.lisp`], page 18.

`category-name` ((*definition* [`file-definition`], page 84)) [Method]
Return "file"

Source [`asdf.lisp`], page 18.

`category-name` ((*definition* [`package-definition`], page 92)) [Method]
Return "package".

Source [`package.lisp`], page 18.

`category-name` ((*definition [`alias-definition`], page 75*)) [Method]
Return the category name of alias DEFINITION's referee.

Source [`symbol.lisp`], page 15.

`category-name` ((*definition [`slot-definition`], page 97*)) [Method]
Return "slot".

Source [`symbol.lisp`], page 15.

`category-name` ((*definition [`class-definition`], page 76*)) [Method]
Return "class".

Source [`symbol.lisp`], page 15.

`category-name` ((*definition [`condition-definition`], page 81*)) [Method]
Return "condition".

Source [`symbol.lisp`], page 15.

`category-name` ((*definition [`structure-definition`], page 99*)) [Method]
Return "structure".

Source [`symbol.lisp`], page 15.

`category-name` ((*definition [`writer-method-definition`], page 104*)) [Method]
Return "writer method".

Source [`symbol.lisp`], page 15.

`category-name` ((*definition [`reader-method-definition`], page 93*)) [Method]
Return "reader method".

Source [`symbol.lisp`], page 15.

`category-name` ((*definition [`method-definition`], page 90*)) [Method]
Return "method".

Source [`symbol.lisp`], page 15.

`category-name` ((*definition [`combination-definition`], page 79*)) [Method]
Return "method combination".

Source [`symbol.lisp`], page 15.

`category-name` ((*definition [`generic-writer-definition`], page 88*)) [Method]
Return "generic writer".

Source [`symbol.lisp`], page 15.

`category-name` ((*definition [`generic-reader-definition`], page 87*)) [Method]
Return "generic reader".

Source [`symbol.lisp`], page 15.

`category-name` ((*definition* *[`generic-function-definition`], page 86*)) [Method]
Return "generic function".

Source [`symbol.lisp`], page 15.

`category-name` ((*definition* *[`ordinary-writer-definition`], page 92*)) [Method]
Return "writer".

Source [`symbol.lisp`], page 15.

`category-name` ((*definition* *[`ordinary-reader-definition`], page 92*)) [Method]
Return "reader".

Source [`symbol.lisp`], page 15.

`category-name` ((*definition* *[`ordinary-function-definition`], page 91*)) [Method]
Return "function".

Source [`symbol.lisp`], page 15.

`category-name` ((*expander* *[`expander-definition`], page 83*)) [Method]
Return "setf expander".

Source [`symbol.lisp`], page 15.

`category-name` ((*definition* *[`type-definition`], page 102*)) [Method]
Return "type".

Source [`symbol.lisp`], page 15.

`category-name` ((*definition* *[`compiler-macro-definition`], page 80*)) [Method]
Return "compiler macro".

Source [`symbol.lisp`], page 15.

`category-name` ((*definition* *[`macro-definition`], page 89*)) [Method]
Return "macro".

Source [`symbol.lisp`], page 15.

`category-name` ((*definition* *[`symbol-macro-definition`], page 100*)) [Method]
Return "symbol macro".

Source [`symbol.lisp`], page 15.

`category-name` ((*definition* *[`special-definition`], page 99*)) [Method]
Return "special variable".

Source [`symbol.lisp`], page 15.

`category-name` ((*definition* *[`constant-definition`], page 82*)) [Method]
Return "constant".

Source [`symbol.lisp`], page 15.

`declt-notice` (*object*) [Generic Reader]

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

Methods

`declt-notice` ((*context* [`context`], *page 142*)) [Reader Method]
whether to add a small credit paragraph about Declt. Possible values are NIL, :short, or :long (the default).

Source [`doc.lisp`], page 15.

Target Slot
[`declt-notice`], page 143.

`default-values` (*object*) [Generic Reader]

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

Methods

`default-values` ((*context* [`context`], *page 142*)) [Reader Method]
Whether to render default / standard values.

Source [`doc.lisp`], page 15.

Target Slot
[`default-values`], page 143.

`document` (*definition context* &key *inline writer merged-methods merge-expander* &allow-other-keys) [Generic Function]
Render DEFINITION's documentation in CONTEXT.

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

Source [`doc.lisp`], page 15.

Method Combination
[`document`], page 141.

Methods

`document` *:open* ((*definition* [`system-definition`], *page 100*) *context* &key) [Method]
Render DEFINITION's system-specific bits in CONTEXT.

Source [`asdf.lisp`], page 18.

`document` *:close* ((*definition* [`module-definition`], *page 90*) *context* &key) [Method]
Render module DEFINITION's references in CONTEXT.

Source [`asdf.lisp`], page 18.

`document` *:close* ((*definition* [`lisp-file-definition`], *page 88*) *context* &key) [Method]
Render lisp file DEFINITION's references in CONTEXT.

Source [`asdf.lisp`], page 18.

`document` ((*definition* [`component-definition`], *page 80*) *context* &key) [Method]
Render ASDF component DEFINITION's documentation in CONTEXT.

Source [`asdf.lisp`], page 18.

`document` *:around* ((*definition* [`component-definition`], *page 80*) *context* &key) [Method]
Anchor, index and document component DEFINITION in CONTEXT. Documentation is done in a @table environment.

Source [`asdf.lisp`], page 18.

`document` ((*definition* [`package-definition`], *page 92*) *context* &key) [Method]
Render package DEFINITION's documentation in context.

Source [`package.lisp`], page 18.

`document` ((*definition* [`alias-definition`], *page 75*) *context* &key) [Method]
Render alias DEFINITION's documentation in CONTEXT.

Source [`symbol.lisp`], page 15.

`document` ((*definition* [`typed-structure-slot-definition`], *page 103*) *context* &key) [Method]
Render typed structure slot DEFINITION's documentation in CONTEXT.
More specifically, render DEFINITION's reader and writer references.

Source [`symbol.lisp`], page 15.

`document` ((*definition* [`clos-slot-definition`], *page 78*) *context* &key) [Method]
Render CLOS slot DEFINITION's documentation in CONTEXT.
More specifically, render DEFINITION's reader and writer references.

Source [`symbol.lisp`], page 15.

`document` ((*definition* [`slot-definition`], *page 97*) *context* &key) [Method]
Render slot DEFINITION's documentation in context.
More specifically, render DEFINITION's value type, and for CLOS slots render allocation, initform, and initargs.

Source [`symbol.lisp`], page 15.

`document` ((*definition* [`typed-structure-definition`], *page 102*) *context* &key) [Method]
Render typed structure DEFINITION's type documentation in CONTEXT.

Source [`symbol.lisp`], page 15.

`document` ((*definition* [`clos-classoid-mixin`], *page 77*) *context* &key) [Method]
Render CLOS classoid mixin DEFINITION's documentation in CONTEXT.
More specifically, render DEFINITION's direct superclasses, subclasses, methods, and initargs references.

Source [`symbol.lisp`], page 15.

`document` *:close* ((*definition* [`classoid-definition`], *page 76*) *context* &key) [Method]
Close classoid DEFINITION's documentation environment in CONTEXT.
More specifically:

- close the @table environment,
- close the @deftp environment.

Source [symbol.lisp], page 15.

`document` ((*definition* [`classoid-definition`], *page 76*) *context* &key) [Method]

Render classoid DEFINITION's direct slots references in CONTEXT.

Source [symbol.lisp], page 15.

`document` *:open* ((*definition* [`classoid-definition`], *page 76*) *context* &key) [Method]

Open classoid DEFINITION's documentation environment in CONTEXT.
More specifically:
- open a @deftp environment,
- anchor and index DEFINITION,
- render DEFINITION's docstring,
- open a @table environment,
- render DEFINITIONS's core documentation.

Source [symbol.lisp], page 15.

`document` *:open* ((*definition* [`method-definition`], *page 90*) *context* &key *inline writer*) [Method]

Open method DEFINITION's documentation in CONTEXT.
More specifically:
- open a @deffn environment, possibly merging a WRITER method,
- anchor and index DEFINITION,
- render DEFINITION's docstring,
- open a @table environment,
- render DEFINITION's source file.

When INLINE, the method definition is documented within its owner's documentation. In such a case, the package reference is not rendered (as it is the same as the owner's), and the source file is only referenced if different from that of the owner's.

Source [symbol.lisp], page 15.

`document` ((*definition* [`short-combination-definition`], *page 96*) *context* &key) [Method]

Render short method combination DEFINITION's documentation in CONTEXT.

Source [symbol.lisp], page 15.

`document` ((*definition* [`combination-definition`], *page 79*) *context* &key) [Method]

Render method combination DEFINITION's client references in CONTEXT.

Source [symbol.lisp], page 15.

`document` *:around* ((*definition* [`generic-writer-definition`], *page 88*) *context* &key) [Method]

Prevent generic writer DEFINITION from being documented when merging.

Source [symbol.lisp], page 15.

`document` *:around* ((*definition* *[`generic-reader-definition`], page 87*) *context* &rest *args* &key) [Method]

Check for potential writer merging with generic reader DEFINITION.

Source [`symbol.lisp`], page 15.

`document` *:close* ((*definition* *[`generic-function-definition`], page 86*) *context* &key *merged-methods*) [Method]

Close generic function DEFINITION's documentation.
More specifically:
- render DEFINITION's method references, possibly merging readers and writers, - close the @table environment,
- close the @deffn environment.

Source [`symbol.lisp`], page 15.

`document` ((*definition* *[`generic-function-definition`], page 86*) *context* &key) [Method]

Render generic function DEFINITION's combination reference in CONTEXT.

Source [`symbol.lisp`], page 15.

`document` *:around* ((*definition* *[`ordinary-writer-definition`], page 92*) *context* &key) [Method]

Prevent ordinary writer DEFINITION from being documented when merging.

Source [`symbol.lisp`], page 15.

`document` *:around* ((*definition* *[`ordinary-reader-definition`], page 92*) *context* &rest *args* &key) [Method]

Check for potential writer merging with ordinary reader DEFINITION.

Source [`symbol.lisp`], page 15.

`document` ((*definition* *[`short-expander-definition`], page 97*) *context* &key) [Method]

Render short expander DEFINITION's standalone writer reference in CONTEXT.

Source [`symbol.lisp`], page 15.

`document` ((*definition* *[`expander-definition`], page 83*) *context* &key) [Method]

Render setf expander DEFINITION's standalone reader reference in CONTEXT.

Source [`symbol.lisp`], page 15.

`document` *:around* ((*definition* *[`long-expander-definition`], page 89*) *context* &key) [Method]

Prevent long expander DEFINITION from being documented when merging.

Source [`symbol.lisp`], page 15.

`document` *:close* ((*definition* *[*`type-definition`*], page 102*) *context* &key) [Method]

Close type DEFINITION's documentation environment in CONTEXT. More specifically:
- close the @table environment,
- close the @deftp environment.

Source [`symbol.lisp`], page 15.

`document` *:open* ((*definition* *[*`type-definition`*], page 102*) *context* &key) [Method]

Open type DEFINITION's documentation environment in CONTEXT. More specifically:
- open a @deftp environment,
- anchor and index DEFINITION,
- render DEFINITION's docstring,
- open a @table environment,
- render DEFINITION's core.

Source [`symbol.lisp`], page 15.

`document` ((*definition* *[*`accessor-mixin`*], page 74*) *context* &key) [Method]

Render accessor mixin DEFINITION's target slot reference in CONTEXT.

Source [`symbol.lisp`], page 15.

`document` ((*definition* *[*`setfable-funcoid-definition`*], page 95*) *context* &key *merge-expander*) [Method]

Render setfable funcoid DEFINITION's expanders information in CONTEXT. More specifically:
- render a reference to a set expander for DEFINITION, unless the definitions are merged,
- render references to all setf expanders expanding to DEFINITION.

Source [`symbol.lisp`], page 15.

`document` *:open* ((*definition* *[*`setfable-funcoid-definition`*], page 95*) *context* &key *merge-expander writer*) [Method]

Open setfable funcoid DEFINITION's documentation environment in CONTEXT. More specifically:
- open a @deffn environment, possibly merging a related setf expander or writer,
- anchor and index DEFINITION,
- render DEFINITION's docstring,
- open a @table environment,
- render DEFINITION's core documentation.

Source [`symbol.lisp`], page 15.

`document` *:around* ((*definition* *[*`setfable-funcoid-definition`*], page 95*) *context* &rest *args* &key) [Method]

Check for potential expander merging of setfable funcoid DEFINITION.

Source [`symbol.lisp`], page 15.

`document` *:close* ((*definition [*`funcoid-definition`*], page 85*) *context* &key) [Method]
Close funcoid DEFINITION's documentation environment in CONTEXT.
More specifically:
- close the @table environment,
- close the @deffn environment.

Source [`symbol.lisp`], page 15.

`document` *:open* ((*definition [*`funcoid-definition`*], page 85*) *context* &key) [Method]
Open funcoid DEFINITION's documentation environment in CONTEXT.
More specifically:
- open a @deffn environment,
- anchor and index DEFINITION,
- render DEFINITION's docstring,
- open a @table environment,
- render DEFINITION's core documentation.

This is the default method.

Source [`symbol.lisp`], page 15.

`document` *:close* ((*definition [*`varoid-definition`*], page 104*) *context* &key) [Method]
Close varoid DEFINITION's documentation environment in CONTEXT.
More specifically:
- close the @table environment,
- close the @defvr environment.

Source [`symbol.lisp`], page 15.

`document` ((*definition [*`varoid-definition`*], page 104*) *context* &key) [Method]
Render varoid DEFINITION's documentation in CONTEXT.
More specifically, render DEFINITION's package and source file references. As a special exception, slots don't reference their package, unless it differs from the slot's owner package, and never reference their source file, which is the same as their owner.

Source [`symbol.lisp`], page 15.

`document` *:open* ((*definition [*`varoid-definition`*], page 104*) *context* &key) [Method]
Open varoid DEFINITIONS's documentation environment in CONTEXT.
More specifically:
- open a @defvr environment,
- anchor and index DEFINITION,
- render DEFINITION's docstring,
- open a @table environment.

Source [`symbol.lisp`], page 15.

`document` *:around* (*definition context* &key) [Method]
Check whether to render foreign DEFINITIONs.

`external-symbols` (*object*) [Generic Reader]
`(setf external-symbols)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`external-symbols` ((*package-definition* *[`package-definition`], page 92*)) [Reader Method]
`(setf external-symbols)` ((*package-definition* *[`package-definition`], page 92*)) [Writer Method]
The list of corresponding external symbols.

Source [`package.lisp`], page 24.

Target Slot
[`external-symbols`], page 93.

`foreign-definitions` (*object*) [Generic Reader]

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

Methods

`foreign-definitions` ((*context* *[`context`], page 142*)) [Reader Method]
Whether to render foreign definitions.

Source [`doc.lisp`], page 15.

Target Slot
[`foreign-definitions`], page 143.

`index-command-name` (*definition*) [Generic Function]
Return DEFINITION's index command name.

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

Source [`doc.lisp`], page 15.

Methods

`index-command-name` ((*definition* *[`system-definition`], page 100*)) [Method]
Return "systemindex"

Source [`asdf.lisp`], page 18.

`index-command-name` ((*definition* *[`module-definition`], page 90*)) [Method]
Return "moduleindex"

Source [`asdf.lisp`], page 18.

`index-command-name` ((*definition* *[`file-definition`], page 84*)) [Method]
Return "fileindex"

Source [`asdf.lisp`], page 18.

`index-command-name` ((*definition* *[`package-definition`], page 92*)) [Method]
Return "packageindex".

Source [`package.lisp`], page 18.

`index-command-name` ((*definition* [`alias-definition`], *page 75*)) [Method]
Return the index command name of alias DEFINITION's referee.

Source [`symbol.lisp`], page 15.

`index-command-name` ((*definition* [`slot-definition`], *page 97*)) [Method]
Return "slotsubindex".

Source [`symbol.lisp`], page 15.

`index-command-name` ((*definition* [`class-definition`], *page 76*)) [Method]
Return "classsubindex".

Source [`symbol.lisp`], page 15.

`index-command-name` ((*definition* [`condition-definition`], *page 81*)) [Method]
Return "conditionsubindex".

Source [`symbol.lisp`], page 15.

`index-command-name` ((*definition* [`structure-definition`], *page 99*)) [Method]
Return "structuresubindex".

Source [`symbol.lisp`], page 15.

`index-command-name` ((*definition* [`method-definition`], *page 90*)) [Method]
Return "methodsubindex".

Source [`symbol.lisp`], page 15.

`index-command-name` ((*definition* [`combination-definition`], *page 79*)) [Method]
Return "combinationsubindex".

Source [`symbol.lisp`], page 15.

`index-command-name` ((*definition* [`generic-function-definition`], *page 86*)) [Method]
Return "genericsubindex".

Source [`symbol.lisp`], page 15.

`index-command-name` ((*definition* [`ordinary-function-definition`], *page 91*)) [Method]
Return "functionsubindex".

Source [`symbol.lisp`], page 15.

`index-command-name` ((*expander* [`expander-definition`], *page 83*)) [Method]
Return "expandersubindex".

Source [`symbol.lisp`], page 15.

`index-command-name` ((*definition* *[*`type-definition`*], page 102*)) [Method]
Return "typesubindex".

Source [`symbol.lisp`], page 15.

`index-command-name` ((*definition* *[*`compiler-macro-definition`*], page 80*)) [Method]
Return "compilermacrosubindex".

Source [`symbol.lisp`], page 15.

`index-command-name` ((*definition* *[*`macro-definition`*], page 89*)) [Method]
Return "macrosubindex".

Source [`symbol.lisp`], page 15.

`index-command-name` ((*definition* *[*`symbol-macro-definition`*], page 100*)) [Method]
Return "symbolmacrosubindex".

Source [`symbol.lisp`], page 15.

`index-command-name` ((*definition* *[*`special-definition`*], page 99*)) [Method]
Return "specialsubindex".

Source [`symbol.lisp`], page 15.

`index-command-name` ((*definition* *[*`constant-definition`*], page 82*)) [Method]
Return "constantsubindex".

Source [`symbol.lisp`], page 15.

`internal-symbols` (*object*) [Generic Reader]
`(setf internal-symbols)` (*object*) [Generic Writer]

Package [`net.didierverna.declt.assess`], page 32.

Methods

`internal-symbols` ((*package-definition* *[*`package-definition`*], page 92*)) [Reader Method]
`(setf internal-symbols)` ((*package-definition* *[*`package-definition`*], page 92*)) [Writer Method]
The list of corresponding internal symbols.

Source [`package.lisp`], page 24.

Target Slot
[`internal-symbols`], page 93.

`locations` (*object*) [Generic Reader]

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

Methods

`locations` ((*context [`context`], page 142*)) [Reader Method]
Whether to hyperlink definitions to their locations.
Currently supported values are NIL (the default), and :file-system.

Source [`doc.lisp`], page 15.

Target Slot
[`locations`], page 143.

`safe-name` (*definition* &optional *qualified*) [Generic Function]
Return DEFINITION's safe name, possibly QUALIFIED.
Safe names have blank characters replaced with visible Unicode symbols. See 'reveal' for more information.

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

Source [`doc.lisp`], page 15.

Methods

`safe-name` *:around* ((*definition [`file-definition`], page 84*) &optional *qualify*) [Method]
Append DEFINITION's file extension at the end, when applicable.

Source [`asdf.lisp`], page 18.

`safe-name` ((*definition [`component-definition`], page 80*) &optional *qualified*) [Method]
Reveal component DEFINITION's name, possibly QUALIFIED.
A QUALIFIED component's name is of the form "path/to/component", each element being the name of a component's parent.

Source [`asdf.lisp`], page 18.

`safe-name` *:around* ((*definition [`slot-definition`], page 97*) &optional *qualified*) [Method]
When QUALIFIED, prepend slot DEFINITION's classoid safe name.

Source [`symbol.lisp`], page 15.

`safe-name` *:around* ((*definition [`method-definition`], page 90*) &optional *qualified*) [Method]
When QUALIFIED, append method DEFINITION's qualifiers and specializers.

Source [`symbol.lisp`], page 15.

`safe-name` ((*definition [`symbol-definition`], page 99*) &optional *qualified*) [Method]
Reveal symbol DEFINITION's name, possibly QUALIFIED.
A QUALIFIED name is of the form "package:[:]symbol", maybe in a setf list.
Uninterned symbols are denoted by the ∅ package.

Source [`symbol.lisp`], page 15.

`safe-name` ((*definition [`definition`], page 82*) &optional *qualified*) [Method]
Reveal unqualifiable DEFINITION's name. This is the default method.

`source-pathname` (*definition*) [Generic Function]
Return DEFINITION's source pathname.

Package [`net.didierverna.declt.assess`], page 32.

Source [`definition.lisp`], page 19.

Methods

`source-pathname` ((*definition* [`component-definition`], *page 80*)) [Method]
Return component DEFINITION's source pathname.
This actually is the corresponding system's source file.

Source [`asdf.lisp`], page 25.

`source-pathname` ((*definition* [`alias-definition`], *page 75*)) [Method]
Return NIL.
Aliases are defined dynamically so it's impossible to locate the code being executed.

Source [`symbol.lisp`], page 20.

`source-pathname` ((*definition* [`slot-definition`], *page 97*)) [Method]
Return slot DEFINITION's owner source pathname.

Source [`symbol.lisp`], page 20.

`source-pathname` ((*definition* [`combination-definition`], *page 79*)) [Method]
Return method combination DEFINITION's source pathname.

Source [`symbol.lisp`], page 20.

`source-pathname` ((*definition* [`expander-definition`], *page 83*)) [Method]
Return setf expander DEFINITION's source pathname.

Source [`symbol.lisp`], page 20.

`source-pathname` ((*definition* [`type-definition`], *page 102*)) [Method]
Return type DEFINITION's source pathname.

Source [`symbol.lisp`], page 20.

`source-pathname` ((*definition* [`symbol-macro-definition`], *page 100*)) [Method]
Return symbol macro DEFINITION's source pathname.

Source [`symbol.lisp`], page 20.

`source-pathname` ((*definition* [`special-definition`], *page 99*)) [Method]
Return special DEFINITION's source pathname.

Source [`symbol.lisp`], page 20.

`source-pathname` ((*definition* [`constant-definition`], *page 82*)) [Method]
Return constant DEFINITION's source pathname.

Source [`symbol.lisp`], page 20.

`source-pathname` (*definition*) [Method]
Return DEFINITION's object source pathname (this is the default method).

`stabilize` (*definition definitions packages pathnames*) [Generic Function]
Stabilize DEFINITION in DEFINITIONS and domestic PACKAGES and PATHNAMES.

Package [`net.didierverna.declt.assess`], page 32.

Source [`finalize.lisp`], page 27.

Method Combination
`progn`.

Options `:most-specific-first`

Methods

`stabilize` *progn* ((*definition* [`system-definition`], *page 100*) *definitions packages pathnames*) [Method]
Compute system DEFINITION's defsystem dependency definitions. Those definitions are guaranteed to be in the original system's order.

`stabilize` *progn* ((*definition* [`module-definition`], *page 90*) *definitions packages pathnames*) [Method]
Compute module DEFINITION's child definitions.
Those definitions are guaranteed to be in the module's original order.

`stabilize` *progn* ((*definition* [`lisp-file-definition`], *page 88*) *definitions packages pathnames*) [Method]
Compute Lisp file DEFINITION's definitions list.

`stabilize` *progn* ((*definition* [`component-definition`], *page 80*) *definitions packages pathnames*) [Method]
Compute component DEFINITION's parent and dependency definitions. Those definitions are guaranteed to be in the original component's order.

`stabilize` *progn* ((*definition* [`package-definition`], *page 92*) *definitions packages pathnames*) [Method]
Compute package DEFINITION's use, used-by, and definitions lists. New foreign package definitions may be created and added at the end of DEFINITIONS in the process.

`stabilize` *progn* ((*definition* [`function-alias-definition`], *page 86*) *definitions packages pathnames*) [Method]
Compute simple function alias DEFINITION's referee.

`stabilize` *progn* ((*definition* [`compiler-macro-alias-definition`], *page 79*) *definitions packages pathnames*) [Method]
Compute compiler macro alias DEFINITION's referee.

`stabilize` *progn* ((*definition* *[*`macro-alias-definition`*], *page 89*) *definitions packages pathnames*) [Method]
Compute macro alias DEFINITION's referee.

`stabilize` *progn* ((*definition* *[*`typed-structure-slot-definition`*], *page 103*) *definitions packages pathnames*) [Method]
Compute typed structure slot DEFINITION's reader and writer definitions.

`stabilize` *progn* ((*definition* *[*`clos-slot-definition`*], *page 78*) *definitions packages pathnames*) [Method]
Compute CLOS slot DEFINITION's reader and writer definitions.

`stabilize` *progn* ((*definition* *[*`clos-classoid-mixin`*], *page 77*) *definitions packages pathnames*) [Method]
Compute classoid DEFINITION's super/sub classoids, and method definitions.

`stabilize` *progn* ((*definition* *[*`method-definition`*], *page 90*) *definitions packages pathnames*) [Method]
Compute method DEFINITION's owner, and specializer references.

`stabilize` *progn* ((*definition* *[*`short-combination-definition`*], *page 96*) *definitions packages pathnames*) [Method]
Compute short combination DEFINITION's standalone combinator definition.

`stabilize` *progn* ((*definition* *[*`combination-definition`*], *page 79*) *definitions packages pathnames*) [Method]
Compute method combination DEFINITION's users.

`stabilize` *progn* ((*definition* *[*`generic-function-definition`*], *page 86*) *definitions packages pathnames*) [Method]
Compute generic function DEFINITION's methods, and combination definition.

`stabilize` *progn* ((*definition* *[*`short-expander-definition`*], *page 97*) *definitions packages pathnames*) [Method]
Compute short setf expander DEFINITION's standalone writer definition.

`stabilize` *progn* ((*definition* *[*`expander-definition`*], *page 83*) *definitions packages pathnames*) [Method]
Compute setf expander DEFINTIION's standalone reader definition.

`stabilize` *progn* ((*definition* *[*`setfable-funcoid-definition`*], *page 95*) *definitions packages pathnames*) [Method]
Compute DEFINITION's expander-for and expanders-to references.

`stabilize` *progn* ((*definition* *[*`symbol-definition`*], *page 99*) *definitions packages pathnames*) [Method]
Compute symbol DEFINITION's home package definition.
New foreign package definitions may be created and added at the end of DEFINITIONS in the process.

`stabilize` *progn* ((*definition* [`definition`], *page 82*) *definitions packages pathnames*) [Method]
Compute DEFINITION's source file definition.

6.2.5 Method combinations

`document` () [Method Combination]
The documentation protocol's method combination.
This method combination provides the following four method groups:
- around methods (optional, :around qualifier),
- opening methods (optional, :open qualifier),
- body methods (no qualifier),
- closing methods (optional, :close qualifier).

Around methods behave like those of the standard method combination, except that they are ordered most specific last. They can be used to conditionalize the actual rendering of documentation, for example in order to filter out definitions that are merged with others.

The main methods block behaves as follows.
- The most specific opening method, if any, is executed.
- All body methods (if any) are executed sequentially in most specific last order.
- Finally, the most specific closing method, if any, is executed.

No method group requires the existence of an applicable method, but for each generic call, there must of course be at least one applicable method, regardless of the group.

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

Source [`doc.lisp`], page 15.

Client Functions
[`document`], page 128.

6.2.6 Structures

`node` [Structure]
The NODE structure.
This structure holds Texinfo nodes.

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

Source [`texi.lisp`], page 13.

Direct superclasses
`structure-object`.

Direct slots

`name` [Slot]
Package [`net.didierverna.declt.assess`], page 32.
Readers [`node-name`], page 119.
Writers [`(setf node-name)`], page 119.

`synopsis` [Slot]
Readers [`node-synopsis`], page 120.
Writers [`(setf node-synopsis)`], page 120.

`section-type` [Slot]

Initform `:numbered`

Readers [`node-section-type`], page 120.

Writers [`(setf node-section-type)`], page 120.

`section-name` [Slot]

Readers [`node-section-name`], page 119.

Writers [`(setf node-section-name)`], page 119.

`next` [Slot]

Readers [`node-next`], page 119.

Writers [`(setf node-next)`], page 119.

`previous` [Slot]

Readers [`node-previous`], page 119.

Writers [`(setf node-previous)`], page 119.

`up` [Slot]

Readers [`node-up`], page 120.

Writers [`(setf node-up)`], page 120.

`children` [Slot]

Package [`net.didierverna.declt.assess`], page 32.

Readers [`node-children`], page 119.

Writers [`(setf node-children)`], page 119.

`before-menu-contents` [Slot]

Readers [`node-before-menu-contents`], page 119.

Writers [`(setf node-before-menu-contents)`], page 119.

`after-menu-contents` [Slot]

Readers [`node-after-menu-contents`], page 118.

Writers [`(setf node-after-menu-contents)`], page 118.

6.2.7 Classes

`context` [Class]

The class of rendering contexts.

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

Source [`doc.lisp`], page 15.

Direct methods

- [`declt-notice`], page 128.
- [`default-values`], page 128.
- [`foreign-definitions`], page 134.
- [`locations`], page 137.

Direct slots

`locations` [Slot]
Whether to hyperlink definitions to their locations.
Currently supported values are NIL (the default), and :file-system.

Initargs `:locations`

Readers [`locations`], page 137.

Writers *This slot is read-only.*

`default-values` [Slot]
Whether to render default / standard values.

Initargs `:default-values`

Readers [`default-values`], page 128.

Writers *This slot is read-only.*

`foreign-definitions` [Slot]
Whether to render foreign definitions.

Initargs `:foreign-definitions`

Readers [`foreign-definitions`], page 134.

Writers *This slot is read-only.*

`declt-notice` [Slot]
whether to add a small credit paragraph about Declt. Possible values are NIL, :short, or :long (the default).

Initform `:long`

Initargs `:declt-notice`

Readers [`declt-notice`], page 128.

Writers *This slot is read-only.*

Appendix A Indexes

A.1 Concepts

D

L

A.2 Functions

%

(

@

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

A.3 Variables

*

A

B

C

D

E

F

H

I

L

M

N

O

P

R

S

T

U

W

A.4 Data types

A

C

D

E

F

G

H

J

L

M

N

O

P

R

S

T

U

V

W