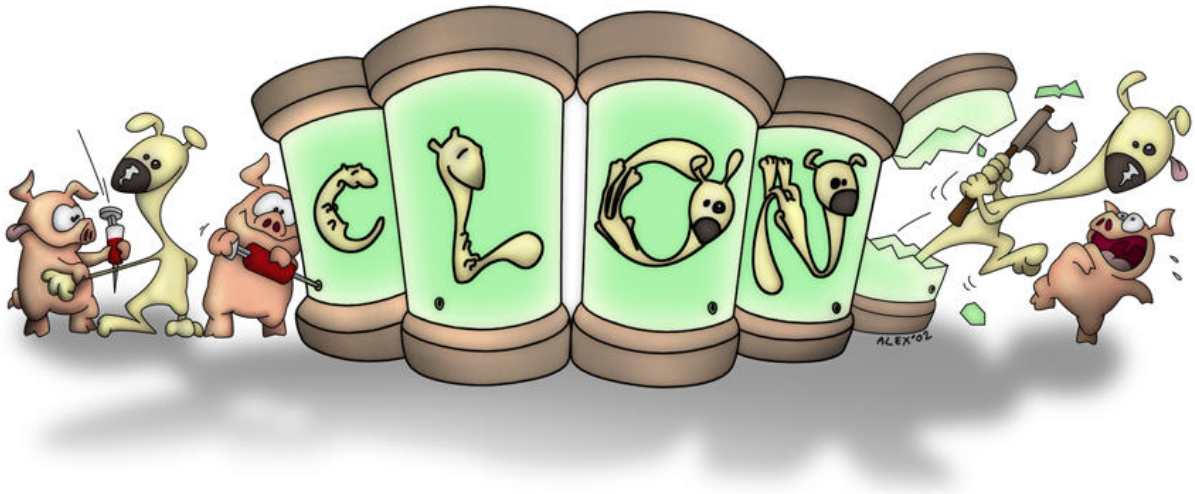
The Clon Reference Manual

The Command-Line Options Nuker, version 1.0 beta 27 "Michael Brecker"



Didier Verna <didier@didierverna.net>

This manual was generated automatically by Declt 4.0 beta 3 "William Riker" on Sun Jun 15 22:17:31 2025 GMT+1.

Table of Contents

Copying

1 Introduction

`Clon` is a library for managing command-line options in standalone Common-Lisp applications. It provides a unified option syntax with both short and long names, automatic completion of partial names and automatic retrieval/conversion of option arguments from the command-line, associated environment variables, fallback or default values. `Clon` comes with a set of extensible option types (switches, paths, strings *etc.*). `Clon` also provides automatic generation and formatting of help strings, with support for highlighting on `tty`'s through ISO/IEC 6429 SGR. This formatting is customizable through *themes*.

Depending on the target audience, `Clon` stands for either "The Command-Line Options Nuker" or "The Common-Lisp Options Nuker". `Clon` also has a recursive acronym: "`Clon` Likes Options Nuking", and a reverse one: "Never Omit to Link with `Clon`". Other possible expansions of the acronym are still being investigated.

This is the `Clon` reference manual, and as such, it is not meant to be read. It may help you find sleep in case of insomnia though. `Clon` comes with two human-readable manuals:

- the "end-user manual" (see *The Clon End-User Manual*) is for the `Clon` *end-user*, that is, the user of an application powered by `Clon`. It describes how to use the command-line of clonified[1] applications and how to customize `Clon`'s output. Everybody should read this manual first.
- the "user manual" (see *The Clon End-User Manual*) is for the `Clon` *user*, that is, the developer of a Common-Lisp application who wants to use `Clon` for command-line option management. It describes how to clonify your application and extend the library with your own option types.

[1] An application using `Clon` for its command-line option management is said to be *clonified*. It is also possible to say *clonfiscated*. However, we advise against using *clonistified*. The term *clonificated* is also considered bad style, and the use of *clonificationated* is strictly prohibited.

2 Systems

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

2.1 net.didierverna.clon

Command-line options management for standalone Common Lisp applications

Long Name
The Command-Line Options Nuker

Author Didier Verna

Contact didier@didierverna.net

Home Page
http://www.lrde.epita.fr/~didier/software/lisp/clon.php

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

License BSD

Long Description
Clon is a library for command-line options management. It is intended to ease the creation of standalone Common Lisp applications by providing a powerful and uniform command-line options interface. The most important features of Clon are the following.

- From the application programmer's point of view: centralized command-line options specification and management, including automatic generation of help strings, conversion from command-line / defaults / fallbacks / environment variables to application-level option values, global or on-demand option retrieval, and extensibility (the programmer can define his own option types).

- From the application user's point of view: uniform command-line option syntax across all Clon applications, customization of the help strings layout (with optional ISO6429 coloring on terminals that support it), automatic completion of abbreviated option names and short/long/pack syntax.

Defsystem Dependency
[net.didierverna.clon.setup/termio], page 5 (system).

Dependencies
- [net.didierverna.clon.core], page 6 (system).
- [net.didierverna.clon.termio], page 7 (system)., for feature :net.didierverna.clon.termio

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

2.2 net.didierverna.clon.setup/termio

Clon's support for automatic configuration of termio support

Long Name
The Command-Line Options Nuker, termio setup

Author Didier Verna <didier@didierverna.net>

Contact `didier@didierverna.net`

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

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

License BSD

Long Description
This is a virtual subsystem or Clon (no actual code). Its purpose is only to autodetect termio support and update Clon's preload configuration on load. For a more complete description of Clon, see the net.didierverna.clon system.

Dependency
[`net.didierverna.clon.setup`], page 6 (system).

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

2.3 `net.didierverna.clon.setup`

Clon's preload setup library

Long Name
The Command-Line Options Nuker, setup library

Author Didier Verna

Contact `didier@didierverna.net`

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

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

License BSD

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

Dependency
`named-readtables` (system).

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

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

2.4 `net.didierverna.clon.core`

Clon's basic, platform-independent functionality

Long Name
The Command-Line Options Nuker, core library

Author Didier Verna

Contact `didier@didierverna.net`

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

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

License BSD

Long Description
Clon's core library provides the platform/feature independent part. For a more complete description of Clon, see the net.didierverna.clon system.

Dependencies
- `sb-posix` (system)., required, for feature `:sbcl`
- [`net.didierverna.clon.setup`], page 6 (system).

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

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

2.5 net.didierverna.clon.termio

Clon's support for termio (tty geometry and fontification)

Long Name
The Command-Line Options Nuker, termio library

Author Didier Verna

Contact `didier@didierverna.net`

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

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

License BSD

Long Description
Clon's termio library provides automatic detection of tty geometry and ISO6429 coloring on terminals that support it. For a more complete description of Clon, see the net.didierverna.clon system.

If Feature `:net.didierverna.clon.termio`

Defsystem Dependencies
- [`net.didierverna.clon.setup/termio`], page 5 (system).
- `sb-grovel` (system)., required, for feature `:sbcl`
- `cffi-grovel` (system)., for feature `(:or :allegro :clisp :lispworks)`

Dependencies
- `sb-posix` (system)., for feature `:sbcl`
- `cffi` (system)., for feature `(:and :clisp :net.didierverna.clon.termio)`
- [`net.didierverna.clon.setup`], page 6 (system).
- [`net.didierverna.clon.core`], page 6 (system).

Source [`net.didierverna.clon.termio.asd`], page 11.

Child Components

- [sbcl/constants.lisp], page 27 (file).
- [termio.lisp], page 27 (file).

3 Modules

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

3.1 net.didierverna.clon.setup/src

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

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

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

Child Components
- [configuration.lisp], page 12 (file).
- [readtable.lisp], page 12 (file).
- [version.lisp], page 12 (file).
- [termio.lisp], page 13 (file).

3.2 net.didierverna.clon.core/src

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

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

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

Child Components
- [util.lisp], page 13 (file).
- [item.lisp], page 14 (file).
- [text.lisp], page 14 (file).
- [options], page 9 (module).
- [container.lisp], page 20 (file).
- [group.lisp], page 20 (file).
- [retrieval], page 10 (module).
- [synopsis.lisp], page 21 (file).
- [output], page 10 (module).
- [context.lisp], page 25 (file).

3.3 net.didierverna.clon.core/src/options

Dependency
[text.lisp], page 14 (file).

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

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

Child Components
- [option.lisp], page 15 (file).

- [flag.lisp], page 15 (file).
- [valued.lisp], page 15 (file).
- [negatable.lisp], page 16 (file).
- [switch-base.lisp], page 17 (file).
- [switch.lisp], page 17 (file).
- [stropt.lisp], page 17 (file).
- [lispobj.lisp], page 18 (file).
- [path.lisp], page 18 (file).
- [enum-base.lisp], page 19 (file).
- [enum.lisp], page 19 (file).
- [xswitch.lisp], page 19 (file).

3.4 net.didierverna.clon.core/src/retrieval

Dependency
[options], page 9 (module).

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

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

Child Components
- [cmdline.lisp], page 20 (file).
- [environ.lisp], page 21 (file).

3.5 net.didierverna.clon.core/src/output

Dependencies
- [synopsis.lisp], page 21 (file).
- [retrieval], page 10 (module).

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

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

Child Components
- [face.lisp], page 22 (file).
- [sheet.lisp], page 23 (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.clon/net.didierverna.clon.asd

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

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

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

4.1.2 net.didierverna.clon.setup/termio/net.didierverna.clon.setup.asd

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

Parent Component
[net.didierverna.clon.setup/termio], page 5 (system).

ASDF Systems
- [net.didierverna.clon.setup/termio], page 5.
- [net.didierverna.clon.setup], page 6.

4.1.3 net.didierverna.clon.core/net.didierverna.clon.core.asd

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

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

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

4.1.4 net.didierverna.clon.termio/net.didierverna.clon.termio.asd

Source [net.didierverna.clon.termio.asd], page 11.

Parent Component
[net.didierverna.clon.termio], page 7 (system).

ASDF Systems
[net.didierverna.clon.termio], page 7.

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

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

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

Packages [net.didierverna.clon.setup], page 36.

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

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

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

Public Interface
- [configuration], page 41 (function).
- [configure], page 42 (function).

Internals [*configuration*], page 49 (special variable).

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

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

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

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

Internals
- [clindent], page 52 (function).
- [defindent], page 50 (macro).
- [i-reader], page 58 (function).
- [~-reader], page 68 (function).

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

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

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

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

Public Interface
- [*copyright-years*], page 39 (special variable).
- [*release-major-level*], page 39 (special variable).
- [*release-minor-level*], page 39 (special variable).
- [*release-name*], page 39 (special variable).
- [*release-status*], page 39 (special variable).
- [*release-status-level*], page 40 (special variable).
- [version], page 47 (function).

Internals
- [%version], page 52 (function).
- [release-status-number], page 65 (function).

4.1.9 net.didierverna.clon.setup/src/termio.lisp

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

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

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

Public Interface
[setup-termio], page 46 (function).

Internals [restrict-because], page 66 (function).

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

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

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

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

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

4.1.11 net.didierverna.clon.core/src/util.lisp

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

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

Public Interface

- [*executablep*], page 39 (special variable).
- [cmdline], page 41 (function).
- [dump], page 40 (macro).
- [executablep], page 42 (function).
- [exit], page 42 (function).
- [make-instance], page 49 (method).
- [validate-superclass], page 49 (method).
- [validate-superclass], page 49 (method).

Internals

- [abstract-class], page 103 (class).
- [accumulate], page 50 (macro).
- [beginning-of-string-p], page 52 (function).
- [closest-match], page 53 (function).
- [complete-string], page 54 (function).
- [copy-instance], page 75 (generic function).
- [declare-valid-superclass], page 50 (macro).
- [defabstract], page 50 (macro).
- [econd], page 50 (macro).
- [endpush], page 50 (macro).

- [error-string], page 77 (reader method).
- [(setf error-string)], page 77 (writer method).
- [getenv], page 56 (function).
- [home-directory], page 57 (function).
- [home-directory], page 96 (condition).
- [list-to-string], page 58 (function).
- [macosp], page 58 (function).
- [maybe-push], page 51 (macro).
- [putenv], page 64 (function).
- [remove-keys], page 65 (function).
- [replace-in-keys], page 51 (macro).
- [replace-key], page 65 (function).
- [replace-keys], page 65 (function).
- [select-keys], page 68 (function).

4.1.12 net.didierverna.clon.core/src/item.lisp

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

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

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

Internals

- [help-spec], page 78 (generic function).
- [hiddenp], page 79 (reader method).
- [item], page 111 (class).
- [traversedp], page 92 (reader method).
- [(setf traversedp)], page 92 (writer method).
- [untraverse], page 93 (generic function).

4.1.13 net.didierverna.clon.core/src/text.lisp

Dependency
[item.lisp], page 14 (file).

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

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

Public Interface
[make-text], page 46 (function).

Internals

- [contents], page 74 (reader method).
- [help-spec], page 79 (method).
- [make-internal-text], page 60 (function).
- [text], page 119 (class).
- [untraverse], page 93 (method).

4.1.14 net.didierverna.clon.core/src/options/option.lisp

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

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

Public Interface
- [initialize-instance], page 48 (method).
- [initialize-instance], page 48 (method).

Internals
- [check-name-clash], page 71 (generic function).
- [description], page 75 (reader method).
- [env-var], page 76 (reader method).
- [help-spec], page 78 (method).
- [long-name], page 82 (reader method).
- [match-option], page 61 (function).
- [negated-pack-char], page 83 (generic function).
- [option], page 84 (reader method).
- [option], page 112 (class).
- [option-abbreviation-distance], page 62 (function).
- [option-error], page 99 (condition).
- [option-sticky-distance], page 85 (generic function).
- [potential-pack-char], page 62 (function).
- [short-name], page 89 (reader method).
- [short-pack-char], page 89 (generic function).
- [untraverse], page 93 (method).

4.1.15 net.didierverna.clon.core/src/options/flag.lisp

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

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

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

Public Interface
[make-flag], page 43 (function).

Internals
- [flag], page 110 (class).
- [make-internal-flag], page 59 (function).

4.1.16 net.didierverna.clon.core/src/options/valued.lisp

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

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

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

Public Interface

- [initialize-instance], page 48 (method).
- [initialize-instance], page 48 (method).

Internals

- [*item-names*], page 49 (special variable).
- [argument], page 69 (reader method).
- [argument-name], page 69 (reader method).
- [argument-required-p], page 69 (reader method).
- [check], page 71 (generic function).
- [comment], page 73 (reader method).
- [comment], page 73 (reader method).
- [convert], page 74 (generic function).
- [default-value], page 75 (reader method).
- [defoption], page 50 (macro).
- [fallback-value], page 77 (reader method).
- [help-spec], page 78 (method).
- [invalid-argument], page 97 (condition).
- [invalid-value], page 98 (condition).
- [option-sticky-distance], page 85 (method).
- [read-argument], page 64 (function).
- [read-value], page 64 (function).
- [restartable-check], page 65 (function).
- [restartable-convert], page 66 (function).
- [short-pack-char], page 90 (method).
- [short-syntax-help-spec-prefix], page 90 (generic function).
- [stringify], page 90 (generic function).
- [value], page 94 (reader method).
- [valued-option], page 119 (class).

4.1.17 net.didierverna.clon.core/src/options/negatable.lisp

Dependency

[valued.lisp], page 15 (file).

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

Parent Component

[options], page 9 (module).

Internals

- [negatable], page 112 (class).
- [negated-pack-char], page 84 (method).
- [short-syntax-help-spec-prefix], page 90 (method).

4.1.18 net.didierverna.clon.core/src/options/switch-base.lisp

Dependency
[negatable.lisp], page 16 (file).

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

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

Public Interface
- [initialize-instance], page 48 (method).
- [initialize-instance], page 48 (method).

Internals
- [argument-style], page 70 (reader method).
- [argument-styles], page 70 (reader method).
- [(setf argument-styles)], page 70 (writer method).
- [no-values], page 84 (reader method).
- [(setf no-values)], page 84 (writer method).
- [switch-base], page 116 (class).
- [yes-values], page 94 (reader method).
- [(setf yes-values)], page 94 (writer method).

4.1.19 net.didierverna.clon.core/src/options/switch.lisp

Dependency
[switch-base.lisp], page 17 (file).

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

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

Public Interface
- [initialize-instance], page 48 (method).
- [make-switch], page 45 (function).

Internals
- [check], page 71 (method).
- [convert], page 74 (method).
- [make-internal-switch], page 60 (function).
- [stringify], page 91 (method).
- [switch], page 116 (class).

4.1.20 net.didierverna.clon.core/src/options/stropt.lisp

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

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

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

Public Interface
[make-stropt], page 45 (function).

Internals

- [check], page 71 (method).
- [convert], page 74 (method).
- [make-internal-stropt], page 60 (function).
- [stringify], page 91 (method).
- [stropt], page 116 (class).

4.1.21 net.didierverna.clon.core/src/options/lispobj.lisp

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

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

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

Public Interface
[make-lispobj], page 44 (function).

Internals

- [check], page 71 (method).
- [convert], page 74 (method).
- [lispobj], page 111 (class).
- [make-internal-lispobj], page 59 (function).
- [stringify], page 91 (method).
- [typespec], page 93 (reader method).

4.1.22 net.didierverna.clon.core/src/options/path.lisp

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

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

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

Public Interface
[make-path], page 44 (function).

Internals

- [check], page 71 (method).
- [convert], page 74 (method).
- [directory-pathname-p], page 55 (function).
- [make-internal-path], page 59 (function).
- [path], page 113 (class).
- [path-type], page 85 (reader method).
- [pathname-component-null-p], page 62 (function).
- [split-path], page 68 (function).
- [stringify], page 91 (method).

4.1.23 net.didierverna.clon.core/src/options/enum-base.lisp

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

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

Public Interface
[initialize-instance], page 47 (method).

Internals

- [enum], page 76 (reader method).
- [enum-base], page 106 (class).

4.1.24 net.didierverna.clon.core/src/options/enum.lisp

Dependencies

- [valued.lisp], page 15 (file).
- [enum-base.lisp], page 19 (file).

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

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

Public Interface
[make-enum], page 43 (function).

Internals

- [check], page 71 (method).
- [convert], page 74 (method).
- [enum], page 105 (class).
- [make-internal-enum], page 58 (function).
- [stringify], page 91 (method).

4.1.25 net.didierverna.clon.core/src/options/xswitch.lisp

Dependencies

- [valued.lisp], page 15 (file).
- [switch-base.lisp], page 17 (file).
- [enum-base.lisp], page 19 (file).

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

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

Public Interface
[make-xswitch], page 46 (function).

Internals

- [check], page 71 (method).
- [convert], page 74 (method).
- [make-internal-xswitch], page 61 (function).
- [stringify], page 91 (method).
- [xswitch], page 121 (class).

4.1.26 net.didierverna.clon.core/src/container.lisp

Dependency
[options], page 9 (module).

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

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

Public Interface
- [initialize-instance], page 47 (method).
- [initialize-instance], page 47 (method).

Internals
- [check-name-clash], page 72 (method).
- [check-name-clash], page 72 (method).
- [check-name-clash], page 72 (method).
- [container], page 103 (class).
- [help-spec], page 78 (method).
- [items], page 81 (reader method).
- [untraverse], page 93 (method).

4.1.27 net.didierverna.clon.core/src/group.lisp

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

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

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

Public Interface
- [defgroup], page 40 (macro).
- [make-group], page 43 (function).

Internals
- [%defgroup], page 50 (macro).
- [group], page 110 (class).
- [header], page 78 (reader method).
- [help-spec], page 78 (method).

4.1.28 net.didierverna.clon.core/src/retrieval/cmdline.lisp

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

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

Internals
- [argument], page 69 (reader method).
- [argument-popable-p], page 52 (function).
- [cmdline-convert], page 53 (function).
- [cmdline-error], page 94 (condition).

- [cmdline-option-error], page 95 (condition).
- [invalid-cmdline-argument], page 97 (condition).
- [invalid-negated-syntax], page 98 (condition).
- [item], page 80 (reader method).
- [maybe-pop-argument], page 51 (macro).
- [missing-cmdline-argument], page 99 (condition).
- [name], page 83 (reader method).
- [option-call-p], page 62 (function).
- [restartable-cmdline-convert], page 65 (function).
- [restartable-invalid-negated-syntax-error], page 51 (macro).
- [restartable-spurious-cmdline-argument-error], page 51 (macro).
- [retrieve-from-long-call], page 87 (generic function).
- [retrieve-from-negated-call], page 87 (generic function).
- [retrieve-from-short-call], page 88 (generic function).
- [spurious-cmdline-argument], page 99 (condition).

4.1.29 net.didierverna.clon.core/src/retrieval/environ.lisp

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

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

Internals

- [env-val], page 76 (reader method).
- [env-var], page 76 (reader method).
- [environment-convert], page 55 (function).
- [environment-error], page 96 (condition).
- [environmental-option-error], page 96 (condition).
- [invalid-environment-value], page 97 (condition).
- [read-env-val], page 64 (function).
- [restartable-environment-convert], page 66 (function).
- [retrieve-from-environment], page 87 (generic function).

4.1.30 net.didierverna.clon.core/src/synopsis.lisp

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

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

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

Public Interface

- [*synopsis*], page 40 (special variable).
- [defsynopsis], page 40 (macro).
- [initialize-instance], page 49 (method).
- [initialize-instance], page 49 (method).
- [make-synopsis], page 45 (function).

Internals

- [clon-options-group], page 72 (reader method).
- [do-options], page 50 (macro).
- [help-spec], page 78 (method).
- [mapoptions], page 82 (generic function).
- [negated-pack], page 83 (reader method).
- [postfix], page 86 (reader method).
- [potential-pack], page 86 (reader method).
- [potential-pack-p], page 86 (generic function).
- [short-pack], page 89 (reader method).
- [synopsis], page 118 (class).

4.1.31 net.didierverna.clon.core/src/output/face.lisp

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

Parent Component

[output], page 10 (module).

Public Interface

- [initialize-instance], page 48 (method).
- [initialize-instance], page 48 (method).
- [initialize-instance], page 48 (method).
- [slot-unbound], page 49 (method).

Internals

- [*highlight-properties*], page 49 (special variable).
- [add-subface], page 52 (function).
- [attach-face-tree], page 52 (function).
- [background], page 70 (reader method).
- [blink], page 70 (reader method).
- [bottom-padding], page 71 (reader method).
- [concealedp], page 74 (reader method).
- [crossed-out-p], page 75 (reader method).
- [face], page 106 (class).
- [face-highlight-property-set-p], page 55 (function).
- [face-highlight-property-value], page 55 (function).
- [foreground], page 77 (reader method).
- [framedp], page 77 (reader method).
- [intensity], page 80 (reader method).
- [inversep], page 80 (reader method).
- [italicp], page 80 (reader method).
- [item-separator], page 80 (reader method).
- [left-padding], page 81 (reader method).
- [make-face-tree], page 82 (generic function).
- [make-raw-face-tree], page 61 (function).
- [name], page 83 (reader method).

- [parent], page 85 (reader method).
- [parent-generation], page 62 (function).
- [right-padding], page 88 (reader method).
- [search-branch], page 67 (function).
- [search-face], page 67 (function).
- [subface], page 91 (generic function).
- [subfaces], page 91 (reader method).
- [top-padding], page 92 (reader method).
- [underline], page 93 (reader method).
- [visiblep], page 94 (reader method).

4.1.32 net.didierverna.clon.core/src/output/sheet.lisp

Dependency
[face.lisp], page 22 (file).

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

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

Public Interface
- [initialize-instance], page 47 (method).
- [initialize-instance], page 47 (method).

Internals
- [available-right-margin], page 52 (function).
- [close-frame], page 72 (generic function).
- [close-line], page 53 (function).
- [close-sface], page 53 (function).
- [column], page 73 (reader method).
- [(setf column)], page 73 (writer method).
- [copy-frame], page 54 (function).
- [copy-highlight-frame], page 54 (function).
- [copy-highlight-property-instance], page 54 (function).
- [current-frame], page 54 (function).
- [current-left-margin], page 54 (function).
- [current-right-margin], page 54 (function).
- [current-sface], page 54 (function).
- [find-sface], page 55 (function).
- [flush-sheet], page 55 (function).
- [frame], page 101 (structure).
- [frame-left-margin], page 55 (reader).
- [(setf frame-left-margin)], page 55 (writer).
- [frame-p], page 56 (function).
- [frame-right-margin], page 56 (reader).
- [(setf frame-right-margin)], page 56 (writer).
- [frame-sface], page 56 (reader).

- [princ-string], page 63 (function).
- [print-faced-help-spec], page 63 (function).
- [print-help], page 63 (function).
- [print-help-spec], page 86 (generic function).
- [print-string], page 63 (function).
- [push-frame], page 64 (function).
- [reach-column], page 64 (function).
- [read-sface-tree], page 64 (function).
- [safe-left-margin], page 66 (function).
- [safe-right-margin], page 67 (function).
- [sface], page 114 (class).
- [sface-tree], page 89 (reader method).
- [sheet], page 114 (class).
- [sibling], page 90 (reader method).
- [top-padding], page 92 (method).
- [top-padding], page 92 (method).
- [try-read-sface-tree], page 68 (function).
- [try-read-theme], page 68 (function).

4.1.33 net.didierverna.clon.core/src/context.lisp

Dependency

[output], page 10 (module).

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

Parent Component

[src], page 9 (module).

Public Interface

- [*context*], page 39 (special variable).
- [cmdline-options-p], page 41 (function).
- [cmdline-p], page 41 (function).
- [do-cmdline-options], page 40 (macro).
- [getopt], page 42 (function).
- [getopt-cmdline], page 42 (function).
- [help], page 42 (function).
- [initialize-instance], page 47 (method).
- [make-context], page 42 (function).
- [multiple-value-getopt-cmdline], page 41 (macro).
- [progname], page 46 (function).
- [remainder], page 46 (function).
- [with-context], page 41 (macro).

Internals

- [argument], page 69 (reader method).
- [clon-options-group], page 72 (method).
- [cmdline-junk-error], page 95 (condition).

- [unrecognized-negated-call-error], page 100 (condition).
- [unrecognized-short-call-error], page 101 (condition).
- [untraverse], page 93 (method).
- [with-context-error-handler], page 52 (macro).

4.1.34 net.didierverna.clon.termio/sbcl/constants.lisp

Source [net.didierverna.clon.termio.asd], page 11.

Parent Component
[net.didierverna.clon.termio], page 7 (system).

4.1.35 net.didierverna.clon.termio/termio.lisp

Dependency
[sbcl/constants.lisp], page 27 (file).

Source [net.didierverna.clon.termio.asd], page 11.

Parent Component
[net.didierverna.clon.termio], page 7 (system).

Internals

- [stream-ioctl-output-handle], page 90 (generic function).
- [stream-line-width], page 68 (function).

5 Packages

Packages are listed by definition order.

5.1 net.didierverna.clon

The Clon library's package.

Source [package.lisp], page 13.

Nickname clon

Use List

- common-lisp.
- [net.didierverna.clon.setup], page 36.

Public Interface

- [*context*], page 39 (special variable).
- [*executablep*], page 39 (special variable).
- [*synopsis*], page 40 (special variable).
- [cmdline], page 41 (function).
- [cmdline-options-p], page 41 (function).
- [cmdline-p], page 41 (function).
- [defgroup], page 40 (macro).
- [defsynopsis], page 40 (macro).
- [do-cmdline-options], page 40 (macro).
- [dump], page 40 (macro).
- [executablep], page 42 (function).
- [exit], page 42 (function).
- [getopt], page 42 (function).
- [getopt-cmdline], page 42 (function).
- [help], page 42 (function).
- [make-context], page 42 (function).
- [make-enum], page 43 (function).
- [make-flag], page 43 (function).
- [make-group], page 43 (function).
- [make-lispobj], page 44 (function).
- [make-path], page 44 (function).
- [make-stropt], page 45 (function).
- [make-switch], page 45 (function).
- [make-synopsis], page 45 (function).
- [make-text], page 46 (function).
- [make-xswitch], page 46 (function).
- [multiple-value-getopt-cmdline], page 41 (macro).
- [nickname-package], page 46 (function).
- [progname], page 46 (function).
- [remainder], page 46 (function).

- [short-pack-char], page 89 (generic function).
- [short-syntax-help-spec-prefix], page 90 (generic function).
- [sibling], page 90 (generic reader).
- [split-path], page 68 (function).
- [spurious-cmdline-argument], page 99 (condition).
- [stream-ioctl-output-handle], page 90 (generic function).
- [stream-line-width], page 68 (function).
- [stringify], page 90 (generic function).
- [stropt], page 116 (class).
- [subface], page 91 (generic function).
- [subfaces], page 91 (generic reader).
- [switch], page 116 (class).
- [switch-base], page 116 (class).
- [synopsis], page 91 (generic reader).
- [synopsis], page 118 (class).
- [text], page 119 (class).
- [theme], page 92 (generic reader).
- [top-padding], page 92 (generic function).
- [traversedp], page 92 (generic reader).
- [(setf traversedp)], page 92 (generic writer).
- [try-read-sface-tree], page 68 (function).
- [try-read-theme], page 68 (function).
- [typespec], page 93 (generic reader).
- [underline], page 93 (generic reader).
- [unknown-cmdline-option-error], page 100 (condition).
- [unrecognized-negated-call-error], page 100 (condition).
- [unrecognized-short-call-error], page 101 (condition).
- [untraverse], page 93 (generic function).
- [value], page 93 (generic reader).
- [valued-option], page 119 (class).
- [visiblep], page 94 (generic reader).
- [with-context-error-handler], page 52 (macro).
- [xswitch], page 121 (class).
- [yes-values], page 94 (generic reader).
- [(setf yes-values)], page 94 (generic writer).

5.2 net.didierverna.clon.setup

The Clon setup library's package.

Source [package.lisp], page 11.

Use List common-lisp.

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

Public Interface

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

`*context*` [Special Variable]
The current context.

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

Source [context.lisp], page 25.

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

Package [net.didierverna.clon.setup], page 36.

Source [version.lisp], page 12.

`*executablep*` [Special Variable]
Whether the current Lisp image is a standalone executable.
This information is needed in several implementations to distinguish user options from implementation-specific ones on the command-line.

It is set automatically to T by the 'dump' function, which see.
If the image is dumped by ASDF's program-op, this variable is ignored. In any other case, that is, when dumping via an implementation-specific function, it must be set manually to T just before dumping.

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

Source [util.lisp], page 13.

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

Package [net.didierverna.clon.setup], page 36.

Source [version.lisp], page 12.

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

Package [net.didierverna.clon.setup], page 36.

Source [version.lisp], page 12.

`*release-name*` [Special Variable]
The name of this release.
The general naming theme for Clon is "Great Jazz musicians".

Package [net.didierverna.clon.setup], page 36.

Source [version.lisp], page 12.

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

Package [net.didierverna.clon.setup], page 36.

Source [version.lisp], page 12.

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

Package [`net.didierverna.clon.setup`], page 36.

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

`*synopsis*` [Special Variable]
The current synopsis.

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

Source [`synopsis.lisp`], page 21.

6.1.2 Macros

`defgroup` ((&rest *keys* &key *header hidden*) &body *forms*) [Macro]
Define a new group.
KEYS are initargs to MAKE-GROUP (currently, only :header).
Each form in FORMS will be treated as a new :item.
The CAR of each form is the name of the operation to perform: TEXT, GROUP, or an option class name. The rest are the arguments to the MAKE-<OP> function or the DEFGROUP macro.

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

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

`defsynopsis` ((&rest *keys* &key *postfix make-default*) &body *forms*) [Macro]
Define a new synopsis.

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

Source [`synopsis.lisp`], page 21.

`do-cmdline-options` ((*option name value source* &key *context*) &body *body*) [Macro]
Evaluate BODY over all command-line options in CONTEXT.
OPTION, NAME and VALUE are bound to each option's object, name used on the command-line and retrieved value.

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

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

`dump` (*name function* &rest *args*) [Macro]
Dump a standalone executable named NAME starting with FUNCTION.
ARGS may be any arguments understood by the underlying implementation's dumping facility. They will simply be passed along. Note that DUMP already passes some such arguments. Some of them are critical for the dumping facility (e.g. :executable) and cannot be overridden. Some others, however, will be if you provide them as well (e.g. :load-init-file).

Since executable dumping is not available in all supported implementations, this function behaves differently in some cases, as described below.

- ECL doesn't create executables by dumping a Lisp image, but relies on having toplevel code to execute instead, so this macro simply expands to a call to FUNCTION. This also means that ARGS is unused.
- ABCL can't dump executables at all because of the underlying Java implementation, so

this macro expands to just (PROGN) but creates a Java class file with a main function that creates an interpreter, loads
the file in which this macro call appears and calls FUNCTION. This also means that ARGS is unused.

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

Source [util.lisp], page 13.

`multiple-value-getopt-cmdline` ((*option name value source* &key *context*) &body *body*) [Macro]
Get the next command-line option in CONTEXT. and evaluate BODY. OPTION, NAME and VALUE are bound to the values returned by GETOPT-CMDLINE. BODY is executed only if there is a next command-line option.

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

Source [context.lisp], page 25.

`with-context` (*context* &body *body*) [Macro]
Execute BODY with *context* bound to CONTEXT.

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

Source [context.lisp], page 25.

6.1.3 Ordinary functions

`cmdline` () [Function]
Get the current application's command-line.
This command-line is not supposed to contain any Lisp implementation specific option; only user-level ones. When a standalone executable is dumped, this is always the case. When used interactively, this depends on the underlying Lisp implementation. See appendix A.5 of the user manual for more information.

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

Source [util.lisp], page 13.

`cmdline-options-p` (&key *context*) [Function]
Return T if CONTEXT has any unprocessed options left.

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

Source [context.lisp], page 25.

`cmdline-p` (&key *context*) [Function]
Return T if CONTEXT has anything on its command-line.

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

Source [context.lisp], page 25.

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

Package [net.didierverna.clon.setup], page 36.

Source [configuration.lisp], page 12.

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

Package [`net.didierverna.clon.setup`], page 36.

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

`executablep` () [Function]
Return T if the current Lisp image is a standalone executable.
This function detects executables dumped by ASDF's program-op operation, those dumped by Clon's 'dump' function (which see), and those in which '*executablep*' (which see) has been set to T manually.

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

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

`exit` (&optional *status*) [Function]
Quit the current application with STATUS.
This function is considered deprecated. Please use UIOP:QUIT instead.

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

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

`getopt` (&rest *keys* &key *context short-name long-name option*) [Function]
Get an option's value in CONTEXT.
The option can be specified either by SHORT-NAME, LONG-NAME, or directly via an OPTION object.
Return two values:
- the retrieved value,
- the value's source.

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

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

`getopt-cmdline` (&key *context*) [Function]
Get the next command-line option in CONTEXT.
When there is no next command-line option, return nil. Otherwise, return four values:
- the option object,
- the option's name used on the command-line,
- the retrieved value,
- the value source.

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

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

`help` (&key *context item output-stream search-path theme line-width highlight*) [Function]
Print CONTEXT's help.

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

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

`make-context` (&rest *keys* &key *synopsis cmdline progname make-current*) [Function]
Make a new context.
- SYNOPSIS is the program synopsis to use in that context.
It defaults to *SYNOPSIS*.

- CMDLINE is the argument list (strings) to process.
It defaults to a POSIX conformant argv.
- PROGNAME is an alternate value for argv[0].
It defaults to NIL, in which case the actual argv[0] is used. Otherwise, it can be a non-empty string, standing for itself,
or :environment meaning to retrieve the value of the __CL_ARGV0 environment variable (ignored if it's empty).
value.
- If MAKE-CURRENT, make the new context current. This is the default.

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

Source [context.lisp], page 25.

`make-enum` (&rest *keys* &key *short-name long-name description argument-name argument-type enum env-var fallback-value default-value hidden*) [Function]

Make a new enum option.
- SHORT-NAME is the option's short name (without the dash).
It defaults to nil.
- LONG-NAME is the option's long name (without the double-dash).
It defaults to nil.
- DESCRIPTION is the option's description appearing in help strings.
It defaults to nil.
- ARGUMENT-NAME is the option's argument name appearing in help strings. - ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENUM is the set of possible values.
- ENV-VAR is the option's associated environment variable.
It defaults to nil.
- FALLBACK-VALUE is the option's fallback value (for missing optional arguments), if any.
- DEFAULT-VALUE is the option's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

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

Source [enum.lisp], page 19.

`make-flag` (&rest *keys* &key *short-name long-name description env-var hidden*) [Function]

Make a new flag.
- SHORT-NAME is the option's short name (without the dash).
It defaults to nil.
- LONG-NAME is the option's long name (without the double-dash). It defaults to nil.
- DESCRIPTION is the option's description appearing in help strings. It defaults to nil.
- ENV-VAR is the flag's associated environment variable.
It defaults to nil.
- When HIDDEN, the option doesn't appear in help strings.

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

Source [flag.lisp], page 15.

`make-group` (&rest *keys* &key *header item hidden*) [Function]

Make a new group.

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

Source [group.lisp], page 20.

`make-lispobj` (&rest *keys* &key *short-name long-name description argument-name argument-type env-var typespec fallback-value default-value hidden*) [Function]

Make a new lispobj option.
- SHORT-NAME is the option's short name (without the dash).
It defaults to nil.
- LONG-NAME is the option's long name (without the double-dash).
It defaults to nil.
- DESCRIPTION is the option's description appearing in help strings.
It defaults to nil.
- ARGUMENT-NAME is the option's argument name appearing in help strings. - ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENV-VAR is the option's associated environment variable.
It defaults to nil.
- TYPESPEC is a type specifier the option's value should satisfy.
- FALLBACK-VALUE is the option's fallback value (for missing optional arguments), if any.
- DEFAULT-VALUE is the option's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

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

Source [lispobj.lisp], page 18.

`make-path` (&rest *keys* &key *short-name long-name description argument-name argument-type env-var fallback-value default-value type hidden*) [Function]

Make a new path option.
- SHORT-NAME is the option's short name (without the dash).
It defaults to nil.
- LONG-NAME is the option's long name (without the double-dash).
It defaults to nil.
- DESCRIPTION is the option's description appearing in help strings.
It defaults to nil.
- ARGUMENT-NAME is the option's argument name appearing in help strings. - ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENV-VAR is the option's associated environment variable.
It defaults to nil.
- FALLBACK-VALUE is the option's fallback value (for missing optional arguments), if any.
- DEFAULT-VALUE is the option's default value, if any.
- TYPE is the pathname type. It can be one of :file, :directory, :file-list, :directory-list or nil meaning that everything is allowed.
- When HIDDEN, the option doesn't appear in help strings.

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

Source [path.lisp], page 18.

`make-stropt` (&rest *keys* &key *short-name long-name description argument-name argument-type env-var fallback-value default-value hidden*) [Function]

Make a new string option.
- SHORT-NAME is the option's short name (without the dash).
It defaults to nil.
- LONG-NAME is the option's long name (without the double-dash).
It defaults to nil.
- DESCRIPTION is the option's description appearing in help strings.
It defaults to nil.
- ARGUMENT-NAME is the option's argument name appearing in help strings. - ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENV-VAR is the option's associated environment variable.
It defaults to nil.
- FALLBACK-VALUE is the option's fallback value (for missing optional arguments), if any.
- DEFAULT-VALUE is the option's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

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

Source [`stropt.lisp`], page 17.

`make-switch` (&rest *keys* &key *short-name long-name description argument-style argument-type env-var default-value hidden*) [Function]

Make a new switch.
- SHORT-NAME is the switch's short name (without the dash).
It defaults to nil.
- LONG-NAME is the switch's long name (without the double-dash).
It defaults to nil.
- DESCRIPTION is the switch's description appearing in help strings.
It defaults to nil.
- ARGUMENT-STYLE is the switch's argument display style. It can be one of :yes/no, :on/off, :true/false, :yup/nope or :yeah/nah.
It defaults to :yes/no.
- ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENV-VAR is the switch's associated environment variable.
It defaults to nil.
- DEFAULT-VALUE is the switch's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

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

Source [`switch.lisp`], page 17.

`make-synopsis` (&rest *keys* &key *postfix item make-default*) [Function]

Make a new SYNOPSIS.
- POSTFIX is a string to append to the program synopsis, in case it accepts a remainder.
- If MAKE-DEFAULT, make the new synopsis the default one.

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

Source [`synopsis.lisp`], page 21.

`make-text` (&rest *keys* &key *contents hidden*) [Function]
Make a new text.
- CONTENTS is the actual text to display.
- When HIDDEN, the text doesn't appear in help strings.

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

Source [`text.lisp`], page 14.

`make-xswitch` (&rest *keys* &key *short-name long-name description argument-name argument-type enum env-var default-value hidden*) [Function]
Make a new xswitch.
- SHORT-NAME is the xswitch's short name (without the dash).
It defaults to nil.
- LONG-NAME is the xswitch's long name (without the double-dash).
It defaults to nil.
- DESCRIPTION is the xswitch's description appearing in help strings. It defaults to nil.
- ARGUMENT-NAME is the option's argument name appearing in help strings. - ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENUM is the set of possible non-boolean values.
- ENV-VAR is the xswitch's associated environment variable.
It defaults to nil.
- DEFAULT-VALUE is the xswitch's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

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

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

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

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

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

`progname` (&key *context*) [Function]
Return CONTEXT's program name.

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

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

`remainder` (&key *context*) [Function]
Return CONTEXT's remainder.

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

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

`setup-termio` () [Function]
Autodetect termio support.
Update Clon configuration and *FEATURES* accordingly.

Package [`net.didierverna.clon.setup`], page 36.

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

`version` (&optional *type*) [Function]
Return the current version of Clon.
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 "Michael Brecker". As for the short version, a patchlevel of 0 is ignored in the output.

Package [`net.didierverna.clon.setup`], page 36.

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

6.1.4 Standalone methods

`initialize-instance` *:around* ((*container* [`container`], *page 103*) &rest *keys* &key *item*) [Method]
Canonicalize initialization arguments.
This involves:
- computing the :items initarg from the :item ones.

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

`initialize-instance` *:after* ((*container* [`container`], *page 103*) &key) [Method]
Perform name clash check on CONTAINER's items.

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

`initialize-instance` *:after* ((*context* [`context`], *page 104*) &key *cmdline* *progname*) [Method]
Parse CMDLINE.

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

`initialize-instance` *:before* ((*enum-base* [`enum-base`], *page 106*) &key *enum*) [Method]

Source [`enum-base.lisp`], page 19.

`initialize-instance` *:around* ((*sheet* [`sheet`], *page 114*) &rest *keys* &key *output-stream* *line-width* *highlight*) [Method]
Handle unset line width and AUTO highlight according to OUTPUT-STREAM.

Source [`sheet.lisp`], page 23.

`initialize-instance` *:after* ((*sheet* [`sheet`], *page 114*) &key *theme* *search-path*) [Method]
Finish initialization of SHEET.
This involves:
- computing SHEET's sface tree from THEME and SEARCH-PATH,
- initializing SHEET's toplevel sface's sibling to a raw face tree.

Source [`sheet.lisp`], page 23.

`initialize-instance` *:after* ((*switch* *[*`switch`*], page 116*) &key) [Method]
Provide an argument name conformant to the selected argument style.

Source [`switch.lisp`], page 17.

`initialize-instance` *:before* ((*option* *[*`valued-option`*], page 119*) &key *argument-type fallback-value default-value*) [Method]
Check validity of the value-related initargs.

Source [`valued.lisp`], page 15.

`initialize-instance` *:after* ((*option* *[*`valued-option`*], page 119*) &key *argument-type fallback-value default-value*) [Method]
Compute uninitialized OPTION slots with indirect initargs.
This currently involves the conversion of the ARGUMENT-TYPE key to the ARGUMENT-REQUIRED-P slot.

Source [`valued.lisp`], page 15.

`initialize-instance` *:before* ((*option* *[*`option`*], page 112*) &key *short-name long-name description internal*) [Method]
Check validity of the name-related initargs.

Source [`option.lisp`], page 15.

`initialize-instance` *:around* ((*option* *[*`option`*], page 112*) &rest *keys* &key *long-name env-var internal*) [Method]
If INTERNAL, prefix LONG-NAME with "clon-" and ENV-VAR with "CLON_".

Source [`option.lisp`], page 15.

`initialize-instance` *:around* ((*instance* *[*`face`*], page 106*) &rest *keys* &key *face bold display hidden revealed*) [Method]
Canonicalize initialization arguments.
This involves:
- computing the :subfaces initarg from the :face ones, - handling convenience highlight properties.

Source [`face.lisp`], page 22.

`initialize-instance` *:before* ((*face* *[*`face`*], page 106*) &key *name subfaces*) [Method]
Check for unicity of FACE subfaces.

Source [`face.lisp`], page 22.

`initialize-instance` *:after* ((*face* *[*`face`*], page 106*) &key) [Method]
Fill in the parent slot of all subfaces.

Source [`face.lisp`], page 22.

`initialize-instance` *:before* ((*switch-base* *[*`switch-base`*], page 116*) &key *argument-style argument-styles*) [Method]
Check for validity of the :ARGUMENT-STYLE initarg.

Source [`switch-base.lisp`], page 17.

`initialize-instance` *:around* ((*switch-base* *[*`switch-base`*], page 116*) &rest *keys* &key *argument-type*) [Method]
Provide a fallback value of t when ARGUMENT-TYPE is optional.

Source [`switch-base.lisp`], page 17.

`initialize-instance` *:around* ((*synopsis* [`synopsis`], *page 118*) &rest *keys*) [Method]
Prepare Clon specific options.

Source [`synopsis.lisp`], page 21.

`initialize-instance` *:after* ((*synopsis* [`synopsis`], *page 118*) &key) [Method]
Compute SYNOSPSIS's short and negated packs.

Source [`synopsis.lisp`], page 21.

`make-instance` ((*class* [`abstract-class`], *page 103*) &rest *initargs*) [Method]

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

`slot-unbound` (*class* (*face* [`face`], *page 106*) *slot*) [Method]
Look up SLOT's value in FACE's parent if it's a highlight property. If FACE has no parent, return nil.
For other properties, trigger an error.

Source [`face.lisp`], page 22.

`validate-superclass` ((*class* [`abstract-class`], *page 103*) (*superclass* `standard-class`)) [Method]

Package `sb-mop`.

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

`validate-superclass` ((*class* `standard-class`) (*superclass* [`abstract-class`], *page 103*)) [Method]

Package `sb-mop`.

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

6.2 Internals

6.2.1 Special variables

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

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

Package [`net.didierverna.clon.setup`], page 36.

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

`*highlight-properties*` [Special Variable]
The highlight face properties.

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

Source [`face.lisp`], page 22.

`*item-names*` [Special Variable]
The list of defined item names.

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

Source [`valued.lisp`], page 15.

6.2.2 Macros

`%defgroup` (*internalp* (&rest *keys* &key *header hidden*) &body *forms*) [Macro]
Define a new group.

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

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

`accumulate` ((*initial-value*) &body *body*) [Macro]
Accumulate BODY forms in a list beginning with INITIAL-VALUE. INITIAL-VALUE is not evaluated. BODY forms are accumulated only when their value is non-nil.
If nothing to accumulate, then return nil instead of the list of INITIAL-VALUE.

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

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

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

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

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

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

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

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

`defindent` (*symbol indent*) [Macro]
Wrapper around 'clindent' to avoid quoting SYMBOL and INDENT.

Package [`net.didierverna.clon.setup`], page 36.

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

`defoption` (*class superclasses slots* &rest *options*) [Macro]
Create a new option CLASS and register it with Clon.

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

Source [`valued.lisp`], page 15.

`do-options` ((*opt there*) &body *body*) [Macro]
Execute BODY with OPT bound to every option in THERE.

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

Source [`synopsis.lisp`], page 21.

`econd` (&body *clauses*) [Macro]
Like COND, but signal an error if no clause evaluates to t.

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

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

`endpush` (*object place*) [Macro]
Like push, but at the end.

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

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

`highlight-property-ecase` (*property value* &body *clauses*) [Macro]
Create an ECASE form to extract PROPERTY's VALUE escape sequence.
Each clause looks like: (PROPERTY-NAME (VALUE-OR-VALUE-LIST ESCAPE-SEQUENCE)*). The value-matching part will itself be enclosed in an ECASE expression.
In addition, the special clause syntax (BOOLEAN <PROPERTY-NAME> <YES> <NO>) is a shortcut for: (PROPERTY-NAME ((on t) YES) ((off nil) NO)).

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

Source [`sheet.lisp`], page 23.

`map-frames` (*function* (*sheet* &key *reverse*)) [Macro]
Map FUNCTION over SHEET's frames. If REVERSE, map in reverse order.

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

Source [`sheet.lisp`], page 23.

`maybe-pop-argument` (*cmdline option cmdline-argument*) [Macro]
Pop OPTION's argument from CMDLINE if needed. If so, store it in CMDLINE-ARGUMENT.

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

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

`maybe-push` (*object place*) [Macro]
Like push, but only if OBJECT is non-nil.

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

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

`replace-in-keys` ((*key val*) *keys the-key form*) [Macro]
Replace every occurrence of THE-KEY in KEYS with FORM.
At every KEYS round, KEY and VAL are bound to the current key-value pair. FORM is evaluated each time and should return a key-value list.

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

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

`restartable-invalid-negated-syntax-error` ((*option*) &body *body*) [Macro]
Restartably throw an invalid-negated-syntax error.
The error relates to the command-line use of OPTION.
BODY constitutes the body of the only restart available,
use-short-call, and should act as if OPTION had been normally called by short name.

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

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

`restartable-spurious-cmdline-argument-error` ((*option name argument*) &body *body*) [Macro]
Restartably throw a spurious-cmdline-argument error.
The error relates to the command-line use of OPTION called by NAME with ARGUMENT.
BODY constitutes the body of the only restart available, discard-argument, and should act as if ARGUMENT had not been provided.

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

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

`with-context-error-handler` (*context* &body *body*) [Macro]
Execute BODY with CONTEXT's error handler bound for CONDITION.

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

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

6.2.3 Ordinary functions

`%version` (*type major minor status level name*) [Function]

Package [`net.didierverna.clon.setup`], page 36.

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

`add-subface` (*face subface*) [Function]
Add SUBFACE to FACE's subfaces and return it.

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

Source [`face.lisp`], page 22.

`argument-popable-p` (*cmdline*) [Function]
Return true if the first CMDLINE item is an argument.

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

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

`attach-face-tree` (*face face-tree*) [Function]
Create a copy of FACE-TREE, attach it to FACE and return it.
Apart from the parenting information, the copied faces share slot values with the original ones.

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

Source [`face.lisp`], page 22.

`available-right-margin` (*sheet*) [Function]
Return SHEET's available right margin.
This margin is the first non-self margin specified by a frame. All inner self frames can potentially write until the available right margin.

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

Source [`sheet.lisp`], page 23.

`beginning-of-string-p` (*beginning string* &optional *ignore-case*) [Function]
Check that STRING starts with BEGINNING. If IGNORE-CASE, well, ignore case.

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

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

`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.clon.setup`], page 36.

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

`close-line` (*sheet*) [Function]
Close all frames on SHEET's current line and go to next line.

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

Source [`sheet.lisp`], page 23.

`close-sface` (*sheet*) [Function]
Close SHEET's current sface.

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

Source [`sheet.lisp`], page 23.

`closest-match` (*match list* &key *ignore-case key*) [Function]
Return the LIST element closest to MATCH, or nil.
If IGNORE-CASE, well, ignore case.
KEY should provide a way to get a string from each LIST element.

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

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

`cmdline-convert` (*valued-option cmdline-name cmdline-argument*) [Function]
Convert CMDLINE-ARGUMENT to VALUED-OPTION's value.
This function is used when the conversion comes from a command-line usage of VALUED-OPTION, called by CMDLINE-NAME, and intercepts invalid-argument errors to raise the higher level invalid-cmdline-argument error instead.

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

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

`cmdline-option-name` (*instance*) [Reader]
`(setf cmdline-option-name)` (*instance*) [Writer]

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

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

Target Slot
[`name`], page 101.

`cmdline-option-option` (*instance*) [Reader]
`(setf cmdline-option-option)` (*instance*) [Writer]

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

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

Target Slot
[`option`], page 101.

`cmdline-option-p` (*object*) [Function]

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

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

`cmdline-option-source` (*instance*) [Reader]
`(setf cmdline-option-source)` (*instance*) [Writer]

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

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

Target Slot
[`source`], page 101.

`cmdline-option-value` (*instance*) [Reader]
`(setf cmdline-option-value)` (*instance*) [Writer]

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

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

Target Slot
[`value`], page 101.

`complete-string` (*beginning complete*) [Function]
Complete BEGINNING with the rest of COMPLETE in parentheses. For instance, completing 'he' with 'help' will produce 'he(lp)'.

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

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

`copy-cmdline-option` (*instance*) [Function]

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

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

`copy-frame` (*instance*) [Function]

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

Source [`sheet.lisp`], page 23.

`copy-highlight-frame` (*instance*) [Function]

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

Source [`sheet.lisp`], page 23.

`copy-highlight-property-instance` (*instance*) [Function]

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

Source [`sheet.lisp`], page 23.

`current-frame` (*sheet*) [Function]
Return SHEET's current frame.

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

Source [`sheet.lisp`], page 23.

`current-left-margin` (*sheet*) [Function]
Return SHEET's current left margin.

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

Source [`sheet.lisp`], page 23.

`current-right-margin` (*sheet*) [Function]
Return SHEET's current right margin.

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

Source [`sheet.lisp`], page 23.

`current-sface` (*sheet*) [Function]
Return SHEET's current sface or nil.

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

Source [`sheet.lisp`], page 23.

`directory-pathname-p` (*pathname*) [Function]
Return true if PATHNAME denotes a directory.

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

Source [`path.lisp`], page 18.

`environment-convert` (*valued-option env-val*) [Function]
Convert ENV-VAL to VALUED-OPTION's value.
This function is used when the conversion comes from an environment variable associated with VALUED-OPTION, and intercepts invalid-argument errors
to raise the higher level invalid-environment-value error instead.

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

Source [`environ.lisp`], page 21.

`exit-abnormally` (*error*) [Function]
Print ERROR on *ERROR-OUTPUT* and exit with status code 1.

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

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

`face-highlight-property-set-p` (*face property*) [Function]
Return t if PROPERTY is set explicitely in FACE.

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

Source [`face.lisp`], page 22.

`face-highlight-property-value` (*face property*) [Function]
Return PROPERTY's value in FACE.
Since faces inherit highlight properties, the actual value might come from one of FACE's ancestors.
if PROPERTY is not et, return nil.

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

Source [`face.lisp`], page 22.

`find-sface` (*sface name*) [Function]
Find an sface starting at SFACE named NAME.
If the sface can't be found in SFACE's face tree, find one in SFACE's sibling instead, and make a copy of it.

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

Source [`sheet.lisp`], page 23.

`flush-sheet` (*sheet*) [Function]
Flush SHEET.

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

Source [`sheet.lisp`], page 23.

`frame-left-margin` (*instance*) [Reader]
`(setf frame-left-margin)` (*instance*) [Writer]

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

Source [`sheet.lisp`], page 23.

Target Slot
[`left-margin`], page 102.

`frame-p` (*object*) [Function]

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

Source [`sheet.lisp`], page 23.

`frame-right-margin` (*instance*) [Reader]
`(setf frame-right-margin)` (*instance*) [Writer]

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

Source [`sheet.lisp`], page 23.

Target Slot
[`right-margin`], page 102.

`frame-sface` (*instance*) [Reader]
`(setf frame-sface)` (*instance*) [Writer]

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

Source [`sheet.lisp`], page 23.

Target Slot
[`sface`], page 102.

`get-top-padding` (*sface items*) [Function]
Return top padding of the next item in ITEMS that will print under SFACE.

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

Source [`sheet.lisp`], page 23.

`getenv` (*variable*) [Function]
Get environment VARIABLE's value. VARIABLE may be null.

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

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

`help-spec-items-will-print` (*sface items*) [Function]
Return t if at least one of ITEMS will print under SFACE.

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

Source [`sheet.lisp`], page 23.

`highlight-frame-highlight-property-instances` (*instance*) [Reader]
`(setf highlight-frame-highlight-property-instances)` (*instance*) [Writer]

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

Source [`sheet.lisp`], page 23.

Target Slot
[`highlight-property-instances`], page 102.

`highlight-frame-left-margin` (*instance*) [Function]

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

Source [`sheet.lisp`], page 23.

`(setf highlight-frame-left-margin)` (*instance*) [Function]

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

Source [`sheet.lisp`], page 23.

`highlight-frame-p` (*object*) [Function]

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

Source [`sheet.lisp`], page 23.

`highlight-frame-right-margin` (*instance*) [Function]

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

Source [`sheet.lisp`], page 23.

`(setf highlight-frame-right-margin)` (*instance*) [Function]

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

Source [`sheet.lisp`], page 23.

`highlight-frame-sface` (*instance*) [Function]

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

Source [`sheet.lisp`], page 23.

`(setf highlight-frame-sface)` (*instance*) [Function]

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

Source [`sheet.lisp`], page 23.

`highlight-property-instance-escape-sequence` (*instance*) [Function]

Return highlight property INSTANCE's escape sequence.

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

Source [`sheet.lisp`], page 23.

`highlight-property-instance-name` (*instance*) [Reader]
`(setf highlight-property-instance-name)` (*instance*) [Writer]

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

Source [`sheet.lisp`], page 23.

Target Slot

[`name`], page 102.

`highlight-property-instance-p` (*object*) [Function]

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

Source [`sheet.lisp`], page 23.

`highlight-property-instance-value` (*instance*) [Reader]
`(setf highlight-property-instance-value)` (*instance*) [Writer]

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

Source [`sheet.lisp`], page 23.

Target Slot

[`value`], page 103.

`home-directory` () [Function]

Return user's home directory in canonical form.
If the user's home directory cannot be computed, signal a warning and return NIL.

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

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

`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.CLON named readtable.

Package [`net.didierverna.clon.setup`], page 36.

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

`list-to-string` (*list* &key *key separator*) [Function]
Return a SEPARATOR-separated string of all LIST elements.
- KEY should provide a way to get a string from each LIST element. - SEPARATOR is the string to insert between elements.

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

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

`macosp` () [Function]
Return t if running on Mac OS.

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

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

`make-cmdline-option` (&key *name option value source*) [Function]

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

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

`make-frame` (&key *sface left-margin right-margin*) [Function]

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

Source [`sheet.lisp`], page 23.

`make-highlight-frame` (&key *sface left-margin right-margin highlight-property-instances*) [Function]

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

Source [`sheet.lisp`], page 23.

`make-highlight-property-instance` (&key *name value*) [Function]

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

Source [`sheet.lisp`], page 23.

`make-internal-enum` (*long-name description* &rest *keys* &key *argument-name argument-type enum env-var fallback-value default-value hidden*) [Function]
Make a new internal (Clon-specific) enum option.
- LONG-NAME is the option's long-name, sans the 'clon-' prefix. (Internal options don't have short names.)
- DESCRIPTION is the options's description.
- ARGUMENT-NAME is the option's argument name appearing in help strings. - ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENUM is the set of possible values.
- ENV-VAR is the option's associated environment variable, sans the 'CLON_' prefix. It

defaults to nil.
- FALLBACK-VALUE is the option's fallback value (for missing optional arguments), if any.
- DEFAULT-VALUE is the option's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

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

Source [enum.lisp], page 19.

`make-internal-flag` (*long-name description* &rest *keys* &key *env-var hidden*) [Function]

Make a new internal (Clon-specific) flag.
- LONG-NAME is the flag's long-name, sans the 'clon-' prefix. (Internal options don't have short names.)
- DESCRIPTION is the flag's description.
- ENV-VAR is the flag's associated environment variable, sans the 'CLON_' prefix. It default to nil.
- When HIDDEN, the option doesn't appear in help strings.

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

Source [flag.lisp], page 15.

`make-internal-lispobj` (*long-name description* &rest *keys* &key *argument-name argument-type env-var typespec fallback-value default-value hidden*) [Function]

Make a new internal (Clon-specific) string option.
- LONG-NAME is the option's long-name, sans the 'clon-' prefix. (Internal options don't have short names.)
- DESCRIPTION is the options's description.
- ARGUMENT-NAME is the option's argument name appearing in help strings. - ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENV-VAR is the option's associated environment variable, sans the 'CLON_' prefix. It defaults to nil.
- TYPESPEC is a type specifier the option's value should satisfy.
- FALLBACK-VALUE is the option's fallback value (for missing optional arguments), if any.
- DEFAULT-VALUE is the option's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

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

Source [lispobj.lisp], page 18.

`make-internal-path` (*long-name description* &rest *keys* &key *argument-name argument-type env-var fallback-value default-value type hidden*) [Function]

Make a new internal (Clon-specific) path option.
- LONG-NAME is the option's long-name, sans the 'clon-' prefix. (Internal options don't have short names.)
- DESCRIPTION is the options's description.
- ARGUMENT-NAME is the option's argument name appearing in help strings. - ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.

- ENV-VAR is the option's associated environment variable, sans the 'CLON_' prefix. It defaults to nil.
- FALLBACK-VALUE is the option's fallback value (for missing optional arguments), if any.
- DEFAULT-VALUE is the option's default value, if any.
- TYPE is the pathname type. It can be one of :file, :directory, :file-list, :directory-list or nil meaning that everything is allowed.
- When HIDDEN, the option doesn't appear in help strings.

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

Source [path.lisp], page 18.

`make-internal-stropt` (*long-name description* &rest *keys* &key *argument-name argument-type env-var fallback-value default-value hidden*) [Function]

Make a new internal (Clon-specific) string option.
- LONG-NAME is the option's long-name, sans the 'clon-' prefix. (Internal options don't have short names.)
- DESCRIPTION is the options's description.
- ARGUMENT-NAME is the option's argument name appearing in help strings. - ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENV-VAR is the option's associated environment variable, sans the 'CLON_' prefix. It defaults to nil.
- FALLBACK-VALUE is the option's fallback value (for missing optional arguments), if any.
- DEFAULT-VALUE is the option's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

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

Source [stropt.lisp], page 17.

`make-internal-switch` (*long-name description* &rest *keys* &key *argument-style argument-type env-var default-value hidden*) [Function]

Make a new internal (Clon-specific) switch.
- LONG-NAME is the switch's long-name, sans the 'clon-' prefix. (Internal options don't have short names.)
- DESCRIPTION is the switch's description.
- ARGUMENT-STYLE is the switch's argument display style. It can be one of :yes/no, :on/off, :true/false, :yup/nope or :yeah/nah.
It defaults to :yes/no.
- ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENV-VAR is the switch's associated environment variable, sans the 'CLON_' prefix. It defaults to nil.
- DEFAULT-VALUE is the switch's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

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

Source [switch.lisp], page 17.

`make-internal-text` (&rest *keys* &key *contents hidden*) [Function]

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

Source [text.lisp], page 14.

`make-internal-xswitch` (*long-name description* &rest *keys* &key *argument-name argument-type enum env-var default-value hidden*) [Function]

Make a new internal (Clon-specific) xswitch.
- LONG-NAME is the xswitch's long-name, sans the 'clon-' prefix. (Internal options don't have short names.)
- DESCRIPTION is the xswitch's description.
- ARGUMENT-NAME is the option's argument name appearing in help strings. - ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).
It defaults to :optional.
- ENUM is the set of possible non-boolean values.
- ENV-VAR is the xswitch's associated environment variable, sans the 'CLON_' prefix. It defaults to nil.
- DEFAULT-VALUE is the xswitch's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

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

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

`make-raw-face-tree` (&optional *face-class*) [Function]

Make a raw (boring yet functional) face tree.

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

Source [`face.lisp`], page 22.

`make-raw-sface` (*sibling*) [Function]

Return a new SFace based on SIBLING.
This function does not consider SIBLING as a face tree:
only face properties are copied; the face parent and children are set to nil.

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

Source [`sheet.lisp`], page 23.

`make-sheet` (&rest *keys* &key *output-stream search-path theme line-width highlight*) [Function]

Make a new SHEET.

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

Source [`sheet.lisp`], page 23.

`match-option` (*option* &key *short-name long-name*) [Function]

Try to match OPTION against SHORT-NAME, LONG-NAME. If OPTION matches, return the name that matched.

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

Source [`option.lisp`], page 15.

`open-line` (*sheet*) [Function]

Open all frames on SHEET's current line.

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

Source [`sheet.lisp`], page 23.

`open-next-line` (*sheet*) [Function]
Close SHEET's current line and open the next one.

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

Source [`sheet.lisp`], page 23.

`open-sface` (*sheet sface*) [Function]
Create a frame for SFACE and open it.

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

Source [`sheet.lisp`], page 23.

`option-abbreviation-distance` (*option partial-name*) [Function]
Return the distance between OPTION's long name and PARTIAL-NAME. If PARTIAL-NAME does not abbreviate OPTION's long name, return MOST-POSITIVE-FIXNUM.

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

Source [`option.lisp`], page 15.

`option-call-p` (*str*) [Function]
Return true if STR looks like an option call.

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

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

`parent-generation` (*face parent-name*) [Function]
Return FACE's parent generation for PARENT-NAME.
That is, 1 if PARENT-NAME names FACE's parent, 2 if it names its grand-parent etc. If PARENT-NAME does not name one of FACE's ancestors, trigger an error.

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

Source [`face.lisp`], page 22.

`pathname-component-null-p` (*component*) [Function]
Return true if COMPONENT is either null or :unspecific.

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

Source [`path.lisp`], page 18.

`pop-frame` (*sheet*) [Function]
Pop SHEET's current frame.

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

Source [`sheet.lisp`], page 23.

`potential-pack-char` (*option* &optional *as-string*) [Function]
Return OPTION's potential pack character, if any. If AS-STRING, return a string of that character.

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

Source [`option.lisp`], page 15.

`princ-char` (*sheet char*) [Function]
Princ CHAR on SHEET's stream and increment the column position.
The effect of printing CHAR must be exactly to move right by one column, so control characters, as well as newlines and tabs are forbidden here.

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

Source [`sheet.lisp`], page 23.

`princ-highlight-property-instances` (*sheet instances*) [Function]
Princ highlight proeprty INSTANCES on SHEET's stream.

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

Source [`sheet.lisp`], page 23.

`princ-spaces` (*sheet number*) [Function]
Princ NUMBER spaces to SHEET's stream and update the column position.

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

Source [`sheet.lisp`], page 23.

`princ-string` (*sheet string*) [Function]
Princ STRING on SHEET's stream and update the column position.
The effect of printing STRING must be exactly to move right by the corresponding string length, so control characters, as well as newlines and tabs are forbidden here.

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

Source [`sheet.lisp`], page 23.

`print-error` (*error* &optional *interactivep*) [Function]
Print ERROR on *ERROR-OUTPUT*.
When INTERACTIVEP, print on *QUERY-IO* instead.

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

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

`print-faced-help-spec` (*sheet sface items*) [Function]
Print all help specification ITEMS on SHEET with SFACE.

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

Source [`sheet.lisp`], page 23.

`print-help` (*sheet help*) [Function]
Open the toplevel help face and print HELP on SHEET with it.

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

Source [`sheet.lisp`], page 23.

`print-string` (*sheet string*) [Function]
Output STRING to SHEET.
STRING is output within the current frame's bounds.
Spacing characters are honored but newlines might replace spaces when the output reaches the rightmost bound.

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

Source [`sheet.lisp`], page 23.

`push-frame` (*sheet frame*) [Function]
Push a new frame to SHEET's frames.

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

Source [`sheet.lisp`], page 23.

`putenv` (*variable value*) [Function]
Set environment VARIABLE to VALUE.

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

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

`reach-column` (*sheet column*) [Function]
Reach COLUMN on SHEET by princ'ing spaces.

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

Source [`sheet.lisp`], page 23.

`read-argument` () [Function]
Read an option argument from standard input.

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

Source [`valued.lisp`], page 15.

`read-call` (&optional *negated*) [Function]
Read an option's call or pack from standard input.
If NEGATED, read a negated call or pack. Otherwise, read a short call or pack.

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

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

`read-env-val` (*env-var*) [Function]
Read ENV-VAR's new value from standard input.

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

Source [`environ.lisp`], page 21.

`read-long-name` () [Function]
Read an option's long name from standard input.

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

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

`read-sface-tree` (*pathname*) [Function]
Read an sface tree from PATHNAME.

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

Source [`sheet.lisp`], page 23.

`read-value` () [Function]
Read an option value from standard input.

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

Source [`valued.lisp`], page 15.

`release-status-number` (*release-status*) [Function]

Package [`net.didierverna.clon.setup`], page 36.

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

`remove-keys` (*keys* &rest *removed*) [Function]

Return a new property list from KEYS without REMOVED ones.

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

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

`replace-key` (*replacement keys*) [Function]

Return a new property list from KEYS with REPLACEMENT.
REPLACEMENT can take the following forms:
- :KEY
The effect is to remove :KEY from KEYS, as per REMOVE-KEYS.
- (:KEY :NEW-KEY)
The effect is to replace :KEY with :NEW-KEY, leaving the values unchanged. - (:KEY :NEW-KEY (VAL-OR-VALS NEW-VAL)*), with VAL-OR-VALS being
either a value or a list of values. The effect is to replace :KEY with :NEW-KEY and a value matching one of the VAL-OR-VALS with the corresponding NEW-VAL. Values not matching any VAL-OR-VALS remain unchanged. - (:KEY (VAL-OR-VALS :NEW-KEY NEW-VAL...)*), with VAL-OR-VALS as above. The effect is the same as above, but :NEW-KEY additionally depends on the matched value. If multiple :NEW-KEY NEW-VAL couples are provided, that many new keys are inserted along with their values. For values not matching any VAL-OR-VALS, :KEY and its value remain unchanged.

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

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

`replace-keys` (*keys* &rest *replacements*) [Function]

Return a new property list from KEYS with REPLACEMENTS.
See REPLACE-KEY for more information on the replacement syntax.

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

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

`restart-on-error` (*error*) [Function]

Print ERROR and offer available restarts on *QUERY-IO*.

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

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

`restartable-check` (*valued-option value*) [Function]

Restartably check that VALUE is valid for VALUED-OPTION.
The only restart available, use-value, offers to try a different value from the one that was provided.

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

Source [`valued.lisp`], page 15.

`restartable-cmdline-convert` (*valued-option cmdline-name cmdline-argument*) [Function]

Restartably convert CMDLINE-ARGUMENT to VALUED-OPTION's value.
This function is used when the conversion comes from a command-line usage of VALUED-OPTION, called by CMDLINE-NAME.

As well as conversion errors, this function might raise a missing-cmdline-argument error if CMDLINE-ARGUMENT is nil and an argument is required.

Available restarts are (depending on the context):
- use-fallback-value: return FALLBACK-VALUE,
- use-default-value: return VALUED-OPTION's default value,
- use-value: return another (already converted) value,
- use-argument: return the conversion of another argument.

Return two values: VALUED-OPTION's value and the actual value source. The value source may be :cmdline, :fallback or :default.

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

Source [cmdline.lisp], page 20.

`restartable-cmdline-junk-error` (*junk*) [Function]

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

Source [context.lisp], page 25.

`restartable-convert` (*valued-option argument*) [Function]

Restartably convert ARGUMENT to VALUED-OPTION's value. Available restarts are:
- use-default-value: return OPTION's default value,
- use-value: return another (already converted) value, - use-argument: return the conversion of another argument.

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

Source [valued.lisp], page 15.

`restartable-environment-convert` (*valued-option env-val*) [Function]

Restartably convert ENV-VAL to VALUED-OPTION's value.
This function is used when the conversion comes from an environment variable associated with VALUED-OPTION.

Available restarts are:
- use-default-value: return VALUED-OPTION's default value,
- use-value: return another (already converted) value,
- use-argument: return the conversion of another argument,
- modify-env: modify the environment variable's value.

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

Source [environ.lisp], page 21.

`restrict-because` (*reason*) [Function]

Put Clon in restricted mode because of REASON.

Package [net.didierverna.clon.setup], page 36.

Source [termio.lisp], page 13.

`safe-left-margin` (*sheet margin*) [Function]

Return either MARGIN or a safe value instead.
To be safe, margin must be greater than the current left margin and smaller than the currently available margin.

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

Source [sheet.lisp], page 23.

`safe-right-margin` (*sheet left-margin margin*) [Function]
Return either MARGIN or a safe value instead.
To be safe, margin must be greater than LEFT-MARGIN and smaller than the currently available right margin.

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

Source [sheet.lisp], page 23.

`search-branch` (*face names*) [Function]
Search for a branch of faces named NAMES starting at FACE.
The branch is searched for as a direct subbranch of FACE, or as a direct subbranch of FACE's ancestors.
If a branch is found, return its leaf face. Otherwise return nil.

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

Source [face.lisp], page 22.

`search-face` (*face name* &optional *error-me*) [Function]
Search for a face named NAME starting at FACE.
The face is looked for as a direct subface of FACE (in which case it is simply returned), or up in the hierarchy and by successive upper branches (in which case it is copied and attached to FACE).
If ERROR-ME, trigger an error if no face is found; otherwise, return nil.

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

Source [face.lisp], page 22.

`search-option` (*context* &rest *keys* &key *short-name long-name partial-name*) [Function]
Search for an option in CONTEXT.
The search is done with SHORT-NAME, LONG-NAME, or PARTIAL-NAME.
In case of a PARTIAL-NAME search, look for an option the long name of which begins with it.
In case of multiple matches by PARTIAL-NAME, the longest match is selected. When such an option exists, return wo values:
- the option itself,
- the name used to find the option, possibly completed if partial.

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

Source [context.lisp], page 25.

`search-option-by-abbreviation` (*context partial-name*) [Function]
Search for option abbreviated with PARTIAL-NAME in CONTEXT. When such an option exists, return two values:
- the option itself,
- the completed name.

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

Source [context.lisp], page 25.

`search-option-by-name` (*context* &rest *keys* &key *short-name long-name*) [Function]
Search for option with either SHORT-NAME or LONG-NAME in CONTEXT. When such an option exists, return two values:
- the option itself,
- the name that matched.

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

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

`search-sticky-option` (*context namearg*) [Function]
Search for a sticky option in CONTEXT, matching NAMEARG.
NAMEARG is the concatenation of the option's short name and its argument. In case of multiple matches, the option with the longest name is selected. When such an option exists, return two values:
- the option itself,
- the argument part of NAMEARG.

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

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

`select-keys` (*keys* &rest *selected*) [Function]
Return a new property list from KEYS with only SELECTED ones.

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

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

`split-path` (*path*) [Function]
Split PATH into a list of directories.

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

Source [`path.lisp`], page 18.

`stream-line-width` (*stream*) [Function]
Get STREAM's line width.
Return two values:
- the stream's line width, or nil if it can't be computed (typically when the stream does not denote a tty), - an error message if the operation failed.

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

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

`try-read-sface-tree` (*pathname*) [Function]
Read an sface tree from PATHNAME if it exists or return nil.

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

Source [`sheet.lisp`], page 23.

`try-read-theme` (*pathname*) [Function]
Read a theme from PATHNAME or PATHNAME.cth if it exists or return nil.

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

Source [`sheet.lisp`], page 23.

`~-reader` (*stream char*) [Function]
Read a series of `~"string"` to be concatenated together.

Package [`net.didierverna.clon.setup`], page 36.

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

6.2.4 Generic functions

`argument` (*condition*) [Generic Reader]

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

Methods

`argument` ((*condition* [`unknown-cmdline-option-error`], *page 100*)) [Reader Method]

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

Target Slot
[`argument`], page 100.

`argument` ((*condition* [`spurious-cmdline-argument`], *page 99*)) [Reader Method]

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

Target Slot
[`argument`], page 100.

`argument` ((*condition* [`invalid-argument`], *page 97*)) [Reader Method]

Source [`valued.lisp`], page 15.

Target Slot
[`argument`], page 97.

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

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

Methods

`argument-name` ((*valued-option* [`valued-option`], *page 119*)) [Reader Method]
The option's argument display name.

Source [`valued.lisp`], page 15.

Target Slot
[`argument-name`], page 120.

`argument-required-p` (*object*) [Generic Reader]

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

Methods

`argument-required-p` ((*valued-option* [`valued-option`], *page 119*)) [Reader Method]
Whether the option's argument is required.

Source [`valued.lisp`], page 15.

Target Slot
[`argument-required-p`], page 120.

`argument-style` (*object*) [Generic Reader]

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

Methods

`argument-style` ((*switch-base [*`switch-base`*], page 116*)) [Reader Method]
The selected argument style.

Source [`switch-base.lisp`], page 17.

Target Slot
[`argument-style`], page 117.

`argument-styles` (*object*) [Generic Reader]
`(setf argument-styles)` (*object*) [Generic Writer]

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

Methods

`argument-styles` ((*switch-base [*`switch-base`*], page 116*)) [Reader Method]
`(setf argument-styles)` ((*switch-base [*`switch-base`*], page 116*)) [Writer Method]
The possible argument styles.
The position of every argument style in the list must correspond to the position of the associated strings in the yes-values and no-values slots.

Source [`switch-base.lisp`], page 17.

Target Slot
[`argument-styles`], page 117.

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

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

Methods

`background` ((*face [*`face`*], page 106*)) [Reader Method]
The face background.

Source [`face.lisp`], page 22.

Target Slot
[`background`], page 110.

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

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

Methods

`blink` ((*face [*`face`*], page 106*)) [Reader Method]
The face's blink speed.

Source [`face.lisp`], page 22.

Target Slot
[`blink`], page 109.

`bottom-padding` (*object*) [Generic Reader]

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

Methods

`bottom-padding` ((*face [face], page 106*)) [Reader Method]
The face bottom padding.
This property can take the following forms:
- nil: the next output can start right at the end of this face's, - 0: the next output should start on the next line,
- N>0: there should be N empty lines before the next output.

Source [face.lisp], page 22.

Target Slot
[bottom-padding], page 108.

`check` (*valued-option value*) [Generic Function]
Check that VALUE is valid for VALUED-OPTION.
If VALUE is valid, return it. Otherwise, raise an invalid-value error.

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

Source [valued.lisp], page 15.

Methods

`check` ((*xswitch [xswitch], page 121*) *value*) [Method]
Check that VALUE is valid for XSWITCH.

Source [xswitch.lisp], page 19.

`check` ((*enum [enum], page 105*) *value*) [Method]
Check that VALUE is valid for ENUM.

Source [enum.lisp], page 19.

`check` ((*path [path], page 113*) *value*) [Method]
Check that VALUE is valid for PATH.

Source [path.lisp], page 18.

`check` ((*lispobj [lispobj], page 111*) *value*) [Method]
Check that VALUE is valid for LISPOBJ.

Source [lispobj.lisp], page 18.

`check` ((*stropt [stropt], page 116*) *value*) [Method]
Check that VALUE is valid for STROPT.

Source [stropt.lisp], page 17.

`check` ((*switch [switch], page 116*) *value*) [Method]
Check that VALUE is valid for SWITCH.

Source [switch.lisp], page 17.

`check-name-clash` (*item1 item2*) [Generic Function]
Check for name clash between ITEM1's options and ITEM2's options.

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

Source [option.lisp], page 15.

Methods

`check-name-clash` ((*container1* [`container`], *page 103*) (*container2* [`container`], *page 103*)) [Method]
Check for name clash between CONTAINER1's options and CONTAINER2's ones.

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

`check-name-clash` (*item1* (*container* [`container`], *page 103*)) [Method]
Check for name clash between ITEM1's options and CONTAINER's ones.

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

`check-name-clash` ((*container* [`container`], *page 103*) *item2*) [Method]
Check for name clash between CONTAINER's options and ITEM2's ones.

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

`check-name-clash` (*item1* (*text* [`text`], *page 119*)) [Method]
Do nothing (no name clash with a text object.

`check-name-clash` ((*text* [`text`], *page 119*) *item2*) [Method]
Do nothing (no name clash with a text object.

`check-name-clash` ((*option1* [`option`], *page 112*) (*option2* [`option`], *page 112*)) [Method]
Ensure that there is no name clash between OPTION1 and OPTION2.

`clon-options-group` (*object*) [Generic Function]

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

Methods

`clon-options-group` ((*context* [`context`], *page 104*)) [Method]
Return the Clon options group of CONTEXT's synopsis.

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

`clon-options-group` ((*synopsis* [`synopsis`], *page 118*)) [Reader Method]
The Clon options group.

Source [`synopsis.lisp`], page 21.

Target Slot
[`clon-options-group`], page 119.

`close-frame` (*sheet frame*) [Generic Function]
Close FRAME on SHEET.

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

Source [`sheet.lisp`], page 23.

Method Combination
`progn`.

Options `:most-specific-last`

Methods

`close-frame` *progn* (*sheet* (*frame* [`frame`], *page 101*)) [Method]
Reach FRAME's right margin if it has one.

`close-frame` *progn* (*sheet* (*frame* [`highlight-frame`], *page 102*)) [Method]
Restore the upper frame's highlight properties.

`cmdline-options` (*object*) [Generic Reader]
`(setf cmdline-options)` (*object*) [Generic Writer]

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

Methods

`cmdline-options` ((*context* [`context`], *page 104*)) [Reader Method]
`(setf cmdline-options)` ((*context* [`context`], *page 104*)) [Writer Method]
The options from the command-line.

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

Target Slot
[`cmdline-options`], page 105.

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

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

Methods

`column` ((*sheet* [`sheet`], *page 114*)) [Reader Method]
`(setf column)` ((*sheet* [`sheet`], *page 114*)) [Writer Method]
The sheet's current column.

Source [`sheet.lisp`], page 23.

Target Slot
[`column`], page 115.

`comment` (*condition*) [Generic Reader]

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

Methods

`comment` ((*condition* [`invalid-argument`], *page 97*)) [Reader Method]

Source [`valued.lisp`], page 15.

Target Slot
[`comment`], page 97.

`comment` ((*condition* [`invalid-value`], *page 98*)) [Reader Method]

Source [`valued.lisp`], page 15.

Target Slot
[`comment`], page 99.

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

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

Methods

`concealedp` ((*face* [`face`], *page 106*)) [Reader Method]
The face's concealed status.

Source [`face.lisp`], page 22.

Target Slot
[`concealedp`], page 109.

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

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

Methods

`contents` ((*text* [`text`], *page 119*)) [Reader Method]
The actual text string.

Source [`text.lisp`], page 14.

Target Slot
[`contents`], page 119.

`convert` (*valued-option argument*) [Generic Function]
Convert ARGUMENT to VALUED-OPTION's value.
If ARGUMENT is invalid, raise an invalid-argument error.

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

Source [`valued.lisp`], page 15.

Methods

`convert` ((*xswitch* [`xswitch`], *page 121*) *argument*) [Method]
Convert ARGUMENT to an XSWITCH value.

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

`convert` ((*enum* [`enum`], *page 105*) *argument*) [Method]
Convert ARGUMENT to an ENUM value.

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

`convert` ((*path* [`path`], *page 113*) *argument*) [Method]
Convert ARGUMENT to a PATH value.

Source [`path.lisp`], page 18.

`convert` ((*lispobj* [`lispobj`], *page 111*) *argument*) [Method]
Convert ARGUMENT to a LISPOBJ value.

Source [`lispobj.lisp`], page 18.

`convert` ((*stropt* [`stropt`], *page 116*) *argument*) [Method]
Convert ARGUMENT to an STROPT value.

Source [`stropt.lisp`], page 17.

`convert` ((*switch* [`switch`], *page 116*) *argument*) [Method]
Convert ARGUMENT to a SWITCH value.

Source [`switch.lisp`], page 17.

`copy-instance` (*instance* &optional *subclass*) [Generic Function]
Return a copy of INSTANCE.
Copy is either an object of INSTANCE's class, or INSTANCE's SUBCLASS if given.

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

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

Methods

`copy-instance` (*instance* &optional *subclass*) [Method]
Return a copy of INSTANCE.
Both instances share the same slot values.

`crossed-out-p` (*object*) [Generic Reader]

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

Methods

`crossed-out-p` ((*face* [`face`], *page 106*)) [Reader Method]
The face's crossed out status.

Source [`face.lisp`], page 22.

Target Slot
[`crossed-out-p`], page 109.

`default-value` (*object*) [Generic Reader]

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

Methods

`default-value` ((*valued-option* [`valued-option`], *page 119*)) [Reader Method]
The option's default value.

Source [`valued.lisp`], page 15.

Target Slot
[`default-value`], page 120.

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

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

Methods

`description` ((*option* [`option`], *page 112*)) [Reader Method]
The option's description.

Source [`option.lisp`], page 15.

Target Slot
[`description`], page 113.

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

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

Methods

`enum` ((*enum-base [`enum-base`], page 106*)) [Reader Method]
The set of possible values.

Source [`enum-base.lisp`], page 19.

Target Slot
[`enum`], page 106.

`env-val` (*condition*) [Generic Reader]

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

Methods

`env-val` ((*condition [`invalid-environment-value`], page 97*)) [Reader Method]

Source [`environ.lisp`], page 21.

Target Slot
[`argument`], page 98.

`env-var` (*object*) [Generic Reader]

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

Methods

`env-var` ((*condition [`environment-error`], page 96*)) [Reader Method]

Source [`environ.lisp`], page 21.

Target Slot
[`env-var`], page 96.

`env-var` ((*option [`option`], page 112*)) [Reader Method]
The option's associated environment variable.

Source [`option.lisp`], page 15.

Target Slot
[`env-var`], page 113.

`error-handler` (*object*) [Generic Reader]

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

Methods

`error-handler` ((*context [`context`], page 104*)) [Reader Method]
The behavior to adopt on option retrieval errors.

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

Target Slot
[`error-handler`], page 105.

`error-string` (*condition*) [Generic Reader]
`(setf error-string)` (*condition*) [Generic Writer]

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

Methods

`error-string` ((*condition* [`home-directory`], *page 96*)) [Reader Method]
`(setf error-string)` ((*condition* [`home-directory`], *page 96*)) [Writer Method]

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

Target Slot
[`error-string`], page 96.

`fallback-value` (*object*) [Generic Reader]

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

Methods

`fallback-value` ((*valued-option* [`valued-option`], *page 119*)) [Reader Method]
The option's fallback value.

Source [`valued.lisp`], page 15.

Target Slot
[`fallback-value`], page 120.

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

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

Methods

`foreground` ((*face* [`face`], *page 106*)) [Reader Method]
The face foreground.

Source [`face.lisp`], page 22.

Target Slot
[`foreground`], page 109.

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

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

Methods

`framedp` ((*face* [`face`], *page 106*)) [Reader Method]
The face's framed status.

Source [`face.lisp`], page 22.

Target Slot
[`framedp`], page 109.

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

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

Methods

`frames` ((*sheet* [`sheet`], *page 114*)) [Reader Method]
`(setf frames)` ((*sheet* [`sheet`], *page 114*)) [Writer Method]
The stack of currently open frames.

Source [`sheet.lisp`], page 23.

Target Slot
[`frames`], page 115.

`get-bottom-padding` (*sface help-spec*) [Generic Function]
Get HELP-SPEC's bottom-padding under SFACE.

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

Source [`sheet.lisp`], page 23.

Methods

`get-bottom-padding` (*sface help-spec*) [Method]
Basic help specifications (chars, strings etc) don't provide a bottom padding.

`get-bottom-padding` (*sface* (*help-spec* `list`)) [Method]
Return the bottom padding of HELP-SPEC's face.

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

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

Methods

`header` ((*group [*`group`*], page 110*)) [Reader Method]
The group's header.

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

Target Slot
[`header`], page 110.

`help-spec` (*item* &key *program unhide* &allow-other-keys) [Generic Function]
Return ITEM's help specification.

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

Source [`item.lisp`], page 14.

Methods

`help-spec` ((*synopsis [*`synopsis`*], page 118*) &key *program*) [Method]
Return SYNOPSIS's help specification.

Source [`synopsis.lisp`], page 21.

`help-spec` ((*group [*`group`*], page 110*) &key) [Method]
Return GROUP's help specification.

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

`help-spec` ((*container [*`container`*], page 103*) &key) [Method]
Return CONTAINER's help specification.

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

`help-spec` ((*option [*`valued-option`*], page 119*) &key) [Method]
Return OPTION's help specification.

Source [`valued.lisp`], page 15.

`help-spec` ((*option [*`option`*], page 112*) &key) [Method]
Return OPTION's help specification.

Source [`option.lisp`], page 15.

`help-spec` ((*text [`text`], page 119*) &key) [Method]
Return TEXT's help specification.

Source [`text.lisp`], page 14.

`help-spec` *:around* ((*item [`item`], page 111*) &key *unhide*) [Method]
Call the actual method only when ITEM is not hidden or UNHIDE.

`help-spec-will-print` (*sface help-spec*) [Generic Function]
Return t if HELP-SPEC will print under FACE.

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

Source [`sheet.lisp`], page 23.

Methods

`help-spec-will-print` *:before* (*sface help-spec*) [Method]

`help-spec-will-print` (*sface help-spec*) [Method]
Basic help specifications (chars, strings etc) do print.

`help-spec-will-print` (*sface* (*help-spec* `list`)) [Method]
Return t if HELP-SPEC's items will print under HELP-SPEC's face.

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

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

Methods

`hiddenp` ((*item [`item`], page 111*)) [Reader Method]
Whether the item is hidden in help strings.

Source [`item.lisp`], page 14.

Target Slot
[`hiddenp`], page 111.

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

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

Methods

`highlight` ((*context [`context`], page 104*)) [Reader Method]
Clon's output highlight mode.

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

Target Slot
[`highlight`], page 105.

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

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

Methods

`highlightp` ((*sheet [`sheet`], page 114*)) [Reader Method]
Whether to highlight SHEET's output.

Source [`sheet.lisp`], page 23.

Target Slot
[`highlightp`], page 115.

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

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

Methods

`intensity` ((*face* [`face`], *page 106*)) [Reader Method]

The face intensity.

Source [`face.lisp`], page 22.

Target Slot

[`intensity`], page 108.

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

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

Methods

`inversep` ((*face* [`face`], *page 106*)) [Reader Method]

The face's inverse video status.

Source [`face.lisp`], page 22.

Target Slot

[`inversep`], page 109.

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

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

Methods

`italicp` ((*face* [`face`], *page 106*)) [Reader Method]

The face's italic status.

Source [`face.lisp`], page 22.

Target Slot

[`italicp`], page 108.

`item` (*condition*) [Generic Reader]

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

Methods

`item` ((*condition* [`cmdline-error`], *page 94*)) [Reader Method]

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

Target Slot

[`item`], page 95.

`item-separator` (*object*) [Generic Reader]

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

Methods

`item-separator` ((*face* [`face`], *page 106*)) [Reader Method]

The face item separator.

Source [`face.lisp`], page 22.

Target Slot

[`item-separator`], page 108.

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

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

Methods

`items` ((*container* [`container`], *page 103*)) [Reader Method]

The items in the container.

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

Target Slot

[`items`], page 103.

`junk` (*condition*) [Generic Reader]

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

Methods

`junk` ((*condition* [`cmdline-junk-error`], *page 95*)) [Reader Method]

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

Target Slot

[`item`], page 95.

`left-padding` (*object*) [Generic Reader]

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

Methods

`left-padding` ((*face* [`face`], *page 106*)) [Reader Method]

The face left padding.
This property can take the following forms:
- <NUMBER>: the padding is relative to the enclosing face,
- SELF: the padding is set to wherever the face happens to be opened, - (<NUMBER> ABSOLUTE): the padding is set in absolute value,
- (<NUMBER> :RELATIVE-TO <FACE-NAME>): the padding is set relatively to a parent face named FACE-NAME.

Source [`face.lisp`], page 22.

Target Slot

[`left-padding`], page 107.

`line-width` (*object*) [Generic Reader]

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

Methods

`line-width` ((*context* [`context`], *page 104*)) [Reader Method]

The line width for help display.

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

Target Slot

[`line-width`], page 105.

`line-width` ((*sheet* [`sheet`], *page 114*)) [Reader Method]

The sheet's line width.

Source [`sheet.lisp`], page 23.

Target Slot

[`line-width`], page 115.

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

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

Methods

`long-name` ((*option* [`option`], *page 112*)) [Reader Method]
The option's long name.

Source [`option.lisp`], page 15.

Target Slot
[`long-name`], page 113.

`make-face-tree` (*definition* &optional *face-class*) [Generic Function]
Make a FACE-CLASS face tree from DEFINITION.

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

Source [`face.lisp`], page 22.

Methods

`make-face-tree` ((*definition* `list`) &optional *face-class*) [Method]
Make a FACE-CLASS face tree from a list of face name and initargs.

`make-face-tree` ((*name* `symbol`) &optional *face-class*) [Method]
Create a face named NAME.

`mapoptions` (*func there*) [Generic Function]
Map FUNC over all options in THERE.

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

Source [`synopsis.lisp`], page 21.

Methods

`mapoptions` (*func* (*context* [`context`], *page 104*)) [Method]
Map FUNC over all options in CONTEXT synopsis.

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

`mapoptions` (*func elsewhere*) [Method]
Do nothing by default.

`mapoptions` *:after* (*func* (*item* [`item`], *page 111*)) [Method]
Mark TRAVERSABLE as traversed.

`mapoptions` (*func* (*container* [`container`], *page 103*)) [Method]
Map FUNC over all containers or options in CONTAINER.

`mapoptions` (*func* (*option* [`option`], *page 112*)) [Method]
Call FUNC on OPTION.

`name` (*condition*) [Generic Reader]

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

Methods

`name` ((*condition* [`unknown-cmdline-option-error`], *page 100*)) [Reader Method]

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

Target Slot
[`item`], page 100.

`name` ((*face* [`face`], *page 106*)) [Reader Method]
The face name.

Source [`face.lisp`], page 22.

Target Slot
[`name`], page 107.

`name` ((*condition* [`cmdline-option-error`], *page 95*)) [Reader Method]

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

Target Slot
[`item`], page 95.

`negated-call` (*condition*) [Generic Reader]

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

Methods

`negated-call` ((*condition* [`unrecognized-negated-call-error`], *page 100*)) [Reader Method]

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

Target Slot
[`item`], page 100.

`negated-pack` (*object*) [Generic Function]

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

Methods

`negated-pack` ((*context* [`context`], *page 104*)) [Method]
Return the negated pack of CONTEXT's synopsis.

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

`negated-pack` ((*synopsis* [`synopsis`], *page 118*)) [Reader Method]
The negated pack string.

Source [`synopsis.lisp`], page 21.

Target Slot
[`negated-pack`], page 118.

`negated-pack-char` (*option* &optional *as-string*) [Generic Function]
Return OPTION's negated pack character, if any. If AS-STRING, return a string of that character.

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

Source [`option.lisp`], page 15.

Methods

`negated-pack-char` ((*negatable* *[*`negatable`*], page 112*) &optional *as-string*) [Method]
Return NEGATABLE's negated pack character, if any.

Source [`negatable.lisp`], page 16.

`negated-pack-char` ((*option* *[*`option`*], page 112*) &optional *as-string*) [Method]
Return nil (only the switch hierarchy is negated-pack'able).

`no-values` (*object*) [Generic Reader]
`(setf no-values)` (*object*) [Generic Writer]

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

Methods

`no-values` ((*switch-base* *[*`switch-base`*], page 116*)) [Reader Method]
`(setf no-values)` ((*switch-base* *[*`switch-base`*], page 116*)) [Writer Method]
The possible 'no' values.

Source [`switch-base.lisp`], page 17.

Target Slot
[`no-values`], page 117.

`open-frame` (*sheet frame*) [Generic Function]
Open FRAME on SHEET.

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

Source [`sheet.lisp`], page 23.

Method Combination
`progn`.

Options `:most-specific-last`

Methods

`open-frame` *progn* (*sheet* (*frame* *[*`frame`*], page 101*)) [Method]
Reach the frame's left margin.

`open-frame` *progn* (*sheet* (*frame* *[*`highlight-frame`*], page 102*)) [Method]
Reach the frame's left margin and output its highlight properties.

`option` (*condition*) [Generic Reader]

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

Methods

`option` ((*condition* *[*`option-error`*], page 99*)) [Reader Method]

Source [`option.lisp`], page 15.

Target Slot
[`option`], page 99.

`option-sticky-distance` (*option namearg*) [Generic Function]
Try to match OPTION's short name with a sticky argument against NAMEARG. If OPTION matches, return the length of OPTION's short name; otherwise 0.

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

Source [`option.lisp`], page 15.

Methods

`option-sticky-distance` ((*option* [`valued-option`], *page 119*) *namearg*) [Method]
Try to match OPTION's short name with a sticky argument against NAMEARG. If OPTION matches, return its short name's length; otherwise 0.

Source [`valued.lisp`], page 15.

`option-sticky-distance` ((*option* [`option`], *page 112*) *namearg*) [Method]
Return 0 (non-valued options don't take any argument, sticky or not).

`output-stream` (*object*) [Generic Reader]

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

Methods

`output-stream` ((*sheet* [`sheet`], *page 114*)) [Reader Method]
The sheet's output stream.

Source [`sheet.lisp`], page 23.

Target Slot
[`output-stream`], page 115.

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

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

Methods

`parent` ((*face* [`face`], *page 106*)) [Reader Method]
The face parent.

Source [`face.lisp`], page 22.

Target Slot
[`parent`], page 110.

`path-type` (*object*) [Generic Reader]

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

Methods

`path-type` ((*path* [`path`], *page 113*)) [Reader Method]
The path type.

Source [`path.lisp`], page 18.

Target Slot
[`path-type`], page 114.

`postfix` (*object*) [Generic Function]

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

Methods

`postfix` ((*context [`context`], page 104*)) [Method]
Return the postfix of CONTEXT's synopsis.

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

`postfix` ((*synopsis [`synopsis`], page 118*)) [Reader Method]
A postfix to the program synopsis.

Source [`synopsis.lisp`], page 21.

Target Slot
[`postfix`], page 118.

`potential-pack` (*object*) [Generic Reader]

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

Methods

`potential-pack` ((*synopsis [`synopsis`], page 118*)) [Reader Method]
The potential pack string.

Source [`synopsis.lisp`], page 21.

Target Slot
[`potential-pack`], page 118.

`potential-pack-p` (*pack there*) [Generic Function]
Return t if PACK is a potential pack in THERE.

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

Source [`synopsis.lisp`], page 21.

Methods

`potential-pack-p` (*pack* (*context [`context`], page 104*)) [Method]
Return t if PACK (a string) is a potential pack in CONTEXT.

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

`potential-pack-p` (*pack* (*synopsis [`synopsis`], page 118*)) [Method]
Return t if PACK is a potential pack for SYNOPSIS.

`print-help-spec` (*sheet help-spec*) [Generic Function]
Print HELP-SPEC on SHEET.

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

Source [`sheet.lisp`], page 23.

Methods

`print-help-spec` *:before* (*sheet help-spec*) [Method]

`print-help-spec` (*sheet* (*char* `character`)) [Method]
Print CHAR on SHEET with the current face.

`print-help-spec` (*sheet* (*char-vector* `simple-vector`)) [Method]
Print CHAR-VECTOR on SHEET with the current face.

`print-help-spec` (*sheet* (*string* `string`)) [Method]
Print STRING on SHEET with the current face.

`print-help-spec` (*sheet* (*help-spec* `list`)) [Method]
Open HELP-SPEC's face and print all of its items with it.

`retrieve-from-environment` (*option env-val*) [Generic Function]
Retrieve OPTION's value from the environment.
ENV-VAL is the value stored in the associated environment variable.

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

Source [`environ.lisp`], page 21.

Methods

`retrieve-from-environment` *:before* (*option env-val*) [Method]
Assert that ENV-VAL is not null.

`retrieve-from-environment` ((*flag* [`flag`], *page 110*) *env-val*) [Method]

`retrieve-from-environment` ((*option* [`valued-option`], *page 119*) *env-val*) [Method]

`retrieve-from-long-call` (*option cmdline-name* &optional *cmdline-argument cmdline*) [Generic Function]
Retrieve OPTION's value from a long call.
CMDLINE-NAME is the name used on the command-line.
CMDLINE-ARGUMENT is a potentially already parsed cmdline argument. Otherwise, CMDLINE is where to find an argument.
This function returns three values:
- the retrieved value,
- the value source,
- the new command-line (possibly with the first item popped if the option requires an argument).

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

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

Methods

`retrieve-from-long-call` ((*option* [`option`], *page 112*) *cmdline-name* &optional *cmdline-argument cmdline*) [Method]

`retrieve-from-long-call` ((*option* [`valued-option`], *page 119*) *cmdline-name* &optional *cmdline-argument cmdline*) [Method]

`retrieve-from-negated-call` (*option*) [Generic Function]
Retrieve OPTION's value from a negated call.

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

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

Methods

`retrieve-from-negated-call` ((*option* [`option`], *page 112*)) [Method]

`retrieve-from-negated-call` ((*option* [`valued-option`], *page 119*)) [Method]

`retrieve-from-negated-call` ((*negatable* *[*`negatable`*], page 112*)) [Method]

`retrieve-from-short-call` (*option* &optional *cmdline-argument cmdline*) [Generic Function]

Retrieve OPTION's value from a short call.
CMDLINE-ARGUMENT is a potentially already parsed cmdline argument. Otherwise, CMDLINE is where to find an argument.
This function returns three values:
- the retrieved value,
- the value source,
- the new command-line (possibly with the first item popped if the option requires an argument).

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

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

Methods

`retrieve-from-short-call` ((*option* *[*`option`*], page 112*) &optional *cmdline-argument cmdline*) [Method]

`retrieve-from-short-call` ((*option* *[*`valued-option`*], page 119*) &optional *cmdline-argument cmdline*) [Method]

`right-padding` (*object*) [Generic Reader]

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

Methods

`right-padding` ((*face* *[*`face`*], page 106*)) [Reader Method]

The face right padding.
This property can take the following forms:
- <NUMBER>: the padding is relative to the enclosing face,
- SELF: the padding is set to wherever the face happens to be closed, - (<NUMBER> ABSOLUTE): the padding is set in absolute value,
- (<NUMBER> :RELATIVE-TO <FACE-NAME>): the padding is set relatively to a parent face named FACE-NAME.

Source [`face.lisp`], page 22.

Target Slot
[`right-padding`], page 107.

`search-path` (*object*) [Generic Reader]

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

Methods

`search-path` ((*context* *[*`context`*], page 104*)) [Reader Method]

The search path for Clon files.

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

Target Slot
[`search-path`], page 105.

`sface-tree` (*object*) [Generic Reader]

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

Methods

`sface-tree` ((*sheet* [`sheet`], *page 114*)) [Reader Method]
The sheet's sface tree.

Source [`sheet.lisp`], page 23.

Target Slot
[`sface-tree`], page 115.

`short-call` (*condition*) [Generic Reader]

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

Methods

`short-call` ((*condition* [`unrecognized-short-call-error`], *page 101*)) [Reader Method]

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

Target Slot
[`item`], page 101.

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

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

Methods

`short-name` ((*option* [`option`], *page 112*)) [Reader Method]
The option's short name.

Source [`option.lisp`], page 15.

Target Slot
[`short-name`], page 113.

`short-pack` (*object*) [Generic Function]

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

Methods

`short-pack` ((*context* [`context`], *page 104*)) [Method]
Return the short pack of CONTEXT's synopsis.

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

`short-pack` ((*synopsis* [`synopsis`], *page 118*)) [Reader Method]
The short pack string.

Source [`synopsis.lisp`], page 21.

Target Slot
[`short-pack`], page 118.

`short-pack-char` (*option* &optional *as-string*) [Generic Function]
Return OPTION's short pack character, if any. If AS-STRING, return a string of that character.

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

Source [`option.lisp`], page 15.

Methods

`short-pack-char` ((*option* [`valued-option`], *page 119*) &optional *as-string*) [Method]
Return OPTION's short pack character if OPTION's argument is optional.

Source [`valued.lisp`], page 15.

`short-pack-char` ((*option* [`option`], *page 112*) &optional *as-string*) [Method]
Return OPTION's potential pack character.

`short-syntax-help-spec-prefix` (*option*) [Generic Function]
Return the help specification prefix for OPTION's short call.

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

Source [`valued.lisp`], page 15.

Methods

`short-syntax-help-spec-prefix` ((*option* [`negatable`], *page 112*)) [Method]

Source [`negatable.lisp`], page 16.

`short-syntax-help-spec-prefix` ((*option* [`valued-option`], *page 119*)) [Method]

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

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

Methods

`sibling` ((*sface* [`sface`], *page 114*)) [Reader Method]
The SFace's raw sibling.

Source [`sheet.lisp`], page 23.

Target Slot
[`sibling`], page 114.

`stream-ioctl-output-handle` (*stream*) [Generic Function]
Return STREAM's ioctl output handle or NIL.

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

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

Methods

`stream-ioctl-output-handle` ((*stream* `synonym-stream`)) [Method]

`stream-ioctl-output-handle` ((*stream* `two-way-stream`)) [Method]

`stream-ioctl-output-handle` ((*stream* `fd-stream`)) [Method]

`stream-ioctl-output-handle` (*stream*) [Method]

`stringify` (*valued-option value*) [Generic Function]
Transform VALUED-OPTION's VALUE into an argument. This is the opposite of argument conversion.

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

Source [`valued.lisp`], page 15.

Methods

`stringify` ((*xswitch [`xswitch`], page 121*) *value*) [Method]
Transform XSWITCH's VALUE into an argument.

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

`stringify` ((*enum [`enum`], page 105*) *value*) [Method]
Transform ENUM's VALUE into an argument.

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

`stringify` ((*path [`path`], page 113*) *value*) [Method]
Transform PATH's VALUE into an argument.

Source [`path.lisp`], page 18.

`stringify` ((*lispobj [`lispobj`], page 111*) *value*) [Method]
Transform LISPOBJ's VALUE into an argument.

Source [`lispobj.lisp`], page 18.

`stringify` ((*stropt [`stropt`], page 116*) *value*) [Method]
Transform STROPT's VALUE into an argument.

Source [`stropt.lisp`], page 17.

`stringify` ((*switch [`switch`], page 116*) *value*) [Method]
Transform SWITCH's VALUE into an argument.

Source [`switch.lisp`], page 17.

`subface` (*face name(s)*) [Generic Function]
Return subface of FACE named NAME(S) or nil.
If a list of names is provided instead of a single one, follow a subface branch matching those names to find the leaf face.

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

Source [`face.lisp`], page 22.

Methods

`subface` (*face* (*name* `symbol`)) [Method]
Return FACE'subface named NAME, or nil.

`subface` (*face* (*names* `list`)) [Method]
Return the leaf face from FACE'subbranch matching NAMES, or nil.

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

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

Methods

`subfaces` ((*face [`face`], page 106*)) [Reader Method]
The face children.

Source [`face.lisp`], page 22.

Target Slot
[`subfaces`], page 110.

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

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

Methods

`synopsis` ((*context [`context`], page 104*)) [Reader Method]
The program synopsis.

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

Target Slot
[`synopsis`], page 104.

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

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

Methods

`theme` ((*context [`context`], page 104*)) [Reader Method]
The theme filename.

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

Target Slot
[`theme`], page 105.

`top-padding` (*object*) [Generic Function]

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

Methods

`top-padding` ((*help-spec* `list`)) [Method]

Source [`sheet.lisp`], page 23.

`top-padding` (*other*) [Method]

Source [`sheet.lisp`], page 23.

`top-padding` ((*face [`face`], page 106*)) [Reader Method]
The face top padding.
This property can take the following forms:
- nil: the output can start right away,
- 0: the output should start on the next line,
- N>0: there should be N empty lines before the output.

Source [`face.lisp`], page 22.

Target Slot
[`top-padding`], page 108.

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

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

Methods

`traversedp` ((*item [`item`], page 111*)) [Reader Method]
`(setf traversedp)` ((*item [`item`], page 111*)) [Writer Method]
The item's traversal state.

Source [`item.lisp`], page 14.

Target Slot
[`traversedp`], page 111.

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

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

Methods

`typespec` ((*lispobj* [`lispobj`], *page 111*)) [Reader Method]
A type specifier the option's value should satisfy.

Source [`lispobj.lisp`], page 18.

Target Slot
[`typespec`], page 112.

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

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

Methods

`underline` ((*face* [`face`], *page 106*)) [Reader Method]
The face's underline level.

Source [`face.lisp`], page 22.

Target Slot
[`underline`], page 109.

`untraverse` (*item*) [Generic Function]
Reset ITEM's traversal state, and return ITEM.

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

Source [`item.lisp`], page 14.

Methods

`untraverse` ((*context* [`context`], *page 104*)) [Method]
Untraverse CONTEXT synopsis.

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

`untraverse` ((*container* [`container`], *page 103*)) [Method]
Untraverse all CONTAINER items.

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

`untraverse` ((*option* [`option`], *page 112*)) [Method]
OPTION is a terminal object: just return it.

Source [`option.lisp`], page 15.

`untraverse` ((*text* [`text`], *page 119*)) [Method]
TEXT is a terminal object: just return it.

Source [`text.lisp`], page 14.

`untraverse` *:after* ((*item* [`item`], *page 111*)) [Method]
Mark ITEM as untraversed.

`value` (*condition*) [Generic Reader]

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

Methods

`value` ((*condition [*`invalid-value`*], page 98*)) [Reader Method]

Source [`valued.lisp`], page 15.

Target Slot
[`value`], page 98.

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

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

Methods

`visiblep` ((*face [*`face`*], page 106*)) [Reader Method]
Whether the face is visible.

Source [`face.lisp`], page 22.

Target Slot
[`visiblep`], page 107.

`yes-values` (*object*) [Generic Reader]
`(setf yes-values)` (*object*) [Generic Writer]

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

Methods

`yes-values` ((*switch-base [*`switch-base`*], page 116*)) [Reader Method]
`(setf yes-values)` ((*switch-base [*`switch-base`*], page 116*)) [Writer Method]
The possible 'yes' values.

Source [`switch-base.lisp`], page 17.

Target Slot
[`yes-values`], page 117.

6.2.5 Conditions

`cmdline-error` [Condition]
An error related to a command-line item.

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

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

Direct superclasses
`error`.

Direct subclasses
- [`cmdline-junk-error`], page 95.
- [`cmdline-option-error`], page 95.
- [`invalid-negated-equal-syntax`], page 98.
- [`invalid-short-equal-syntax`], page 98.
- [`unknown-cmdline-option-error`], page 100.
- [`unrecognized-negated-call-error`], page 100.
- [`unrecognized-short-call-error`], page 101.

Direct methods
[`item`], page 80.

Direct slots

`item` [Slot]
The concerned command-line item.

Initargs `:item`

Readers [`item`], page 80.

Writers *This slot is read-only.*

`cmdline-junk-error` [Condition]
An error related to a command-line piece of junk.

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

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

Direct superclasses
[`cmdline-error`], page 94.

Direct methods
[`junk`], page 81.

Direct slots

`item` [Slot]
The piece of junk appearing on the command-line.

Initargs `:junk`, `:item`

Readers [`junk`], page 81.

Writers *This slot is read-only.*

`cmdline-option-error` [Condition]
An error related to a command-line (known) option.

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

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

Direct superclasses
- [`cmdline-error`], page 94.
- [`option-error`], page 99.

Direct subclasses
- [`invalid-cmdline-argument`], page 97.
- [`invalid-negated-syntax`], page 98.
- [`missing-cmdline-argument`], page 99.
- [`spurious-cmdline-argument`], page 99.

Direct methods
[`name`], page 83.

Direct slots

`item` [Slot]
The option's name as it appears on the command-line.

Initargs `:name`, `:item`

Readers [`name`], page 83.

Writers *This slot is read-only.*

`environment-error` [Condition]

An error related to an environment variable.

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

Source [`environ.lisp`], page 21.

Direct superclasses

`error`.

Direct subclasses

[`environmental-option-error`], page 96.

Direct methods

[`env-var`], page 76.

Direct slots

`env-var` [Slot]

The concerned environment variable.

Initargs `:env-var`

Readers [`env-var`], page 76.

Writers *This slot is read-only.*

`environmental-option-error` [Condition]

An error related to an option's environment variable.

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

Source [`environ.lisp`], page 21.

Direct superclasses

- [`environment-error`], page 96.
- [`option-error`], page 99.

Direct subclasses

[`invalid-environment-value`], page 97.

`home-directory` [Condition]

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

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

Direct superclasses

`warning`.

Direct methods

- [`error-string`], page 77.
- [`(setf error-string)`], page 77.

Direct slots

`error-string` [Slot]

Initargs `:error-string`

Readers [`error-string`], page 77.

Writers [`(setf error-string)`], page 77.

`invalid-argument` [Condition]
An invalid argument error.

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

Source [`valued.lisp`], page 15.

Direct superclasses
[`option-error`], page 99.

Direct subclasses
- [`invalid-cmdline-argument`], page 97.
- [`invalid-environment-value`], page 97.

Direct methods
- [`argument`], page 69.
- [`comment`], page 73.

Direct slots

`argument` [Slot]
The invalid argument.

Initargs `:argument`

Readers [`argument`], page 69.

Writers *This slot is read-only.*

`comment` [Slot]
An additional comment about the error.

Initargs `:comment`

Readers [`comment`], page 73.

Writers *This slot is read-only.*

`invalid-cmdline-argument` [Condition]
An invalid command-line argument error.

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

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

Direct superclasses
- [`cmdline-option-error`], page 95.
- [`invalid-argument`], page 97.

`invalid-environment-value` [Condition]
An invalid environment variable's value error.

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

Source [`environ.lisp`], page 21.

Direct superclasses
- [`environmental-option-error`], page 96.
- [`invalid-argument`], page 97.

Direct methods
[`env-val`], page 76.

Direct slots

`argument` [Slot]
The invalid environment variable value.

Initargs `:env-val`, `:argument`

Readers [`env-val`], page 76.

Writers *This slot is read-only.*

`invalid-negated-equal-syntax` [Condition]
An error related to a negated-equal syntax.

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

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

Direct superclasses
[`cmdline-error`], page 94.

`invalid-negated-syntax` [Condition]
An invalid negated syntax error.

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

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

Direct superclasses
[`cmdline-option-error`], page 95.

`invalid-short-equal-syntax` [Condition]
An error related to a short-equal syntax.

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

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

Direct superclasses
[`cmdline-error`], page 94.

`invalid-value` [Condition]
An invalid value error.

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

Source [`valued.lisp`], page 15.

Direct superclasses
[`option-error`], page 99.

Direct methods
- [`comment`], page 73.
- [`value`], page 94.

Direct slots

`value` [Slot]
The invalid value.

Initargs `:value`

Readers [`value`], page 94.

Writers *This slot is read-only.*

`comment` [Slot]
An additional comment about the error.

Initargs `:comment`

Readers [`comment`], page 73.

Writers *This slot is read-only.*

`missing-cmdline-argument` [Condition]
A missing command-line argument error.

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

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

Direct superclasses
[`cmdline-option-error`], page 95.

`option-error` [Condition]
An error related to an option.

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

Source [`option.lisp`], page 15.

Direct superclasses
`error`.

Direct subclasses
- [`cmdline-option-error`], page 95.
- [`environmental-option-error`], page 96.
- [`invalid-argument`], page 97.
- [`invalid-value`], page 98.

Direct methods
[`option`], page 84.

Direct slots

`option` [Slot]
The concerned option.

Initargs `:option`

Readers [`option`], page 84.

Writers *This slot is read-only.*

`spurious-cmdline-argument` [Condition]
A spurious command-line argument error.

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

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

Direct superclasses
[`cmdline-option-error`], page 95.

Direct methods
[`argument`], page 69.

Direct slots

`argument` [Slot]
The spurious argument.

Initargs `:argument`

Readers [`argument`], page 69.

Writers *This slot is read-only.*

`unknown-cmdline-option-error` [Condition]
An error related to an unknown command-line option.

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

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

Direct superclasses
[`cmdline-error`], page 94.

Direct methods
- [`argument`], page 69.
- [`name`], page 83.

Direct slots

`item` [Slot]
The option's name as it appears on the command-line.

Initargs `:name`, `:item`

Readers [`name`], page 83.

Writers *This slot is read-only.*

`argument` [Slot]
The option's command-line argument.

Initargs `:argument`

Readers [`argument`], page 69.

Writers *This slot is read-only.*

`unrecognized-negated-call-error` [Condition]
An error related to an unrecognized negated call.

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

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

Direct superclasses
[`cmdline-error`], page 94.

Direct methods
[`negated-call`], page 83.

Direct slots

`item` [Slot]
The unrecognized negated call on the command-line.

Initargs `:negated-call`, `:item`

Readers [`negated-call`], page 83.

Writers *This slot is read-only.*

`unrecognized-short-call-error` [Condition]
An error related to an unrecognized short call.

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

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

Direct superclasses
[`cmdline-error`], page 94.

Direct methods
[`short-call`], page 89.

Direct slots

`item` [Slot]
The unrecognized short call on the command-line.

Initargs `:short-call`, `:item`

Readers [`short-call`], page 89.

Writers *This slot is read-only.*

6.2.6 Structures

`cmdline-option` [Structure]

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

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

Direct superclasses
`structure-object`.

Direct slots

`name` [Slot]

Readers [`cmdline-option-name`], page 53.

Writers [`(setf cmdline-option-name)`], page 53.

`option` [Slot]

Readers [`cmdline-option-option`], page 53.

Writers [`(setf cmdline-option-option)`], page 53.

`value` [Slot]

Readers [`cmdline-option-value`], page 54.

Writers [`(setf cmdline-option-value)`], page 54.

`source` [Slot]

Readers [`cmdline-option-source`], page 53.

Writers [`(setf cmdline-option-source)`], page 53.

`frame` [Structure]
The FRAME structure.
This structure hold layout properties used for printing.

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

Source [`sheet.lisp`], page 23.

Direct superclasses
`structure-object`.

Direct subclasses
[`highlight-frame`], page 102.

Direct methods
- [`close-frame`], page 73.
- [`open-frame`], page 84.

Direct slots

`sface` [Slot]
Readers [`frame-sface`], page 56.
Writers [`(setf frame-sface)`], page 56.

`left-margin` [Slot]
Readers [`frame-left-margin`], page 55.
Writers [`(setf frame-left-margin)`], page 55.

`right-margin` [Slot]
Readers [`frame-right-margin`], page 56.
Writers [`(setf frame-right-margin)`], page 56.

`highlight-frame` [Structure]
The HIGHLIGHT-FRAME structure.
This structure holds both layout and highlight properties used for printing.

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

Source [`sheet.lisp`], page 23.

Direct superclasses
[`frame`], page 101.

Direct methods
- [`close-frame`], page 73.
- [`open-frame`], page 84.

Direct slots

`highlight-property-instances` [Slot]
Readers [`highlight-frame-highlight-property-instances`], page 56.
Writers [`(setf highlight-frame-highlight-property-instances)`], page 56.

`highlight-property-instance` [Structure]
The HIGHLIGHT-PROPERTY-INSTANCE structure.

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

Source [`sheet.lisp`], page 23.

Direct superclasses
`structure-object`.

Direct slots

`name` [Slot]
Readers [`highlight-property-instance-name`], page 57.
Writers [`(setf highlight-property-instance-name)`], page 57.

`value` [Slot]

Readers [`highlight-property-instance-value`], page 57.

Writers [`(setf highlight-property-instance-value)`], page 57.

6.2.7 Classes

`abstract-class` [Class]

The ABSTRACT-CLASS class.
This is the meta-class for abstract classes.

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

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

Direct superclasses

`standard-class`.

Direct methods

- [`make-instance`], page 49.
- [`validate-superclass`], page 49.
- [`validate-superclass`], page 49.

`container` [Class]

The CONTAINER class.
This class is a mixin used in synopsis and groups to represent the program's command-line hierarchy.

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

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

Direct superclasses

[`item`], page 111.

Direct subclasses

- [`group`], page 110.
- [`synopsis`], page 118.

Direct methods

- [`check-name-clash`], page 72.
- [`check-name-clash`], page 72.
- [`check-name-clash`], page 72.
- [`help-spec`], page 78.
- [`initialize-instance`], page 47.
- [`initialize-instance`], page 47.
- [`items`], page 81.
- [`mapoptions`], page 82.
- [`untraverse`], page 93.

Direct slots

`items` [Slot]

The items in the container.

Type `list`

Initargs `:items`

Readers [items], page 81.

Writers *This slot is read-only.*

`context` [Class]

The CONTEXT class.
This class represents the associatiion of a synopsis and a set of command-line options based on it.

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

Source [context.lisp], page 25.

Direct methods

- [clon-options-group], page 72.
- [cmdline-options], page 73.
- [(setf cmdline-options)], page 73.
- [error-handler], page 76.
- [highlight], page 79.
- [initialize-instance], page 47.
- [line-width], page 81.
- [mapoptions], page 82.
- [negated-pack], page 83.
- [postfix], page 86.
- [potential-pack-p], page 86.
- [search-path], page 88.
- [short-pack], page 89.
- [synopsis], page 92.
- [theme], page 92.
- [untraverse], page 93.

Direct Default Initargs

Initarg	Value
:cmdline	(cmdline)

Direct slots

`synopsis` [Slot]

The program synopsis.

Type net.didierverna.clon::synopsis

Initform net.didierverna.clon:*synopsis*

Initargs :synopsis

Readers [synopsis], page 92.

Writers *This slot is read-only.*

`progname` [Slot]

The program name as it appears on the command-line.

Type string

`cmdline-options` [Slot]
The options from the command-line.

Type `list`

Readers [`cmdline-options`], page 73.

Writers [`(setf cmdline-options)`], page 73.

`remainder` [Slot]
The non-Clon part of the command-line.

Type `list`

`search-path` [Slot]
The search path for Clon files.

Readers [`search-path`], page 88.

Writers *This slot is read-only.*

`theme` [Slot]
The theme filename.

Readers [`theme`], page 92.

Writers *This slot is read-only.*

`line-width` [Slot]
The line width for help display.

Readers [`line-width`], page 81.

Writers *This slot is read-only.*

`highlight` [Slot]
Clon's output highlight mode.

Readers [`highlight`], page 79.

Writers *This slot is read-only.*

`error-handler` [Slot]
The behavior to adopt on option retrieval errors.

Type `symbol`

Initform `:quit`

Readers [`error-handler`], page 76.

Writers *This slot is read-only.*

`enum` [Class]
The ENUM class.
This class implements options whose values belong to a set of keywords.

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

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

Direct superclasses

- [`enum-base`], page 106.
- [`valued-option`], page 119.

Direct methods

- [check], page 71.
- [convert], page 74.
- [stringify], page 91.

Direct slots

`argument-name` [Slot]

Initform `"type"`

`enum-base` [Class]

The ENUM-BASE abstract class.
This class provides support for options including enumerated values.

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

Source [enum-base.lisp], page 19.

Direct subclasses

- [enum], page 105.
- [xswitch], page 121.

Direct methods

- [enum], page 76.
- [initialize-instance], page 47.

Direct slots

`enum` [Slot]

The set of possible values.

Initargs `:enum`

Readers [enum], page 76.

Writers *This slot is read-only.*

`face` [Class]

The FACE class.

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

Source [face.lisp], page 22.

Direct subclasses

[sface], page 114.

Direct methods

- [background], page 70.
- [blink], page 70.
- [bottom-padding], page 71.
- [concealedp], page 74.
- [crossed-out-p], page 75.
- [foreground], page 77.
- [framedp], page 77.
- [initialize-instance], page 48.
- [initialize-instance], page 48.
- [initialize-instance], page 48.

- [intensity], page 80.
- [inversep], page 80.
- [italicp], page 80.
- [item-separator], page 80.
- [left-padding], page 81.
- [name], page 83.
- [parent], page 85.
- [right-padding], page 88.
- [slot-unbound], page 49.
- [subfaces], page 91.
- [top-padding], page 92.
- [underline], page 93.
- [visiblep], page 94.

Direct slots

`name` [Slot]

The face name.

Initargs `:name`

Readers [name], page 83.

Writers *This slot is read-only.*

`visiblep` [Slot]

Whether the face is visible.

Initform `t`

Initargs `:visible`

Readers [visiblep], page 94.

Writers *This slot is read-only.*

`left-padding` [Slot]

The face left padding.
This property can take the following forms:
- <NUMBER>: the padding is relative to the enclosing face,
- SELF: the padding is set to wherever the face happens to be opened, - (<NUMBER> ABSOLUTE): the padding is set in absolute value,
- (<NUMBER> :RELATIVE-TO <FACE-NAME>): the padding is set relatively to a parent face named FACE-NAME.

Initform `0`

Initargs `:padding-left`

Readers [left-padding], page 81.

Writers *This slot is read-only.*

`right-padding` [Slot]

The face right padding.
This property can take the following forms:
- <NUMBER>: the padding is relative to the enclosing face,

- SELF: the padding is set to wherever the face happens to be closed, - (<NUMBER> ABSOLUTE): the padding is set in absolute value,
- (<NUMBER> :RELATIVE-TO <FACE-NAME>): the padding is set relatively to a parent face named FACE-NAME.

Initform `(quote net.didierverna.clon::self)`

Initargs `:padding-right`

Readers [`right-padding`], page 88.

Writers *This slot is read-only.*

`top-padding` [Slot]

The face top padding.
This property can take the following forms:
- nil: the output can start right away,
- 0: the output should start on the next line,
- N>0: there should be N empty lines before the output.

Initargs `:padding-top`

Readers [`top-padding`], page 92.

Writers *This slot is read-only.*

`bottom-padding` [Slot]

The face bottom padding.
This property can take the following forms:
- nil: the next output can start right at the end of this face's, - 0: the next output should start on the next line,
- N>0: there should be N empty lines before the next output.

Initargs `:padding-bottom`

Readers [`bottom-padding`], page 71.

Writers *This slot is read-only.*

`item-separator` [Slot]

The face item separator.

Initform `#\`

Initargs `:item-separator`

Readers [`item-separator`], page 80.

Writers *This slot is read-only.*

`intensity` [Slot]

The face intensity.

Initargs `:intensity`

Readers [`intensity`], page 80.

Writers *This slot is read-only.*

`italicp` [Slot]

The face's italic status.

Initargs `:italic`

Readers [`italicp`], page 80.

Writers *This slot is read-only.*

`underline` [Slot]
The face's underline level.

Initargs `:underline`

Readers [`underline`], page 93.

Writers *This slot is read-only.*

`blink` [Slot]
The face's blink speed.

Initargs `:blink`

Readers [`blink`], page 70.

Writers *This slot is read-only.*

`inversep` [Slot]
The face's inverse video status.

Initargs `:inverse`

Readers [`inversep`], page 80.

Writers *This slot is read-only.*

`concealedp` [Slot]
The face's concealed status.

Initargs `:concealed`

Readers [`concealedp`], page 74.

Writers *This slot is read-only.*

`crossed-out-p` [Slot]
The face's crossed out status.

Initargs `:crossed-out`

Readers [`crossed-out-p`], page 75.

Writers *This slot is read-only.*

`framedp` [Slot]
The face's framed status.

Initargs `:framed`

Readers [`framedp`], page 77.

Writers *This slot is read-only.*

`foreground` [Slot]
The face foreground.

Initargs `:foreground`

Readers [`foreground`], page 77.

Writers *This slot is read-only.*

`background` [Slot]
The face background.

Initargs `:background`

Readers [`background`], page 70.

Writers *This slot is read-only.*

`subfaces` [Slot]
The face children.

Initargs `:subfaces`

Readers [`subfaces`], page 91.

Writers *This slot is read-only.*

`parent` [Slot]
The face parent.

Readers [`parent`], page 85.

Writers *This slot is read-only.*

`flag` [Class]
The FLAG class.
This class implements options that don't take any argument.

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

Source [`flag.lisp`], page 15.

Direct superclasses
[`option`], page 112.

Direct methods
[`retrieve-from-environment`], page 87.

`group` [Class]
The GROUP class.
This class groups other groups, options or strings together, effectively implementing hierarchical program command-line.

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

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

Direct superclasses
[`container`], page 103.

Direct methods
- [`header`], page 78.
- [`help-spec`], page 78.

Direct slots

`header` [Slot]
The group's header.

Initargs `:header`

Readers [`header`], page 78.

Writers *This slot is read-only.*

`item` [Class]

The ITEM class.
This class is the base class for all synopsis items.

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

Source [`item.lisp`], page 14.

Direct subclasses

- [`container`], page 103.
- [`option`], page 112.
- [`text`], page 119.

Direct methods

- [`help-spec`], page 79.
- [`hiddenp`], page 79.
- [`mapoptions`], page 82.
- [`traversedp`], page 92.
- [`(setf traversedp)`], page 92.
- [`untraverse`], page 93.

Direct slots

`traversedp` [Slot]

The item's traversal state.

Readers [`traversedp`], page 92.

Writers [`(setf traversedp)`], page 92.

`hiddenp` [Slot]

Whether the item is hidden in help strings.

Initargs `:hidden`

Readers [`hiddenp`], page 79.

Writers *This slot is read-only.*

`lispobj` [Class]

The LISPOBJ class.
This class implements read-from-string options.

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

Source [`lispobj.lisp`], page 18.

Direct superclasses

[`valued-option`], page 119.

Direct methods

- [`check`], page 71.
- [`convert`], page 74.
- [`stringify`], page 91.
- [`typespec`], page 93.

Direct slots

`argument-name` [Slot]

Initform `"obj"`

`typespec` [Slot]
A type specifier the option's value should satisfy.

Initform `t`

Initargs `:typespec`

Readers [`typespec`], page 93.

Writers *This slot is read-only.*

`negatable` [Class]
The NEGATABLE Class.
This class implements the negated syntax for the switch-based hierarchy.

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

Source [`negatable.lisp`], page 16.

Direct subclasses
[`switch-base`], page 116.

Direct methods
- [`negated-pack-char`], page 84.
- [`retrieve-from-negated-call`], page 88.
- [`short-syntax-help-spec-prefix`], page 90.

`option` [Class]
The OPTION class.
This is the base class for all options.

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

Source [`option.lisp`], page 15.

Direct superclasses
[`item`], page 111.

Direct subclasses
- [`flag`], page 110.
- [`valued-option`], page 119.

Direct methods
- [`check-name-clash`], page 72.
- [`description`], page 75.
- [`env-var`], page 76.
- [`help-spec`], page 78.
- [`initialize-instance`], page 48.
- [`initialize-instance`], page 48.
- [`long-name`], page 82.
- [`mapoptions`], page 82.
- [`negated-pack-char`], page 84.
- [`option-sticky-distance`], page 85.
- [`retrieve-from-long-call`], page 87.
- [`retrieve-from-negated-call`], page 87.
- [`retrieve-from-short-call`], page 88.

- [short-name], page 89.
- [short-pack-char], page 90.
- [untraverse], page 93.

Direct Default Initargs

Initarg	Value
:internal	nil

Direct slots

short-name [Slot]
The option's short name.

Type (or null string)

Initargs :short-name

Readers [short-name], page 89.

Writers *This slot is read-only.*

long-name [Slot]
The option's long name.

Type (or null string)

Initargs :long-name

Readers [long-name], page 82.

Writers *This slot is read-only.*

description [Slot]
The option's description.

Type (or null string)

Initargs :description

Readers [description], page 75.

Writers *This slot is read-only.*

env-var [Slot]
The option's associated environment variable.

Type (or null string)

Initargs :env-var

Readers [env-var], page 76.

Writers *This slot is read-only.*

path [Class]
The PATH class.
This class implements options whose values are (colon-separated lists of) pathnames.

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

Source [path.lisp], page 18.

Direct superclasses

[valued-option], page 119.

Direct methods

- [check], page 71.
- [convert], page 74.
- [path-type], page 85.
- [stringify], page 91.

Direct slots

`argument-name` [Slot]

Initform `"path"`

`path-type` [Slot]

The path type.

Initargs `:type`

Readers [path-type], page 85.

Writers *This slot is read-only.*

`sface` [Class]

The SFACE class.
An SFace is the association of a face and its raw sibling. The sibling is used to create subfaces which would be missing from the original, user defined one.

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

Source [sheet.lisp], page 23.

Direct superclasses

[face], page 106.

Direct methods

[sibling], page 90.

Direct slots

`sibling` [Slot]

The SFace's raw sibling.

Readers [sibling], page 90.

Writers *This slot is read-only.*

`sheet` [Class]

The SHEET class.
This class implements the notion of sheet for printing Clon help.

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

Source [sheet.lisp], page 23.

Direct methods

- [column], page 73.
- [(setf column)], page 73.
- [frames], page 77.
- [(setf frames)], page 77.
- [highlightp], page 79.
- [initialize-instance], page 47.
- [initialize-instance], page 47.

- [line-width], page 81.
- [output-stream], page 85.
- [sface-tree], page 89.

Direct slots

output-stream [Slot]
The sheet's output stream.

Type stream

Initargs :output-stream

Readers [output-stream], page 85.

Writers *This slot is read-only.*

line-width [Slot]
The sheet's line width.

Type (integer 1)

Initargs :line-width

Readers [line-width], page 81.

Writers *This slot is read-only.*

highlightp [Slot]
Whether to highlight SHEET's output.

Initargs :highlightp

Readers [highlightp], page 79.

Writers *This slot is read-only.*

sface-tree [Slot]
The sheet's sface tree.

Readers [sface-tree], page 89.

Writers *This slot is read-only.*

column [Slot]
The sheet's current column.

Type (integer 0)

Initform 0

Readers [column], page 73.

Writers [(setf column)], page 73.

frames [Slot]
The stack of currently open frames.

Type list

Readers [frames], page 77.

Writers [(setf frames)], page 77.

`stropt` [Class]

The STROPT class.
This class implements options the values of which are strings.

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

Source [`stropt.lisp`], page 17.

Direct superclasses
[`valued-option`], page 119.

Direct methods
- [`check`], page 71.
- [`convert`], page 74.
- [`stringify`], page 91.

Direct slots

`argument-name` [Slot]

Initform `"str"`

`switch` [Class]

The SWITCH class.
This class implements boolean options.

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

Source [`switch.lisp`], page 17.

Direct superclasses
- [`switch-base`], page 116.
- [`valued-option`], page 119.

Direct methods
- [`check`], page 71.
- [`convert`], page 74.
- [`initialize-instance`], page 48.
- [`stringify`], page 91.

`switch-base` [Class]

The SWITCH-BASE abstract class.
This class provides support for options including boolean values.

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

Source [`switch-base.lisp`], page 17.

Direct superclasses
[`negatable`], page 112.

Direct subclasses
- [`switch`], page 116.
- [`xswitch`], page 121.

Direct methods
- [`argument-style`], page 70.
- [`argument-styles`], page 70.
- [`(setf argument-styles)`], page 70.

- [initialize-instance], page 48.
- [initialize-instance], page 48.
- [no-values], page 84.
- [(setf no-values)], page 84.
- [yes-values], page 94.
- [(setf yes-values)], page 94.

Direct Default Initargs

Initarg	Value
:argument-type	optional
:argument-styles	(quote (yes/no on/off true/false yup/nope yeah/nah))
:yes-values	(quote (yes on true yup yeah))
:no-values	(quote (no off false nope nah))

Direct slots

argument-styles [Slot]

The possible argument styles.
The position of every argument style in the list must correspond to the position of the associated strings in the yes-values and no-values slots.

Type list

Initargs :argument-styles

Readers [argument-styles], page 70.

Writers [(setf argument-styles)], page 70.

yes-values [Slot]

The possible 'yes' values.

Type list

Initargs :yes-values

Readers [yes-values], page 94.

Writers [(setf yes-values)], page 94.

no-values [Slot]

The possible 'no' values.

Type list

Initargs :no-values

Readers [no-values], page 84.

Writers [(setf no-values)], page 84.

argument-style [Slot]

The selected argument style.

Type keyword

Initform :yes/no

Initargs :argument-style

Readers [argument-style], page 70.

Writers *This slot is read-only.*

`synopsis` [Class]

The SYNOPSIS class.
This class handles the description of the program's command-line options.

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

Source [`synopsis.lisp`], page 21.

Direct superclasses
[`container`], page 103.

Direct methods

- [`clon-options-group`], page 72.
- [`help-spec`], page 78.
- [`initialize-instance`], page 49.
- [`initialize-instance`], page 49.
- [`negated-pack`], page 83.
- [`postfix`], page 86.
- [`potential-pack`], page 86.
- [`potential-pack-p`], page 86.
- [`short-pack`], page 89.

Direct slots

`postfix` [Slot]

A postfix to the program synopsis.

Type `(or null string)`

Initargs `:postfix`

Readers [`postfix`], page 86.

Writers *This slot is read-only.*

`short-pack` [Slot]

The short pack string.

Type `(or null string)`

Readers [`short-pack`], page 89.

Writers *This slot is read-only.*

`negated-pack` [Slot]

The negated pack string.

Type `(or null string)`

Readers [`negated-pack`], page 83.

Writers *This slot is read-only.*

`potential-pack` [Slot]

The potential pack string.

Type `(or null string)`

Readers [`potential-pack`], page 86.

Writers *This slot is read-only.*

`clon-options-group` [Slot]
The Clon options group.

Type `net.didierverna.clon::group`

Initargs `:clon-options-group`

Readers [`clon-options-group`], page 72.

Writers *This slot is read-only.*

`text` [Class]
The TEXT class.
This class implements plain text objects appearing in a synopsis.

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

Source [`text.lisp`], page 14.

Direct superclasses
[`item`], page 111.

Direct methods
- [`check-name-clash`], page 72.
- [`check-name-clash`], page 72.
- [`contents`], page 74.
- [`help-spec`], page 79.
- [`untraverse`], page 93.

Direct slots

`contents` [Slot]
The actual text string.

Type `string`

Initargs `:contents`

Readers [`contents`], page 74.

Writers *This slot is read-only.*

`valued-option` [Class]
The VALUED-OPTION class.
This is the base class for options accepting arguments.

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

Source [`valued.lisp`], page 15.

Direct superclasses
[`option`], page 112.

Direct subclasses
- [`enum`], page 105.
- [`lispobj`], page 111.
- [`path`], page 113.
- [`stropt`], page 116.
- [`switch`], page 116.
- [`xswitch`], page 121.

Direct methods

- [argument-name], page 69.
- [argument-required-p], page 69.
- [default-value], page 75.
- [fallback-value], page 77.
- [help-spec], page 78.
- [initialize-instance], page 48.
- [initialize-instance], page 48.
- [option-sticky-distance], page 85.
- [retrieve-from-environment], page 87.
- [retrieve-from-long-call], page 87.
- [retrieve-from-negated-call], page 87.
- [retrieve-from-short-call], page 88.
- [short-pack-char], page 90.
- [short-syntax-help-spec-prefix], page 90.

Direct Default Initargs

Initarg	Value
:argument-type	required

Direct slots

argument-name [Slot]

The option's argument display name.

Initform "arg"

Initargs :argument-name

Readers [argument-name], page 69.

Writers *This slot is read-only.*

argument-required-p [Slot]

Whether the option's argument is required.

Readers [argument-required-p], page 69.

Writers *This slot is read-only.*

fallback-value [Slot]

The option's fallback value.

Initargs :fallback-value

Readers [fallback-value], page 77.

Writers *This slot is read-only.*

default-value [Slot]

The option's default value.

Initargs :default-value

Readers [default-value], page 75.

Writers *This slot is read-only.*

`xswitch` [Class]

The XSWITCH class.
This class merges the functionalities of switches and enumerations. As such, the negated syntax is available for extended xswitches.

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

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

Direct superclasses

- [`enum-base`], page 106.
- [`switch-base`], page 116.
- [`valued-option`], page 119.

Direct methods

- [`check`], page 71.
- [`convert`], page 74.
- [`stringify`], page 91.

Direct slots

`enum` [Slot]

The set of possible non-boolean values.

Appendix A Indexes

A.1 Concepts

F

L

M

N

A.2 Functions

%

(

~

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

Y

A.3 Variables

*

A

B

C

D

E

F

H

I

L

N

O

P

R

S

T

U

V

Y

A.4 Data types

A

C

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

X