The exil Reference Manual

EXpert system In Lisp, version 0.01

Jakub Kalab <jakubkalab@gmail.com>

This manual was generated automatically by Declt 1.0 beta 15 "James T. Kirk" on Sat Dec 01 18:35:52 2012 GMT+1.

Table of Contents

1 System

Name exil

Author Jakub Kalab <jakubkalab@gmail.com>

Maintainer
Jakub Kalab <jakubkalab@gmail.com>

License BSD

Description
EXpert system In Lisp

Long Description
Version 0.1

Definition file
[exil.asd], page 3 (Lisp file)

Components

2 Files

Files are sorted by type and then listed depth-first from the system components tree.

2.1 Lisp

2.1.1 `exil.asd`

Location `exil.asd`

Packages [exil-system], page 15

2.1.2 `packages.lisp`

Parent [exil], page 1 (system)

Location `packages.lisp`

Packages

- [exil-utils], page 15
- [exil-core], page 16
- [exil-rete], page 19
- [exil-env], page 24
- [exil], page 26
- [exil-user], page 27

2.1.3 `utils.lisp`

Dependency
`packages`

Parent [exil], page 1 (system)

Location `utils.lisp`

Exported Definitions

- [alistp], page 31 (function)
- [assoc-key], page 31 (function)
- [assoc-value], page 32 (function)
- [(setf assoc-value)], page 32 (function)
- [class-slot-value], page 32 (function)
- [cpl-assoc-val], page 32 (function)
- [(setf cpl-assoc-val)], page 32 (function)
- [diff-delete], page 29 (macro)
- [doplist], page 29 (macro)
- [every-couple], page 32 (function)
- [exil-equal-p], page 37 (generic function)
- [exil-equal-p], page 37 (method)
- [exil-equal-p], page 37 (method)
- [exil-equal-p], page 37 (method)
- [exil-equal-p], page 37 (method)
- [exil-equal-p], page 37 (method)

2.1.4 templates.lisp

Dependency

`utils`

Parent [exil], page 1 (system)

Location `templates.lisp`

Exported Definitions

- [template], page 48 (class)
- [tmpl-name], page 43 (method)
- [variable-p], page 35 (function)

Internal Definitions

- [clips-tmpl-slot-spec-p], page 50 (function)
- [make-tmpl-obj-clips], page 51 (function)
- [make-tmpl-obj-nonclips], page 51 (function)
- [make-tmpl-object], page 51 (function)
- [template-object], page 76 (class)
- [tmpl-object-equal-p], page 66 (method)
- [tmpl-object-slot-value], page 66 (method)
- [(setf tmpl-object-slot-value)], page 66 (method)
- [tmpl-object-specification-p], page 53 (function)
- [tmpl-slot-spec-p], page 53 (function)

2.1.5 facts.lisp

Dependency

templates

Parent [exil], page 1 (system)

Location facts.lisp

Exported Definitions

- [atom-position], page 36 (method)
- [copy-fact], page 37 (method)
- [fact], page 38 (method)
- [fact], page 45 (class)
- [fact-description], page 38 (method)
- [fact-description], page 38 (method)
- [fact-equal-p], page 38 (generic function)
- [fact-equal-p], page 38 (method)
- [fact-equal-p], page 38 (method)
- [fact-equal-p], page 38 (method)
- [fact-slot], page 38 (generic function)
- [fact-slot], page 38 (method)
- [fact-slot], page 39 (method)
- [find-atom], page 39 (method)
- [make-fact], page 33 (function)
- [simple-fact], page 47 (class)
- [template-fact], page 48 (class)
- [tmpl-fact-slot-value], page 43 (method)
- [(setf tmpl-fact-slot-value)], page 43 (method)

Internal Definitions

- [make-tmpl-fact], page 51 (function)
- [tmpl-fact-specification-p], page 53 (function)

2.1.6 patterns.lisp

Dependency
facts

Parent [exil], page 1 (system)

Location patterns.lisp

Exported Definitions

- [atom-equal-p], page 36 (generic function)
- [atom-equal-p], page 36 (method)
- [atom-position], page 36 (method)
- [constant-test], page 32 (function)
- [find-atom], page 39 (method)
- [make-pattern], page 33 (function)
- [match-var], page 40 (method)
- [(setf match-var)], page 40 (method)
- [negated-p], page 40 (method)
- [(setf negated-p)], page 40 (method)
- [pattern], page 41 (method)
- [pattern], page 45 (class)
- [pattern-equal-p], page 41 (generic function)
- [pattern-equal-p], page 41 (method)
- [pattern-equal-p], page 41 (method)
- [pattern-equal-p], page 41 (method)
- [simple-pattern], page 47 (class)
- [template-pattern], page 48 (class)
- [var-or-equal-p], page 35 (function)

Internal Definitions

- [make-tmpl-pattern], page 51 (function)
- [tmpl-pattern-slot-value], page 66 (method)
- [tmpl-pattern-specification-p], page 53 (function)

2.1.7 rules.lisp

Dependency
patterns

Parent [exil], page 1 (system)

Location rules.lisp

Exported Definitions

- [activations], page 35 (method)
- [conditions], page 37 (method)
- [make-rule], page 39 (method)
- [name], page 40 (method)
- [rule], page 46 (class)
- [rule-equal-p], page 42 (method)

2.1.8 tokens.lisp

Dependency
rules

Parent [exil], page 1 (system)

Location tokens.lisp

Exported Definitions
- [token->list], page 43 (method)
- [token-equal-p], page 43 (generic function)
- [token-equal-p], page 43 (method)
- [token-equal-p], page 43 (method)
- [token-equal-p], page 44 (method)

Internal Definitions
- [empty-token], page 72 (class)
- [includes-p], page 60 (method)
- [includes-p], page 60 (method)
- [negative-wmes], page 62 (method)
- [(setf negative-wmes)], page 62 (method)
- [parent], page 63 (method)
- [previous-wme], page 64 (method)
- [token], page 66 (method)
- [token], page 78 (class)
- [wme], page 67 (method)

2.1.9 rete-generic-node.lisp

Dependency
tokens

Parent [exil], page 1 (system)

Location rete-generic-node.lisp

Internal Definitions
- [activate], page 53 (generic function)
- [activate-children], page 54 (generic function)
- [activate-children], page 54 (method)
- [add-child], page 54 (method)
- [add-children], page 54 (method)
- [add-item], page 54 (method)
- [children], page 56 (method)
- [(setf children)], page 56 (method)
- [described-object], page 72 (class)
- [description], page 57 (method)
- [(setf description)], page 57 (method)
- [inactivate], page 59 (generic function)
- [inactivate], page 60 (method)

- [inactivate-children], page 60 (method)
- [items], page 60 (method)
- [(setf items)], page 61 (method)
- [memory-node], page 74 (class)
- [node], page 74 (class)
- [node-equal-p], page 62 (generic function)
- [node-equal-p], page 62 (method)
- [node-equal-p], page 63 (method)

2.1.10 rete-alpha-part.lisp

Dependency

rete-generic-node

Parent [exil], page 1 (system)

Location rete-alpha-part.lisp

Internal Definitions

- [activate], page 53 (method)
- [activate], page 53 (method)
- [activate], page 54 (method)
- [activate], page 54 (method)
- [activate-children], page 54 (method)
- [activate-memory], page 54 (method)
- [alpha-memory-node], page 67 (class)
- [alpha-node], page 67 (class)
- [alpha-subtop-node], page 68 (class)
- [alpha-test-node], page 68 (class)
- [alpha-top-node], page 69 (class)
- [get-network], page 59 (method)
- [(setf get-network)], page 59 (method)
- [get/initialize-network], page 59 (method)
- [inactivate], page 60 (method)
- [inactivate], page 60 (method)
- [inactivate], page 60 (method)
- [initialize-network], page 60 (method)
- [memory], page 61 (method)
- [(setf memory)], page 61 (method)
- [networks], page 62 (method)
- [(setf networks)], page 62 (method)
- [node-equal-p], page 62 (method)
- [simple-fact-alpha-node], page 75 (class)
- [simple-fact-key-name], page 64 (method)
- [simple-fact-subtop-node], page 75 (class)
- [simple-fact-test-node], page 76 (class)
- [template-fact-alpha-node], page 76 (class)

- [template-fact-subtop-node], page 76 (class)
- [template-fact-test-node], page 76 (class)
- [test], page 65 (generic function)
- [test], page 65 (method)
- [test], page 65 (method)
- [test], page 65 (method)
- [tested-field], page 65 (method)
- [value], page 67 (method)

2.1.11 rete-beta-part.lisp

Dependency

rete-alpha-part

Parent [exil], page 1 (system)

Location rete-beta-part.lisp

Internal Definitions

- [activate], page 53 (method)
- [activate], page 53 (method)
- [activate], page 53 (method)
- [activate], page 53 (method)
- [activate], page 53 (method)
- [add-production], page 55 (method)
- [alpha-memory], page 55 (method)
- [beta-join-node], page 70 (class)
- [beta-memory], page 55 (method)
- [beta-memory-node], page 70 (class)
- [beta-negative-node], page 71 (class)
- [beta-node], page 71 (class)
- [beta-top-node], page 71 (class)
- [broken-match], page 55 (method)
- [complete-match], page 56 (method)
- [current-field], page 56 (method)
- [delete-production], page 57 (method)
- [get-bad-wmes], page 58 (method)
- [inactivate], page 59 (method)
- [inactivate], page 59 (method)
- [inactivate], page 59 (method)
- [make-test], page 51 (function)
- [node-equal-p], page 62 (method)
- [parent], page 63 (method)
- [(setf parent)], page 63 (method)
- [perform-join-test], page 63 (method)
- [perform-join-tests], page 63 (method)
- [previous-condition], page 63 (method)

- [previous-field], page 63 (method)
- [productions], page 64 (method)
- [(setf productions)], page 64 (method)
- [test], page 77 (class)
- [test-equal-p], page 65 (method)
- [tests], page 65 (method)
- [(setf tests)], page 65 (method)
- [tests-equal-p], page 66 (method)

2.1.12 rete-net-creation.lisp

Dependency
rete-beta-part

Parent [exil], page 1 (system)

Location rete-net-creation.lisp

Exported Definitions
- [add-wme], page 36 (method)
- [make-rete], page 33 (function)
- [new-production], page 40 (method)
- [rem-wme], page 41 (method)
- [remove-production], page 41 (method)

Internal Definitions
- [alpha-top-node], page 55 (method)
- [beta-top-node], page 55 (method)
- [(setf beta-top-node)], page 55 (method)
- [create-alpha-net], page 56 (method)
- [create-alpha-net], page 56 (method)
- [create-alpha-net%], page 56 (method)
- [create-alpha-net%], page 56 (method)
- [find-atom-in-cond-list%], page 51 (function)
- [find-test-node], page 57 (method)
- [find/create-join-node], page 57 (method)
- [find/create-neg-node], page 58 (method)
- [find/create-test-node], page 58 (generic function)
- [find/create-test-node], page 58 (method)
- [find/create-test-node], page 58 (method)
- [find/create-test-node%], page 58 (method)
- [get-intercondition-tests%], page 58 (method)
- [get-intercondition-tests%], page 58 (method)
- [get-intracondition-tests%], page 59 (method)
- [get-intracondition-tests%], page 59 (method)
- [get-join-tests-from-condition], page 59 (method)
- [rete], page 75 (class)

2.1.13 matches.lisp

Dependency
rete-net-creation

Parent [exil], page 1 (system)

Location matches.lisp

Internal Definitions
- [make-match], page 51 (function)
- [match], page 73 (class)
- [match-equal-p], page 61 (method)
- [match-rule], page 61 (method)
- [match-token], page 61 (method)
- [timestamp], page 66 (method)

2.1.14 activations.lisp

Dependency
matches

Parent [exil], page 1 (system)

Location activations.lisp

Exported Definitions
[activate-rule], page 35 (method)

Internal Definitions
- [get-variable-bindings], page 51 (function)
- [substitute-variables], page 52 (function)
- [variable-bindings], page 67 (method)
- [variable-bindings], page 67 (method)

2.1.15 strategies.lisp

Dependency
activations

Parent [exil], page 1 (system)

Location strategies.lisp

Internal Definitions
- [breadth-strategy], page 50 (function)
- [complexity-strategy], page 51 (function)
- [depth-strategy], page 51 (function)
- [newer-than], page 62 (method)
- [simpler-than], page 64 (method)
- [simpler-than], page 64 (method)
- [simplicity-strategy], page 52 (function)

2.1.16 environment.lisp

Dependency
`strategies`

Parent [exil], page 1 (system)

Location `environment.lisp`

Exported Definitions

- [add-fact], page 31 (function)
- [add-fact-group], page 31 (function)
- [add-match], page 36 (method)
- [add-rule], page 31 (function)
- [add-strategy], page 36 (method)
- [add-template], page 31 (function)
- [agenda], page 36 (method)
- [fact-groups], page 38 (method)
- [facts], page 39 (method)
- [find-fact], page 32 (function)
- [find-rule], page 32 (function)
- [find-template], page 39 (method)
- [modify-fact], page 33 (function)
- [rem-fact], page 33 (function)
- [rem-fact-group], page 33 (function)
- [rem-rule], page 34 (function)
- [remove-match], page 41 (method)
- [reset-environment], page 34 (function)
- [reset-facts], page 34 (function)
- [rete], page 41 (method)
- [rules], page 42 (method)
- [select-activation], page 42 (method)
- [set-strategy], page 42 (method)
- [set-watcher], page 42 (method)
- [templates], page 43 (method)
- [unset-watcher], page 44 (method)
- [unwatch-all], page 44 (method)
- [watch-all], page 44 (method)
- [watched-p], page 44 (method)

Internal Definitions

- [*current-environment*], page 49 (special variable)
- [*environments*], page 49 (special variable)
- [completely-reset-environment], page 50 (function)
- [current-strategy], page 57 (method)
- [current-strategy-name], page 57 (method)
- [defenv], page 49 (macro)

- [exil-env-accessor], page 49 (macro)
- [exil-env-accessors], page 49 (macro)
- [exil-env-reader], page 50 (macro)
- [exil-env-writer], page 50 (macro)
- [exil-environment], page 72 (class)
- [is-watcher], page 60 (method)
- [remove-matches], page 64 (method)
- [setenv], page 50 (macro)
- [strategies], page 65 (method)
- [watchers], page 67 (method)

2.1.17 export.lisp

Dependency
environment

Parent [exil], page 1 (system)

Location export.lisp

Exported Definitions

- [assert], page 29 (macro)
- [clear], page 32 (function)
- [deffacts], page 29 (macro)
- [defrule], page 29 (macro)
- [defstrategy], page 29 (macro)
- [deftemplate], page 29 (macro)
- [facts], page 32 (function)
- [halt], page 33 (function)
- [modify], page 30 (macro)
- [ppdefrule], page 30 (macro)
- [reset], page 34 (function)
- [retract], page 30 (macro)
- [retract-all], page 34 (function)
- [run], page 34 (function)
- [setstrategy], page 30 (macro)
- [step], page 34 (function)
- [undeffacts], page 31 (macro)
- [undefrule], page 31 (macro)
- [unwatch], page 31 (macro)
- [watch], page 31 (macro)

Internal Definitions

- [*clips-mode*], page 49 (special variable)
- [*exil-running*], page 49 (special variable)
- [assert%], page 50 (function)
- [assert-group%], page 50 (function)
- [clips->nonclips-mod-list], page 50 (function)

3 Packages

Packages are listed by definition order.

3.1 exil-system

Source [exil.asd], page 3 (Lisp file)

Use List

- asdf
- common-lisp

3.2 exil-utils

Source [packages], page 3 (Lisp file)

Use List common-lisp

Used By List

- [exil], page 26
- [exil-env], page 24
- [exil-rete], page 19
- [exil-core], page 16

Exported Definitions

- [alistp], page 31 (function)
- [assoc-key], page 31 (function)
- [assoc-value], page 32 (function)
- [(setf assoc-value)], page 32 (function)
- [class-slot-value], page 32 (function)
- [cpl-assoc-val], page 32 (function)
- [(setf cpl-assoc-val)], page 32 (function)
- [diff-delete], page 29 (macro)
- [doplist], page 29 (macro)
- [every-couple], page 32 (function)
- [exil-equal-p], page 37 (generic function)
- [exil-equal-p], page 37 (method)
- [exil-equal-p], page 37 (method)
- [exil-equal-p], page 37 (method)
- [exil-equal-p], page 37 (method)
- [exil-equal-p], page 37 (method)
- [exil-equal-p], page 37 (method)
- [exil-weak-equal-p], page 37 (generic function)
- [exil-weak-equal-p], page 37 (method)
- [exil-weak-equal-p], page 37 (method)
- [exil-weak-equal-p], page 37 (method)
- [ext-delete], page 29 (macro)
- [ext-pushnew], page 30 (macro)

- [from-keyword], page 33 (function)
- [hash->list], page 39 (generic function)
- [hash->list], page 39 (method)
- [intern], page 33 (function)
- [mac-exp], page 30 (macro)
- [my-pushnew], page 30 (macro)
- [plistp], page 33 (function)
- [push-end], page 30 (macro)
- [push-update], page 30 (macro)
- [pushnew-end], page 30 (macro)
- [select], page 34 (function)
- [string-append], page 34 (function)
- [subsets], page 34 (function)
- [symbol-append], page 35 (function)
- [symbol-name], page 42 (generic function)
- [symbol-name], page 43 (method)
- [symbol-name], page 43 (method)
- [to-keyword], page 35 (function)
- [to-list], page 35 (function)
- [to-list-of-lists], page 35 (function)
- [weak-symbol-equal-p], page 44 (generic function)
- [weak-symbol-equal-p], page 44 (method)
- [weak-symbol-equal-p], page 44 (method)

3.3 exil-core

Source [packages], page 3 (Lisp file)

Use List

- common-lisp
- [exil-utils], page 15

Used By List

- [exil], page 26
- [exil-env], page 24
- [exil-rete], page 19

Exported Definitions

- [activations], page 35 (generic function)
- [activations], page 35 (method)
- [atom-equal-p], page 36 (generic function)
- [atom-equal-p], page 36 (method)
- [atom-position], page 36 (generic function)
- [atom-position], page 36 (method)
- [atom-position], page 36 (method)
- [atom-position], page 36 (method)

Internal Definitions

3.4 exil-rete

Source [packages], page 3 (Lisp file)

Use List

- common-lisp
- [exil-utils], page 15
- [exil-core], page 16

Used By List

[exil-env], page 24

Exported Definitions

- [add-wme], page 36 (generic function)
- [add-wme], page 36 (method)
- [make-rete], page 33 (function)
- [new-production], page 40 (generic function)
- [new-production], page 40 (method)
- [rem-wme], page 41 (generic function)
- [rem-wme], page 41 (method)
- [remove-production], page 41 (generic function)
- [remove-production], page 41 (method)
- [token->list], page 43 (generic function)
- [token->list], page 43 (method)
- [token-equal-p], page 43 (generic function)
- [token-equal-p], page 43 (method)
- [token-equal-p], page 43 (method)
- [token-equal-p], page 44 (method)

Internal Definitions

- [activate], page 53 (generic function)
- [activate], page 53 (method)
- [activate], page 53 (method)
- [activate], page 53 (method)
- [activate], page 53 (method)
- [activate], page 53 (method)
- [activate], page 53 (method)
- [activate], page 53 (method)
- [activate], page 54 (method)
- [activate], page 54 (method)
- [activate-children], page 54 (generic function)
- [activate-children], page 54 (method)
- [activate-children], page 54 (method)
- [activate-memory], page 54 (generic function)
- [activate-memory], page 54 (method)
- [add-child], page 54 (generic function)
- [add-child], page 54 (method)

3.5 exil-env

Source [packages], page 3 (Lisp file)

Use List

- common-lisp
- [exil-utils], page 15
- [exil-core], page 16
- [exil-rete], page 19

Used By List

[exil], page 26

Exported Definitions

- [activate-rule], page 35 (generic function)
- [activate-rule], page 35 (method)
- [add-fact], page 31 (function)
- [add-fact-group], page 31 (function)
- [add-match], page 35 (generic function)
- [add-match], page 36 (method)
- [add-rule], page 31 (function)
- [add-strategy], page 36 (generic function)
- [add-strategy], page 36 (method)
- [add-template], page 31 (function)
- [agenda], page 36 (generic function)
- [agenda], page 36 (method)
- [fact-groups], page 38 (generic function)
- [fact-groups], page 38 (method)
- [facts], page 39 (generic function)
- [facts], page 39 (method)
- [find-fact], page 32 (function)
- [find-rule], page 32 (function)
- [find-template], page 39 (generic function)
- [find-template], page 39 (method)
- [modify-fact], page 33 (function)
- [rem-fact], page 33 (function)
- [rem-fact-group], page 33 (function)
- [rem-rule], page 34 (function)
- [remove-match], page 41 (generic function)
- [remove-match], page 41 (method)
- [reset-environment], page 34 (function)
- [reset-facts], page 34 (function)
- [rete], page 41 (generic function)
- [rete], page 41 (method)
- [rules], page 42 (generic function)
- [rules], page 42 (method)

Internal Definitions

- [match-token], page 61 (method)
- [newer-than], page 62 (generic function)
- [newer-than], page 62 (method)
- [remove-matches], page 64 (generic function)
- [remove-matches], page 64 (method)
- [setenv], page 50 (macro)
- [simpler-than], page 64 (generic function)
- [simpler-than], page 64 (method)
- [simpler-than], page 64 (method)
- [simplicity-strategy], page 52 (function)
- [strategies], page 64 (generic function)
- [strategies], page 65 (method)
- [substitute-variables], page 52 (function)
- [timestamp], page 66 (generic function)
- [timestamp], page 66 (method)
- [variable-bindings], page 67 (generic function)
- [variable-bindings], page 67 (method)
- [variable-bindings], page 67 (method)
- [watchers], page 67 (generic function)
- [watchers], page 67 (method)

3.6 exil

Source [packages], page 3 (Lisp file)

Use List

- common-lisp
- [exil-utils], page 15
- [exil-core], page 16
- [exil-env], page 24

Used By List

[exil-user], page 27

Exported Definitions

- [assert], page 29 (macro)
- [clear], page 32 (function)
- [deffacts], page 29 (macro)
- [defrule], page 29 (macro)
- [defstrategy], page 29 (macro)
- [deftemplate], page 29 (macro)
- [facts], page 32 (function)
- [halt], page 33 (function)
- [modify], page 30 (macro)
- [ppdefrule], page 30 (macro)
- [reset], page 34 (function)

- [retract], page 30 (macro)
- [retract-all], page 34 (function)
- [run], page 34 (function)
- [setstrategy], page 30 (macro)
- [step], page 34 (function)
- [undeffacts], page 31 (macro)
- [undefrule], page 31 (macro)
- [unwatch], page 31 (macro)
- [watch], page 31 (macro)

Internal Definitions

- [*clips-mode*], page 49 (special variable)
- [*exil-running*], page 49 (special variable)
- [assert%], page 50 (function)
- [assert-group%], page 50 (function)
- [clips->nonclips-mod-list], page 50 (function)
- [clips-mod-list-p], page 50 (function)
- [clips-slot->slot-des%], page 50 (function)
- [clips-slot-spec-p], page 50 (function)
- [extract-conditions%], page 51 (function)
- [modify%], page 61 (generic function)
- [modify%], page 61 (method)
- [modify%], page 61 (method)
- [my-position], page 52 (function)
- [nonclips-mod-list-p], page 52 (function)
- [nonclips-slot->slot-des%], page 52 (function)
- [nonclips-slot-spec-p], page 52 (function)
- [ppdefrule%], page 52 (function)
- [retract%], page 52 (function)
- [set-clips-mode], page 52 (function)
- [slot->slot-designator%], page 52 (function)
- [slot-spec-p], page 52 (function)
- [slots->slot-designators%], page 52 (function)
- [to-mod-spec-list], page 53 (function)

3.7 exil-user

Source [packages], page 3 (Lisp file)

Use List

- common-lisp
- [exil], page 26

4 Definitions

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

4.1 Exported definitions

4.1.1 Macros

`assert` **&rest** *FACT-SPECS* [Macro]
Add fact into working memory

Package [exil], page 26

Source [export], page 13 (Lisp file)

`deffacts` *NAME* **&body** *FACT-DESCRIPTIONS* [Macro]
Create group of facts to be asserted after (reset)

Package [exil], page 26

Source [export], page 13 (Lisp file)

`defrule` *NAME* **&body** *RULE* [Macro]
Define rule

Package [exil], page 26

Source [export], page 13 (Lisp file)

`defstrategy` *NAME FUNCTION* [Macro]
Define strategy

Package [exil], page 26

Source [export], page 13 (Lisp file)

`deftemplate` *NAME* **&body** *SLOTS* [Macro]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`diff-delete` *ITEM SEQUENCE* **&key** *TEST KEY* [Macro]
like delete, but as a second value returns list of deleted items

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`doplist` (*KEY VAL PLIST* **&optional** *RETVAL*) **&body** *BODY* [Macro]

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`ext-delete` *ITEM PLACE* **&key** *TEST KEY* [Macro]
like delete, but as a second value returns, whether the list was actualy altered

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`ext-pushnew` *ITEM PLACE* **&key** *TEST KEY* [Macro]
like pushnew, but as a second value returns, whether the list was actualy altered

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`mac-exp` **&body** *BODY* [Macro]
shortcut for calling macroexpand-1

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`modify` *FACT-SPEC* **&rest** *MOD-LIST* [Macro]
Replace old-fact by new-fact

Package [exil], page 26

Source [export], page 13 (Lisp file)

`my-pushnew` *ITEM PLACE* **&key** *TEST KEY* [Macro]
slightly altered pushnew

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`ppdefrule` *NAME* [Macro]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`push-end` *ITEM LIST* [Macro]

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`push-update` *ITEM PLACE* **&key** *TEST KEY* [Macro]
like pushnew, but if there is test-equal atom in the place, replaces it by item

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`pushnew-end` *ITEM LIST* **&key** *KEY TEST* [Macro]

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`retract` **&rest** *FACT-SPECS* [Macro]
Remove fact from working memory

Package [exil], page 26

Source [export], page 13 (Lisp file)

`setstrategy` *NAME* [Macro]
Set strategy to use

Package [exil], page 26

Source [export], page 13 (Lisp file)

`undeffacts` *NAME* [Macro]
Delete fact group

Package [exil], page 26

Source [export], page 13 (Lisp file)

`undefrule` *NAME* [Macro]
Undefine rule

Package [exil], page 26

Source [export], page 13 (Lisp file)

`unwatch` *WATCHER* [Macro]
Unwatch selected item

Package [exil], page 26

Source [export], page 13 (Lisp file)

`watch` *WATCHER* [Macro]
Watch selected item (facts, rules, activations)

Package [exil], page 26

Source [export], page 13 (Lisp file)

4.1.2 Functions

`add-fact` *FACT* [Function]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

`add-fact-group` *GROUP-NAME FACT-DESCRIPTIONS* [Function]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

`add-rule` *RULE* [Function]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

`add-template` *TEMPLATE* [Function]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

`alistp` *LIST* [Function]

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`assoc-key` *VALUE ALIST* [Function]
get key from assoc-list according to the value

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`assoc-value` *THE-KEY ALIST* **&key** *KEY TEST* [Function]
get value from assoc-list according to the key

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`(setf assoc-value)` *VALUE THE-KEY ALIST* **&key** *KEY TEST* [Function]
set value in assoc-list according to the key

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`class-slot-value` *CLASS-NAME SLOT-NAME* [Function]
get a class-slot value from class-name

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`clear` [Function]
Delete all facts

Package [exil], page 26

Source [export], page 13 (Lisp file)

`constant-test` *DESIRED-VALUE REAL-VALUE* [Function]

Package [exil-core], page 16

Source [patterns], page 6 (Lisp file)

`cpl-assoc-val` *KEY CPL-LIST* [Function]
`(setf cpl-assoc-val)` *NEW-VAL KEY CPL-LIST* [Function]
get value from couple list according to the key

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`every-couple` *PREDICATE LIST* [Function]

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`facts` **&optional** *START-INDEX END-INDEX AT-MOST* [Function]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`find-fact` *FACT* [Function]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

`find-rule` *NAME* [Function]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

`from-keyword` *KEY* **&optional** *PACKAGE* [Function]
get sybol from key

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`halt` [Function]
Stop the inference engine

Package [exil], page 26

Source [export], page 13 (Lisp file)

`intern` *STRING* **&optional** *PACKAGE* [Function]
create symbol from string

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`make-fact` *FACT-SPEC* [Function]

Package [exil-core], page 16

Source [facts], page 5 (Lisp file)

`make-pattern` *SPECIFICATION* **&key** *MATCH-VAR* [Function]

Package [exil-core], page 16

Source [patterns], page 6 (Lisp file)

`make-rete` [Function]

Package [exil-rete], page 19

Source [rete-net-creation], page 10 (Lisp file)

`make-template` *NAME SLOTS* [Function]

Package [exil-core], page 16

Source [templates], page 4 (Lisp file)

`modify-fact` *FACT MOD-LIST* [Function]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

`plistp` *LIST* [Function]

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`rem-fact` *FACT* [Function]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

`rem-fact-group` *NAME* [Function]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

`rem-rule` *RULE* [Function]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

`reset` [Function]

Clear all facts and add all fact groups

Package [exil], page 26

Source [export], page 13 (Lisp file)

`reset-environment` [Function]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

`reset-facts` [Function]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

`retract-all` [Function]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`run` [Function]

Run the infenece engine

Package [exil], page 26

Source [export], page 13 (Lisp file)

`select` *LIST INDICES* [Function]

get a list of values from list according to the list of indices

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`step` [Function]

Run inference engine for one turn

Package [exil], page 26

Source [export], page 13 (Lisp file)

`string-append` **&rest** *STRINGS* [Function]

append two strings

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`subsets` *LIST* [Function]

get a list of all subsets of given list

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`symbol-append` **&rest** *SYMBOLS* [Function]
concatenate several symbols

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`to-keyword` *SYMBOL* [Function]
get keyword form of symbol

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`to-list` *X* [Function]
when given an atom, returns list containing it, when given a list, just returns it

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`to-list-of-lists` *LIST* [Function]
collects return value of to-list on each element

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

`var-or-equal-p` *ATOM1 ATOM2* [Function]

Package [exil-core], page 16

Source [patterns], page 6 (Lisp file)

`variable-p` *EXPR* [Function]
is expr an exil variable?

Package [exil-core], page 16

Source [templates], page 4 (Lisp file)

4.1.3 Generic functions

`activate-rule` *ACTIVATION* [Generic Function]

Package [exil-env], page 24

Methods

`activate-rule` (*ACTIVATION* `match`) [Method]

Source [activations], page 11 (Lisp file)

`activations` *OBJECT* [Generic Function]

Package [exil-core], page 16

Methods

`activations` (*RULE* `rule`) [Method]
automatically generated reader method

Source [rules], page 6 (Lisp file)

`add-match` *PRODUCTION TOKEN* [Generic Function]

Package [exil-env], page 24

Methods

`add-match` *PRODUCTION TOKEN* [Method]

Source [environment], page 12 (Lisp file)

`add-strategy` *NAME FUNCTION* [Generic Function]

Package [exil-env], page 24

Methods

`add-strategy` *NAME FUNCTION* [Method]

Source [environment], page 12 (Lisp file)

`add-wme` *FACT* **&optional** *RETE* [Generic Function]

Package [exil-rete], page 19

Methods

`add-wme` (*FACT* `fact`) **&optional** *RETE* [Method]

Source [rete-net-creation], page 10 (Lisp file)

`agenda` [Generic Function]

Package [exil-env], page 24

Methods

`agenda` [Method]

Source [environment], page 12 (Lisp file)

`atom-equal-p` *OBJECT1 OBJECT2* [Generic Function]

equality predicate for fact atoms

Package [exil-core], page 16

Source [patterns], page 6 (Lisp file)

Methods

`atom-equal-p` *OBJECT1 OBJECT2* [Method]

Source [patterns], page 6 (Lisp file)

`atom-position` *OBJECT ATOM* [Generic Function]

Package [exil-core], page 16

Methods

`atom-position` (*PATTERN* `simple-pattern`) *ATOM* [Method]

Source [patterns], page 6 (Lisp file)

`atom-position` (*FACT* `simple-fact`) *ATOM* [Method]

Source [facts], page 5 (Lisp file)

`atom-position` (*OBJECT* `template-object`) *ATOM* [Method]

get the atom position in template-object slots

Source [templates], page 4 (Lisp file)

`conditions` *OBJECT* [Generic Function]

Package [exil-core], page 16

Methods

`conditions` (*RULE* `rule`) [Method]
automatically generated reader method

Source [rules], page 6 (Lisp file)

`copy-fact` *FACT* [Generic Function]

Package [exil-core], page 16

Methods

`copy-fact` (*FACT* `fact`) [Method]

Source [facts], page 5 (Lisp file)

`exil-equal-p` *OBJ1 OBJ2* [Generic Function]
ExiL default equality predicate

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

Methods

`exil-equal-p` (*OBJ1* `cons`) (*OBJ2* `cons`) [Method]

Source [utils], page 3 (Lisp file)

`exil-equal-p` (*OBJ1* `number`) (*OBJ2* `number`) [Method]

Source [utils], page 3 (Lisp file)

`exil-equal-p` (*OBJ1* `symbol`) (*OBJ2* `symbol`) [Method]

Source [utils], page 3 (Lisp file)

`exil-equal-p` (*OBJ1* `string`) (*OBJ2* `string`) [Method]

Source [utils], page 3 (Lisp file)

`exil-equal-p` *OBJ1 OBJ2* [Method]

Source [utils], page 3 (Lisp file)

`exil-equal-p` (*OBJ1* `null`) (*OBJ2* `null`) [Method]

Source [utils], page 3 (Lisp file)

`exil-weak-equal-p` *OBJ1 OBJ2* [Generic Function]
ExiL default weak equality predicate

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

Methods

`exil-weak-equal-p` (*OBJ1* `cons`) (*OBJ2* `cons`) [Method]

Source [utils], page 3 (Lisp file)

`exil-weak-equal-p` (*OBJ1* `symbol`) (*OBJ2* `symbol`) [Method]

Source [utils], page 3 (Lisp file)

`exil-weak-equal-p` *OBJ1 OBJ2* [Method]

Source [utils], page 3 (Lisp file)

`fact` *OBJECT* [Generic Function]

Package [exil-core], page 16

Methods

`fact` (*SIMPLE-FACT* `simple-fact`) [Method]
automatically generated reader method

Source [facts], page 5 (Lisp file)

`fact-description` *FACT* [Generic Function]

Package [exil-core], page 16

Methods

`fact-description` (*FACT* `template-fact`) [Method]

Source [facts], page 5 (Lisp file)

`fact-description` (*FACT* `simple-fact`) [Method]

Source [facts], page 5 (Lisp file)

`fact-equal-p` *FACT1 FACT2* [Generic Function]

Package [exil-core], page 16

Source [facts], page 5 (Lisp file)

Methods

`fact-equal-p` (*FACT1* `template-fact`) (*FACT2* `template-fact`) [Method]

Source [facts], page 5 (Lisp file)

`fact-equal-p` (*FACT1* `simple-fact`) (*FACT2* `simple-fact`) [Method]

Source [facts], page 5 (Lisp file)

`fact-equal-p` *FACT1 FACT2* [Method]

Source [facts], page 5 (Lisp file)

`fact-groups` [Generic Function]

Package [exil-env], page 24

Methods

`fact-groups` [Method]

Source [environment], page 12 (Lisp file)

`fact-slot` *FACT SLOT-SPEC* [Generic Function]
returns fact's slot specified by slot-spec

Package [exil-core], page 16

Source [facts], page 5 (Lisp file)

Methods

`fact-slot` (*FACT* `simple-fact`) (*SLOT-SPEC* `integer`) [Method]

Source [facts], page 5 (Lisp file)

`fact-slot` (*FACT* `template-fact`) (*SLOT-SPEC* `symbol`) [Method]

Source [facts], page 5 (Lisp file)

`facts` [Generic Function]

Package [exil-env], page 24

Methods

`facts` [Method]

Source [environment], page 12 (Lisp file)

`find-atom` *OBJECT ATOM* [Generic Function]

Package [exil-core], page 16

Methods

`find-atom` (*PATTERN* `simple-pattern`) *ATOM* [Method]

Source [patterns], page 6 (Lisp file)

`find-atom` (*FACT* `simple-fact`) *ATOM* [Method]

Source [facts], page 5 (Lisp file)

`find-atom` (*OBJECT* `template-object`) *ATOM* [Method]

find the given atom in template-object slots

Source [templates], page 4 (Lisp file)

`find-template` *NAME* [Generic Function]

Package [exil-env], page 24

Methods

`find-template` *NAME* [Method]

Source [environment], page 12 (Lisp file)

`has-slot-p` *OBJECT SLOT-NAME* [Generic Function]

Package [exil-core], page 16

Methods

`has-slot-p` (*OBJECT* `template-object`) *SLOT-NAME* [Method]

Source [templates], page 4 (Lisp file)

`hash->list` *HASH* [Generic Function]

Package [exil-utils], page 15

Methods

`hash->list` (*HASH* `hash-table`) [Method]

Source [utils], page 3 (Lisp file)

`make-rule` *NAME CONDITIONS ACTIVATIONS* [Generic Function]

Package [exil-core], page 16

Methods

`make-rule` *NAME CONDITIONS ACTIVATIONS* [Method]

Source [rules], page 6 (Lisp file)

`match-var` *OBJECT* [Generic Function]
`(setf match-var)` *NEW-VALUE OBJECT* [Generic Function]

Package [exil-core], page 16

Methods

`match-var` (*PATTERN* `pattern`) [Method]
automatically generated reader method

Source [patterns], page 6 (Lisp file)

`(setf match-var)` *NEW-VALUE* (*PATTERN* `pattern`) [Method]
automatically generated writer method

Source [patterns], page 6 (Lisp file)

`name` *OBJECT* [Generic Function]

Package [exil-core], page 16

Methods

`name` (*RULE* `rule`) [Method]
automatically generated reader method

Source [rules], page 6 (Lisp file)

`name` (*TEMPLATE* `template`) [Method]
automatically generated reader method

Source [templates], page 4 (Lisp file)

`negated-p` *OBJECT* [Generic Function]
`(setf negated-p)` *NEW-VALUE OBJECT* [Generic Function]

Package [exil-core], page 16

Methods

`negated-p` (*PATTERN* `pattern`) [Method]
automatically generated reader method

Source [patterns], page 6 (Lisp file)

`(setf negated-p)` *NEW-VALUE* (*PATTERN* `pattern`) [Method]
automatically generated writer method

Source [patterns], page 6 (Lisp file)

`new-production` *RULE* **&optional** *RETE* [Generic Function]

Package [exil-rete], page 19

Methods

`new-production` (*RULE* `rule`) **&optional** *RETE* [Method]

Source [rete-net-creation], page 10 (Lisp file)

`pattern` *OBJECT* [Generic Function]

Package [exil-core], page 16

Methods

`pattern` (*SIMPLE-PATTERN* `simple-pattern`) [Method]
automatically generated reader method

Source [patterns], page 6 (Lisp file)

`pattern-equal-p` *PATTERN1 PATTERN2* [Generic Function]

Package [exil-core], page 16

Source [patterns], page 6 (Lisp file)

Methods

`pattern-equal-p` (*PATTERN1* `template-pattern`) (*PATTERN2* `template-pattern`) [Method]

Source [patterns], page 6 (Lisp file)

`pattern-equal-p` (*PATTERN1* `simple-pattern`) (*PATTERN2* `simple-pattern`) [Method]

Source [patterns], page 6 (Lisp file)

`pattern-equal-p` (*PATTERN1* `pattern`) (*PATTERN2* `pattern`) [Method]

Source [patterns], page 6 (Lisp file)

`rem-wme` *FACT* **&optional** *RETE* [Generic Function]

Package [exil-rete], page 19

Methods

`rem-wme` (*FACT* `fact`) **&optional** *RETE* [Method]

Source [rete-net-creation], page 10 (Lisp file)

`remove-match` *PRODUCTION TOKEN* [Generic Function]

Package [exil-env], page 24

Methods

`remove-match` *PRODUCTION TOKEN* [Method]

Source [environment], page 12 (Lisp file)

`remove-production` *RULE* **&optional** *RETE* [Generic Function]

Package [exil-rete], page 19

Methods

`remove-production` (*RULE* `rule`) **&optional** *RETE* [Method]

Source [rete-net-creation], page 10 (Lisp file)

`rete` [Generic Function]

Package [exil-env], page 24

Methods

`rete` [Method]

Source [environment], page 12 (Lisp file)

`rule-equal-p` *RULE1 RULE2* [Generic Function]

Package [exil-core], page 16

Methods

`rule-equal-p` (*RULE1* `rule`) (*RULE2* `rule`) [Method]

Source [rules], page 6 (Lisp file)

`rules` [Generic Function]

Package [exil-env], page 24

Methods

`rules` [Method]

Source [environment], page 12 (Lisp file)

`select-activation` [Generic Function]

Package [exil-env], page 24

Methods

`select-activation` [Method]

Source [environment], page 12 (Lisp file)

`set-strategy` **&optional** *NAME* [Generic Function]

Package [exil-env], page 24

Methods

`set-strategy` **&optional** *NAME* [Method]

Source [environment], page 12 (Lisp file)

`set-watcher` *WATCHER* [Generic Function]

Package [exil-env], page 24

Methods

`set-watcher` *WATCHER* [Method]

Source [environment], page 12 (Lisp file)

`slots` *OBJECT* [Generic Function]

Package [exil-core], page 16

Methods

`slots` (*TEMPLATE-OBJECT* `template-object`) [Method]

automatically generated reader method

Source [templates], page 4 (Lisp file)

`slots` (*TEMPLATE* `template`) [Method]

automatically generated reader method

Source [templates], page 4 (Lisp file)

`symbol-name` *SYMBOL* [Generic Function]

For symbol return its name, for string just return itself

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

Methods

`symbol-name` (*SYMBOL* `symbol`) [Method]

Source [utils], page 3 (Lisp file)

`symbol-name` (*STRING* `string`) [Method]

Source [utils], page 3 (Lisp file)

`templates` [Generic Function]

Package [exil-env], page 24

Methods

`templates` [Method]

Source [environment], page 12 (Lisp file)

`tmpl-fact-slot-value` *FACT SLOT-NAME* [Generic Function]
`(setf tmpl-fact-slot-value)` *VAL FACT SLOT-NAME* [Generic Function]

Package [exil-core], page 16

Methods

`tmpl-fact-slot-value` (*FACT* `template-fact`) *SLOT-NAME* [Method]
`(setf tmpl-fact-slot-value)` *VAL* (*FACT* `template-fact`) *SLOT-NAME* [Method]

Source [facts], page 5 (Lisp file)

`tmpl-name` *OBJECT* [Generic Function]

Package [exil-core], page 16

Methods

`tmpl-name` (*TEMPLATE-OBJECT* `template-object`) [Method]
automatically generated reader method

Source [templates], page 4 (Lisp file)

`token->list` *TOKEN* [Generic Function]

Package [exil-rete], page 19

Methods

`token->list` (*TOKEN* `token`) [Method]

Source [tokens], page 7 (Lisp file)

`token-equal-p` *TOKEN1 TOKEN2* [Generic Function]
token equality predicate

Package [exil-rete], page 19

Source [tokens], page 7 (Lisp file)

Methods

`token-equal-p` *TOKEN1 TOKEN2* [Method]

Source [tokens], page 7 (Lisp file)

`token-equal-p` (*TOKEN1* `empty-token`) (*TOKEN2* `empty-token`) [Method]

Source [tokens], page 7 (Lisp file)

`token-equal-p` (*TOKEN1* `token`) (*TOKEN2* `token`) [Method]

Source [tokens], page 7 (Lisp file)

`unset-watcher` *WATCHER* [Generic Function]

Package [exil-env], page 24

Methods

`unset-watcher` *WATCHER* [Method]

Source [environment], page 12 (Lisp file)

`unwatch-all` [Generic Function]

Package [exil-env], page 24

Methods

`unwatch-all` [Method]

Source [environment], page 12 (Lisp file)

`watch-all` [Generic Function]

Package [exil-env], page 24

Methods

`watch-all` [Method]

Source [environment], page 12 (Lisp file)

`watched-p` *WATCHER* [Generic Function]

Package [exil-env], page 24

Methods

`watched-p` *WATCHER* [Method]

Source [environment], page 12 (Lisp file)

`weak-symbol-equal-p` *SYM1 SYM2* [Generic Function]

Test if the symbol name is equal, omits the package name

Package [exil-utils], page 15

Source [utils], page 3 (Lisp file)

Methods

`weak-symbol-equal-p` (*SYM1* `symbol`) (*SYM2* `symbol`) [Method]

Source [utils], page 3 (Lisp file)

`weak-symbol-equal-p` *SYM1 SYM2* [Method]

Source [utils], page 3 (Lisp file)

4.1.4 Classes

`fact` [Class]

Package [exil-core], page 16

Source [facts], page 5 (Lisp file)

Direct superclasses

`standard-object`

Direct subclasses

- [simple-fact], page 47 (class)
- [template-fact], page 48 (class)

Direct methods

- [rem-wme], page 41 (method)
- [add-wme], page 36 (method)
- [inactivate], page 59 (method)
- [activate], page 53 (method)
- [activate], page 53 (method)
- [perform-join-tests], page 63 (method)
- [perform-join-test], page 63 (method)
- [inactivate], page 59 (method)
- [inactivate], page 60 (method)
- [activate], page 53 (method)
- [inactivate], page 60 (method)
- [activate], page 53 (method)
- [activate], page 54 (method)
- [inactivate], page 60 (method)
- [activate], page 54 (method)
- [activate-memory], page 54 (method)
- [activate-children], page 54 (method)
- [test], page 65 (method)
- [includes-p], page 60 (method)
- [copy-fact], page 37 (method)

`pattern` [Class]

Package [exil-core], page 16

Source [patterns], page 6 (Lisp file)

Direct superclasses

`standard-object`

Direct subclasses

- [simple-pattern], page 47 (class)
- [template-pattern], page 48 (class)

Direct methods

- [get-join-tests-from-condition], page 59 (method)
- [pattern-equal-p], page 41 (method)

- match-var
- [match-var], page 40 (method)
- negated-p
- [negated-p], page 40 (method)

Direct slots

`negated` [Slot]

Initargs :negated

Readers [negated-p], page 40 (generic function)

Writers [(setf negated-p)], page 40 (generic function)

`match-variable` [Slot]

Initargs :match-var

Readers [match-var], page 40 (generic function)

Writers [(setf match-var)], page 40 (generic function)

`rule` [Class]

Package [exil-core], page 16

Source [rules], page 6 (Lisp file)

Direct superclasses

standard-object

Direct methods

- [simpler-than], page 64 (method)
- [remove-production], page 41 (method)
- [new-production], page 40 (method)
- [delete-production], page 57 (method)
- [add-production], page 55 (method)
- print-object
- [rule-equal-p], page 42 (method)
- [activations], page 35 (method)
- [conditions], page 37 (method)
- [name], page 40 (method)

Direct slots

`name` [Slot]

Initargs :name

Readers [name], page 40 (generic function)

`conditions` [Slot]

Initargs :conditions

Readers [conditions], page 36 (generic function)

`activations` [Slot]

Initargs :activations

Readers [activations], page 35 (generic function)

`simple-fact` [Class]

Package [exil-core], page 16

Source [facts], page 5 (Lisp file)

Direct superclasses

[fact], page 45 (class)

Direct methods

- [variable-bindings], page 67 (method)
- [test], page 65 (method)
- [fact-slot], page 38 (method)
- [fact-description], page 38 (method)
- [atom-position], page 36 (method)
- [find-atom], page 39 (method)
- [fact-equal-p], page 38 (method)
- `print-object`
- `initialize-instance`
- [fact], page 38 (method)

Direct slots

`fact` [Slot]

Initargs `:fact`

Initform `(error "fact slot must be specified")`

Readers [fact], page 38 (generic function)

`simple-pattern` [Class]

Package [exil-core], page 16

Source [patterns], page 6 (Lisp file)

Direct superclasses

[pattern], page 45 (class)

Direct methods

- [variable-bindings], page 67 (method)
- [get-intracondition-tests%], page 59 (method)
- [get-intercondition-tests%], page 58 (method)
- [create-alpha-net], page 56 (method)
- [create-alpha-net%], page 56 (method)
- [atom-position], page 36 (method)
- [find-atom], page 39 (method)
- [pattern-equal-p], page 41 (method)
- `print-object`
- [pattern], page 41 (method)

Direct slots

`pattern` [Slot]

Initargs `:pattern`

Initform `(error "pattern slot must be specified")`

Readers [pattern], page 40 (generic function)

`template` [Class]

Package [exil-core], page 16

Source [templates], page 4 (Lisp file)

Direct superclasses

`standard-object`

Direct methods

- `print-object`
- [slots], page 42 (method)
- [name], page 40 (method)

Direct slots

`name` [Slot]

Initargs `:name`

Initform `(error "name slot has to be specified")`

Readers [name], page 40 (generic function)

`slots` [Slot]

Initargs `:slots`

Initform `(error "slots slot has to be specified")`

Readers [slots], page 42 (generic function)

`template-fact` [Class]

Package [exil-core], page 16

Source [facts], page 5 (Lisp file)

Direct superclasses

- [template-object], page 76 (class)
- [fact], page 45 (class)

Direct methods

- [variable-bindings], page 67 (method)
- [test], page 65 (method)
- [fact-slot], page 39 (method)
- [fact-description], page 38 (method)
- [fact-equal-p], page 38 (method)
- `tmpl-fact-slot-value`
- [tmpl-fact-slot-value], page 43 (method)
- `initialize-instance`

`template-pattern` [Class]

Package [exil-core], page 16

Source [patterns], page 6 (Lisp file)

Direct superclasses

- [template-object], page 76 (class)
- [pattern], page 45 (class)

Direct methods

- [variable-bindings], page 67 (method)
- [get-intracondition-tests%], page 59 (method)
- [get-intercondition-tests%], page 58 (method)
- [create-alpha-net], page 56 (method)
- [create-alpha-net%], page 56 (method)
- print-object
- [pattern-equal-p], page 41 (method)
- [tmpl-pattern-slot-value], page 66 (method)

Direct slots

`slot-default` [Slot]

Allocation `:class`

Initform `'exil-core::?`

4.2 Internal definitions

4.2.1 Special variables

`*clips-mode*` [Special Variable]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`*current-environment*` [Special Variable]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

`*environments*` [Special Variable]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

`*exil-running*` [Special Variable]

Package [exil], page 26

Source [export], page 13 (Lisp file)

4.2.2 Macros

`defenv` *NAME* **&key** *REDEFINE* [Macro]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

`exil-env-accessor` *SLOT-NAME* [Macro]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

`exil-env-accessors` **&rest** *SLOT-NAMES* [Macro]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

`exil-env-reader` *SLOT-NAME* [Macro]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

`exil-env-writer` *SLOT-NAME* [Macro]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

`setenv` *NAME* [Macro]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

4.2.3 Functions

`assert%` *FACT-SPEC* [Function]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`assert-group%` *FACT-DESCRIPTIONS* [Function]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`breadth-strategy` *AGENDA* [Function]

Package [exil-env], page 24

Source [strategies], page 11 (Lisp file)

`clips->nonclips-mod-list` *MOD-LIST* [Function]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`clips-mod-list-p` *MOD-LIST* [Function]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`clips-slot->slot-des%` *SLOT-SPEC* [Function]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`clips-slot-spec-p` *SLOT-SPEC* [Function]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`clips-tmpl-slot-spec-p` *SPECIFICATION* [Function]

Package [exil-core], page 16

Source [templates], page 4 (Lisp file)

`completely-reset-environment` [Function]

Package [exil-env], page 24

Source [environment], page 12 (Lisp file)

`complexity-strategy` *AGENDA* [Function]

Package [exil-env], page 24

Source [strategies], page 11 (Lisp file)

`depth-strategy` *AGENDA* [Function]

Package [exil-env], page 24

Source [strategies], page 11 (Lisp file)

`extract-conditions%` *COND-LIST* [Function]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`find-atom-in-cond-list%` *ATOM COND-LIST* [Function]

Package [exil-rete], page 19

Source [rete-net-creation], page 10 (Lisp file)

`get-variable-bindings` *PATTERN-LIST FACT-LIST* [Function]

Package [exil-env], page 24

Source [activations], page 11 (Lisp file)

`make-match` *RULE TOKEN* **&optional** *TIMESTAMP* [Function]

Package [exil-env], page 24

Source [matches], page 11 (Lisp file)

`make-test` *CURRENT-FIELD PREVIOUS-CONDITION PREVIOUS-FIELD* [Function]

Package [exil-rete], page 19

Source [rete-beta-part], page 9 (Lisp file)

`make-tmpl-fact` *FACT-SPEC* [Function]

Package [exil-core], page 16

Source [facts], page 5 (Lisp file)

`make-tmpl-obj-clips` *OBJECT-TYPE TEMPLATE SLOT-SPECS* [Function]

Package [exil-core], page 16

Source [templates], page 4 (Lisp file)

`make-tmpl-obj-nonclips` *OBJECT-TYPE TEMPLATE SLOT-SPECS* [Function]

Package [exil-core], page 16

Source [templates], page 4 (Lisp file)

`make-tmpl-object` *SPECIFICATION OBJECT-TYPE* [Function]

creates template-object of given type from its specification

Package [exil-core], page 16

Source [templates], page 4 (Lisp file)

`make-tmpl-pattern` *PATTERN-SPEC* **&optional** *NEGATED MATCH-VAR* [Function]

Package [exil-core], page 16

Source [patterns], page 6 (Lisp file)

`my-position` *ATOM LIST* [Function]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`nonclips-mod-list-p` *MOD-LIST* [Function]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`nonclips-slot->slot-des%` *SLOT-SPEC* [Function]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`nonclips-slot-spec-p` *SLOT-SPEC* [Function]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`ppdefrule%` *NAME* [Function]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`retract%` *FACT-SPECS* [Function]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`set-clips-mode` *VAL* [Function]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`simplicity-strategy` *AGENDA* [Function]

Package [exil-env], page 24

Source [strategies], page 11 (Lisp file)

`slot->slot-designator%` *SLOT-SPEC* [Function]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`slot-spec-p` *SLOT-SPEC* [Function]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`slots->slot-designators%` *SLOTS* [Function]

Package [exil], page 26

Source [export], page 13 (Lisp file)

`substitute-variables` *ACTIVATIONS-WITH-VARS VAR-BIND-LIST* [Function]

Package [exil-env], page 24

Source [activations], page 11 (Lisp file)

`tmpl-fact-specification-p` *FACT-SPEC* [Function]

Package [exil-core], page 16

Source [facts], page 5 (Lisp file)

`tmpl-object-specification-p` *SPECIFICATION* [Function]

is this a template-object specification?

Package [exil-core], page 16

Source [templates], page 4 (Lisp file)

`tmpl-pattern-specification-p` *SPECIFICATION* [Function]

Package [exil-core], page 16

Source [patterns], page 6 (Lisp file)

`tmpl-slot-spec-p` *SPECIFICATION* [Function]

Package [exil-core], page 16

Source [templates], page 4 (Lisp file)

`to-mod-spec-list` *MOD-LIST* [Function]

Package [exil], page 26

Source [export], page 13 (Lisp file)

4.2.4 Generic functions

`activate` *NODE OBJECT* [Generic Function]

handels various node activations

Package [exil-rete], page 19

Source [rete-generic-node], page 7 (Lisp file)

Methods

`activate` (*NODE* `beta-negative-node`) (*WME* `fact`) [Method]

Source [rete-beta-part], page 9 (Lisp file)

`activate` (*NODE* `beta-negative-node`) (*TOKEN* `token`) [Method]

Source [rete-beta-part], page 9 (Lisp file)

`activate` (*NODE* `beta-join-node`) (*WME* `fact`) [Method]

Source [rete-beta-part], page 9 (Lisp file)

`activate` (*NODE* `beta-join-node`) (*TOKEN* `token`) [Method]

Source [rete-beta-part], page 9 (Lisp file)

`activate` (*NODE* `beta-memory-node`) (*TOKEN* `token`) [Method]

Source [rete-beta-part], page 9 (Lisp file)

`activate` (*NODE* `alpha-memory-node`) (*WME* `fact`) [Method]

Source [rete-alpha-part], page 8 (Lisp file)

`activate` (*NODE* `alpha-top-node`) (*WME* `fact`) [Method]

Source [rete-alpha-part], page 8 (Lisp file)

`activate` (*NODE* `alpha-subtop-node`) (*WME* `fact`) [Method]

Source [rete-alpha-part], page 8 (Lisp file)

`activate` (*NODE* `alpha-test-node`) (*WME* `fact`) [Method]

Source [rete-alpha-part], page 8 (Lisp file)

`activate-children` *NODE OBJECT* [Generic Function]

Package [exil-rete], page 19

Source [rete-generic-node], page 7 (Lisp file)

Methods

`activate-children` (*NODE* `alpha-test-node`) (*WME* `fact`) [Method]

Source [rete-alpha-part], page 8 (Lisp file)

`activate-children` (*NODE* `node`) *OBJECT* [Method]

Source [rete-generic-node], page 7 (Lisp file)

`activate-memory` *NODE WME* [Generic Function]

Package [exil-rete], page 19

Methods

`activate-memory` (*NODE* `alpha-test-node`) (*WME* `fact`) [Method]

Source [rete-alpha-part], page 8 (Lisp file)

`add-child` *NODE CHILD* [Generic Function]

Package [exil-rete], page 19

Methods

`add-child` (*NODE* `node`) (*CHILD* `node`) [Method]

Source [rete-generic-node], page 7 (Lisp file)

`add-children` *NODE CHILDREN* [Generic Function]

Package [exil-rete], page 19

Methods

`add-children` (*NODE* `node`) (*CHILDREN* `list`) [Method]

Source [rete-generic-node], page 7 (Lisp file)

`add-item` *NODE ITEM* **&optional** *EQUALITY-PREDICATE* [Generic Function]

Package [exil-rete], page 19

Methods

`add-item` (*NODE* `memory-node`) *ITEM* **&optional** *EQUALITY-PREDICATE* [Method]

Source [rete-generic-node], page 7 (Lisp file)

`add-production` *NODE PRODUCTION* [Generic Function]

Package [exil-rete], page 19

Methods

`add-production` (*NODE* `beta-memory-node`) (*PRODUCTION* `rule`) [Method]

Source [rete-beta-part], page 9 (Lisp file)

`alpha-memory` *OBJECT* [Generic Function]

Package [exil-rete], page 19

Methods

`alpha-memory` (*BETA-JOIN-NODE* `beta-join-node`) [Method]
automatically generated reader method

Source [rete-beta-part], page 9 (Lisp file)

`alpha-top-node` *OBJECT* [Generic Function]

Package [exil-rete], page 19

Methods

`alpha-top-node` (*RETE* `rete`) [Method]
automatically generated reader method

Source [rete-net-creation], page 10 (Lisp file)

`beta-memory` *NODE* [Generic Function]

Package [exil-rete], page 19

Methods

`beta-memory` (*NODE* `beta-join-node`) [Method]

Source [rete-beta-part], page 9 (Lisp file)

`beta-top-node` *OBJECT* [Generic Function]
`(setf beta-top-node)` *NEW-VALUE OBJECT* [Generic Function]

Package [exil-rete], page 19

Methods

`beta-top-node` (*RETE* `rete`) [Method]
automatically generated reader method

Source [rete-net-creation], page 10 (Lisp file)

`(setf beta-top-node)` *NEW-VALUE* (*RETE* `rete`) [Method]
automatically generated writer method

Source [rete-net-creation], page 10 (Lisp file)

`broken-match` *NODE TOKEN* [Generic Function]

Package [exil-rete], page 19

Methods

`broken-match` (*NODE* `beta-memory-node`) (*TOKEN* `token`) [Method]

Source [rete-beta-part], page 9 (Lisp file)

`children` *OBJECT* [Generic Function]
`(setf children)` *NEW-VALUE OBJECT* [Generic Function]

Package [exil-rete], page 19

Methods

`children` (*NODE* `node`) [Method]
automatically generated reader method

Source [rete-generic-node], page 7 (Lisp file)

`(setf children)` *NEW-VALUE* (*NODE* `node`) [Method]
automatically generated writer method

Source [rete-generic-node], page 7 (Lisp file)

`complete-match` *NODE TOKEN* [Generic Function]

Package [exil-rete], page 19

Methods

`complete-match` (*NODE* `beta-memory-node`) (*TOKEN* `token`) [Method]

Source [rete-beta-part], page 9 (Lisp file)

`create-alpha-net` *PATTERN* **&optional** *RETE* [Generic Function]

Package [exil-rete], page 19

Methods

`create-alpha-net` (*PATTERN* `template-pattern`) **&optional** *RETE* [Method]

Source [rete-net-creation], page 10 (Lisp file)

`create-alpha-net` (*PATTERN* `simple-pattern`) **&optional** *RETE* [Method]

Source [rete-net-creation], page 10 (Lisp file)

`create-alpha-net%` *PATTERN ROOT* [Generic Function]

Package [exil-rete], page 19

Methods

`create-alpha-net%` (*PATTERN* `template-pattern`) (*ROOT* `template-fact-subtop-node`) [Method]

Source [rete-net-creation], page 10 (Lisp file)

`create-alpha-net%` (*PATTERN* `simple-pattern`) (*ROOT* `simple-fact-subtop-node`) [Method]

Source [rete-net-creation], page 10 (Lisp file)

`current-field` *OBJECT* [Generic Function]

Package [exil-rete], page 19

Methods

`current-field` (*TEST* `test`) [Method]
automatically generated reader method

Source [rete-beta-part], page 9 (Lisp file)

`current-strategy` [Generic Function]

Package [exil-env], page 24

Methods

`current-strategy` [Method]

Source [environment], page 12 (Lisp file)

`current-strategy-name` [Generic Function]

Package [exil-env], page 24

Methods

`current-strategy-name` [Method]

Source [environment], page 12 (Lisp file)

`delete-production` *NODE PRODUCTION* [Generic Function]

Package [exil-rete], page 19

Methods

`delete-production` (*NODE* `beta-memory-node`) (*PRODUCTION* `rule`) [Method]

Source [rete-beta-part], page 9 (Lisp file)

`description` *OBJECT* [Generic Function]
`(setf description)` *NEW-VALUE OBJECT* [Generic Function]

Package [exil-rete], page 19

Methods

`description` (*DESCRIBED-OBJECT* `described-object`) [Method]

automatically generated reader method

Source [rete-generic-node], page 7 (Lisp file)

`(setf description)` *NEW-VALUE* (*DESCRIBED-OBJECT* `described-object`) [Method]

automatically generated writer method

Source [rete-generic-node], page 7 (Lisp file)

`find-test-node` *PARENT FIELD VALUE* [Generic Function]

Package [exil-rete], page 19

Methods

`find-test-node` (*PARENT* `alpha-node`) *FIELD VALUE* [Method]

Source [rete-net-creation], page 10 (Lisp file)

`find/create-join-node` *PARENT TESTS A-MEMORY* [Generic Function]

Package [exil-rete], page 19

Methods

`find/create-join-node` (*PARENT* `beta-memory-node`) (*TESTS* `list`) (*A-MEMORY* `alpha-memory-node`) [Method]

Source [rete-net-creation], page 10 (Lisp file)

`find/create-neg-node` *PARENT TESTS A-MEMORY* [Generic Function]

Package [exil-rete], page 19

Methods

`find/create-neg-node` (*PARENT* `beta-memory-node`) (*TESTS* `list`) (*A-MEMORY* `alpha-memory-node`) [Method]

Source [rete-net-creation], page 10 (Lisp file)

`find/create-test-node` *PARENT FIELD VALUE* [Generic Function]

Package [exil-rete], page 19

Source [rete-net-creation], page 10 (Lisp file)

Methods

`find/create-test-node` (*PARENT* `simple-fact-alpha-node`) *FIELD VALUE* [Method]

Source [rete-net-creation], page 10 (Lisp file)

`find/create-test-node` (*PARENT* `template-fact-alpha-node`) *FIELD VALUE* [Method]

Source [rete-net-creation], page 10 (Lisp file)

`find/create-test-node%` *PARENT FIELD VALUE NEW-NODE-TYPE* [Generic Function]

Package [exil-rete], page 19

Methods

`find/create-test-node%` *PARENT FIELD VALUE NEW-NODE-TYPE* [Method]

Source [rete-net-creation], page 10 (Lisp file)

`get-bad-wmes` *NODE TOKEN* [Generic Function]

Package [exil-rete], page 19

Methods

`get-bad-wmes` (*NODE* `beta-negative-node`) (*TOKEN* `token`) [Method]

Source [rete-beta-part], page 9 (Lisp file)

`get-intercondition-tests%` *CONDITION PREV-CONDS* [Generic Function]

Package [exil-rete], page 19

Methods

`get-intercondition-tests%` (*CONDITION* `template-pattern`) (*PREV-CONDS* `list`) [Method]

Source [rete-net-creation], page 10 (Lisp file)

`get-intercondition-tests%` (*CONDITION* `simple-pattern`) (*PREV-CONDS* `list`) [Method]

Source [rete-net-creation], page 10 (Lisp file)

`get-intracondition-tests%` *CONDITION* [Generic Function]

Package [exil-rete], page 19

Methods

`get-intracondition-tests%` (*CONDITION* `template-pattern`) [Method]

Source [rete-net-creation], page 10 (Lisp file)

`get-intracondition-tests%` (*CONDITION* `simple-pattern`) [Method]

Source [rete-net-creation], page 10 (Lisp file)

`get-join-tests-from-condition` *CONDITION PREV-CONDS* [Generic Function]

Package [exil-rete], page 19

Methods

`get-join-tests-from-condition` (*CONDITION* `pattern`) (*PREV-CONDS* `list`) [Method]

Source [rete-net-creation], page 10 (Lisp file)

`get-network` *NODE* **&optional** *TEMPLATE-NAME* [Generic Function]

`(setf get-network)` *VALUE NODE* **&optional** *TEMPLATE-NAME* [Generic Function]

Package [exil-rete], page 19

Methods

`get-network` (*NODE* `alpha-top-node`) **&optional** *TEMPLATE-NAME* [Method]

`(setf get-network)` *VALUE* (*NODE* `alpha-top-node`) **&optional** *TEMPLATE-NAME* [Method]

Source [rete-alpha-part], page 8 (Lisp file)

`get/initialize-network` *NODE* **&optional** *TEMPLATE-NAME* [Generic Function]

Package [exil-rete], page 19

Methods

`get/initialize-network` (*NODE* `alpha-top-node`) **&optional** *TEMPLATE-NAME* [Method]

Source [rete-alpha-part], page 8 (Lisp file)

`inactivate` *NODE OBJECT* [Generic Function]

Package [exil-rete], page 19

Source [rete-generic-node], page 7 (Lisp file)

Methods

`inactivate` (*NODE* `beta-negative-node`) (*WME* `fact`) [Method]

Source [rete-beta-part], page 9 (Lisp file)

`inactivate` (*NODE* `beta-memory-node`) (*TOKEN* `token`) `before` [Method]

Source [rete-beta-part], page 9 (Lisp file)

`inactivate` (*NODE* `beta-memory-node`) (*FACT* `fact`) `before` [Method]

Source [rete-beta-part], page 9 (Lisp file)

`inactivate` (*NODE* `alpha-memory-node`) (*WME* `fact`) [Method]
Source [rete-alpha-part], page 8 (Lisp file)

`inactivate` (*NODE* `alpha-top-node`) (*WME* `fact`) [Method]
Source [rete-alpha-part], page 8 (Lisp file)

`inactivate` (*NODE* `alpha-test-node`) (*WME* `fact`) `after` [Method]
Source [rete-alpha-part], page 8 (Lisp file)

`inactivate` (*NODE* `node`) *OBJECT* [Method]
Source [rete-generic-node], page 7 (Lisp file)

`inactivate-children` *NODE OBJECT* [Generic Function]
Package [exil-rete], page 19
Methods

`inactivate-children` (*NODE* `node`) *OBJECT* [Method]
Source [rete-generic-node], page 7 (Lisp file)

`includes-p` *FACT TOKEN* [Generic Function]
Package [exil-rete], page 19
Methods

`includes-p` (*INCLUDED-TOKEN* `token`) (*TOKEN* `token`) [Method]
Source [tokens], page 7 (Lisp file)

`includes-p` (*FACT* `fact`) (*TOKEN* `token`) [Method]
Source [tokens], page 7 (Lisp file)

`initialize-network` *NODE* **&optional** *TEMPLATE-NAME* [Generic Function]
Package [exil-rete], page 19
Methods

`initialize-network` (*NODE* `alpha-top-node`) **&optional** *TEMPLATE-NAME* [Method]
Source [rete-alpha-part], page 8 (Lisp file)

`is-watcher` *WATCHER* [Generic Function]
Package [exil-env], page 24
Methods

`is-watcher` (*WATCHER* `symbol`) [Method]
Source [environment], page 12 (Lisp file)

`items` *OBJECT* [Generic Function]
`(setf items)` *NEW-VALUE OBJECT* [Generic Function]
Package [exil-rete], page 19
Methods

`items` (*MEMORY-NODE* `memory-node`) [Method]
automatically generated reader method
Source [rete-generic-node], page 7 (Lisp file)

`(setf items)` *NEW-VALUE* (*MEMORY-NODE* `memory-node`) [Method]
automatically generated writer method

Source [rete-generic-node], page 7 (Lisp file)

`match-equal-p` *MATCH1 MATCH2* [Generic Function]

Package [exil-env], page 24

Methods

`match-equal-p` (*MATCH1* `match`) (*MATCH2* `match`) [Method]

Source [matches], page 11 (Lisp file)

`match-rule` *OBJECT* [Generic Function]

Package [exil-env], page 24

Methods

`match-rule` (*MATCH* `match`) [Method]
automatically generated reader method

Source [matches], page 11 (Lisp file)

`match-token` *OBJECT* [Generic Function]

Package [exil-env], page 24

Methods

`match-token` (*MATCH* `match`) [Method]
automatically generated reader method

Source [matches], page 11 (Lisp file)

`memory` *OBJECT* [Generic Function]
`(setf memory)` *NEW-VALUE OBJECT* [Generic Function]

Package [exil-rete], page 19

Methods

`memory` (*ALPHA-TEST-NODE* `alpha-test-node`) [Method]
automatically generated reader method

Source [rete-alpha-part], page 8 (Lisp file)

`(setf memory)` *NEW-VALUE* (*ALPHA-TEST-NODE* `alpha-test-node`) [Method]
automatically generated writer method

Source [rete-alpha-part], page 8 (Lisp file)

`modify%` *FACT-SPEC MOD-LIST* [Generic Function]

Package [exil], page 26

Methods

`modify%` (*FACT-SPEC* `integer`) *MOD-LIST* [Method]

Source [export], page 13 (Lisp file)

`modify%` (*FACT-SPEC* `list`) *MOD-LIST* [Method]

Source [export], page 13 (Lisp file)

`negative-wmes` *OBJECT* [Generic Function]
`(setf negative-wmes)` *NEW-VALUE OBJECT* [Generic Function]

Package [exil-rete], page 19

Methods

`negative-wmes` (*TOKEN* `token`) [Method]
automatically generated reader method

Source [tokens], page 7 (Lisp file)

`(setf negative-wmes)` *NEW-VALUE* (*TOKEN* `token`) [Method]
automatically generated writer method

Source [tokens], page 7 (Lisp file)

`networks` *OBJECT* [Generic Function]
`(setf networks)` *NEW-VALUE OBJECT* [Generic Function]

Package [exil-rete], page 19

Methods

`networks` (*ALPHA-TOP-NODE* `alpha-top-node`) [Method]
automatically generated reader method

Source [rete-alpha-part], page 8 (Lisp file)

`(setf networks)` *NEW-VALUE* (*ALPHA-TOP-NODE* `alpha-top-node`) [Method]
automatically generated writer method

Source [rete-alpha-part], page 8 (Lisp file)

`newer-than` *MATCH1 MATCH2* [Generic Function]

Package [exil-env], page 24

Methods

`newer-than` (*MATCH1* `match`) (*MATCH2* `match`) [Method]

Source [strategies], page 11 (Lisp file)

`node-equal-p` *NODE1 NODE2* [Generic Function]

Package [exil-rete], page 19

Source [rete-generic-node], page 7 (Lisp file)

Methods

`node-equal-p` (*NODE1* `beta-join-node`) (*NODE2* `beta-join-node`) [Method]

Source [rete-beta-part], page 9 (Lisp file)

`node-equal-p` (*NODE1* `alpha-test-node`) (*NODE2* `alpha-test-node`) [Method]

Source [rete-alpha-part], page 8 (Lisp file)

`node-equal-p` (*NODE1* `(eql nil)`) (*NODE2* `(eql nil)`) [Method]

Source [rete-generic-node], page 7 (Lisp file)

`node-equal-p` (*NODE1* `node`) (*NODE2* `node`) [Method]

Source [rete-generic-node], page 7 (Lisp file)

`parent` *OBJECT* [Generic Function]
`(setf parent)` *NEW-VALUE OBJECT* [Generic Function]

Package [exil-rete], page 19

Methods

`parent` (*BETA-NODE* `beta-node`) [Method]
automatically generated reader method

Source [rete-beta-part], page 9 (Lisp file)

`(setf parent)` *NEW-VALUE* (*BETA-NODE* `beta-node`) [Method]
automatically generated writer method

Source [rete-beta-part], page 9 (Lisp file)

`parent` (*TOKEN* `token`) [Method]
automatically generated reader method

Source [tokens], page 7 (Lisp file)

`perform-join-test` *TEST TOKEN WME* [Generic Function]

Package [exil-rete], page 19

Methods

`perform-join-test` (*TEST* `test`) (*TOKEN* `token`) (*WME* `fact`) [Method]

Source [rete-beta-part], page 9 (Lisp file)

`perform-join-tests` *TESTS TOKEN WME* [Generic Function]

Package [exil-rete], page 19

Methods

`perform-join-tests` (*TESTS* `list`) (*TOKEN* `token`) (*WME* `fact`) [Method]

Source [rete-beta-part], page 9 (Lisp file)

`previous-condition` *OBJECT* [Generic Function]

Package [exil-rete], page 19

Methods

`previous-condition` (*TEST* `test`) [Method]
tells, how many conditions back i must go

Source [rete-beta-part], page 9 (Lisp file)

`previous-field` *OBJECT* [Generic Function]

Package [exil-rete], page 19

Methods

`previous-field` (*TEST* `test`) [Method]
automatically generated reader method

Source [rete-beta-part], page 9 (Lisp file)

`previous-wme` *TOKEN* **&optional** *N* [Generic Function]

Package [exil-rete], page 19

Methods

`previous-wme` (*TOKEN* `token`) **&optional** *N* [Method]

gives wme from token n wmes back

Source [tokens], page 7 (Lisp file)

`productions` *OBJECT* [Generic Function]
`(setf productions)` *NEW-VALUE OBJECT* [Generic Function]

Package [exil-rete], page 19

Methods

`productions` (*BETA-MEMORY-NODE* `beta-memory-node`) [Method]

automatically generated reader method

Source [rete-beta-part], page 9 (Lisp file)

`(setf productions)` *NEW-VALUE* (*BETA-MEMORY-NODE* `beta-memory-node`) [Method]

automatically generated writer method

Source [rete-beta-part], page 9 (Lisp file)

`remove-matches` *RULE* [Generic Function]

Package [exil-env], page 24

Methods

`remove-matches` *RULE* [Method]

Source [environment], page 12 (Lisp file)

`simple-fact-key-name` *OBJECT* [Generic Function]

Package [exil-rete], page 19

Methods

`simple-fact-key-name` (*ALPHA-TOP-NODE* `alpha-top-node`) [Method]

automatically generated reader method

Source [rete-alpha-part], page 8 (Lisp file)

`simpler-than` *RULE1 RULE2* [Generic Function]

Package [exil-env], page 24

Methods

`simpler-than` (*MATCH1* `match`) (*MATCH2* `match`) [Method]

Source [strategies], page 11 (Lisp file)

`simpler-than` (*RULE1* `rule`) (*RULE2* `rule`) [Method]

Source [strategies], page 11 (Lisp file)

`strategies` [Generic Function]

Package [exil-env], page 24

Methods

`strategies` [Method]

Source [environment], page 12 (Lisp file)

`test` *NODE WME* [Generic Function]

provides testing part of alpha-test-node activation

Package [exil-rete], page 19

Source [rete-alpha-part], page 8 (Lisp file)

Methods

`test` (*NODE* `template-fact-test-node`) (*WME* `template-fact`) [Method]

Source [rete-alpha-part], page 8 (Lisp file)

`test` (*NODE* `simple-fact-test-node`) (*WME* `simple-fact`) [Method]

Source [rete-alpha-part], page 8 (Lisp file)

`test` (*NODE* `alpha-test-node`) (*WME* `fact`) [Method]

Source [rete-alpha-part], page 8 (Lisp file)

`test-equal-p` *TEST1 TEST2* [Generic Function]

Package [exil-rete], page 19

Methods

`test-equal-p` (*TEST1* `test`) (*TEST2* `test`) [Method]

Source [rete-beta-part], page 9 (Lisp file)

`tested-field` *OBJECT* [Generic Function]

Package [exil-rete], page 19

Methods

`tested-field` (*ALPHA-TEST-NODE* `alpha-test-node`) [Method]

automatically generated reader method

Source [rete-alpha-part], page 8 (Lisp file)

`tests` *OBJECT* [Generic Function]

`(setf tests)` *NEW-VALUE OBJECT* [Generic Function]

Package [exil-rete], page 19

Methods

`tests` (*BETA-JOIN-NODE* `beta-join-node`) [Method]

automatically generated reader method

Source [rete-beta-part], page 9 (Lisp file)

`(setf tests)` *NEW-VALUE* (*BETA-JOIN-NODE* `beta-join-node`) [Method]

automatically generated writer method

Source [rete-beta-part], page 9 (Lisp file)

`tests-equal-p` *TEST-LIST1 TEST-LIST2* [Generic Function]

Package [exil-rete], page 19

Methods

`tests-equal-p` (*TEST-LIST1* `list`) (*TEST-LIST2* `list`) [Method]

Source [rete-beta-part], page 9 (Lisp file)

`timestamp` *OBJECT* [Generic Function]

Package [exil-env], page 24

Methods

`timestamp` (*MATCH* `match`) [Method]

automatically generated reader method

Source [matches], page 11 (Lisp file)

`tmpl-object-equal-p` *OBJECT1 OBJECT2* [Generic Function]

Package [exil-core], page 16

Methods

`tmpl-object-equal-p` (*OBJECT1* `template-object`) (*OBJECT2* `template-object`) [Method]

template-object equality predicate

Source [templates], page 4 (Lisp file)

`tmpl-object-slot-value` *OBJECT SLOT-NAME* [Generic Function]
`(setf tmpl-object-slot-value)` *VAL OBJECT SLOT-NAME* [Generic Function]

Package [exil-core], page 16

Methods

`tmpl-object-slot-value` (*OBJECT* `template-object`) *SLOT-NAME* [Method]

`(setf tmpl-object-slot-value)` *VAL* (*OBJECT* `template-object`) *SLOT-NAME* [Method]

get the template-object slot value according to the slot name

Source [templates], page 4 (Lisp file)

`tmpl-pattern-slot-value` *PATTERN SLOT-NAME* [Generic Function]

Package [exil-core], page 16

Methods

`tmpl-pattern-slot-value` (*PATTERN* `template-pattern`) *SLOT-NAME* [Method]

Source [patterns], page 6 (Lisp file)

`token` *WME* **&optional** *PARENT* [Generic Function]

Package [exil-rete], page 19

Methods

`token` *WME* **&optional** *PARENT* [Method]

Source [tokens], page 7 (Lisp file)

`value` *OBJECT* [Generic Function]

Package [exil-rete], page 19

Methods

`value` (*ALPHA-TEST-NODE* `alpha-test-node`) [Method]
automatically generated reader method

Source [rete-alpha-part], page 8 (Lisp file)

`variable-bindings` *PATTERN FACT* [Generic Function]

Package [exil-env], page 24

Methods

`variable-bindings` (*PATTERN* `template-pattern`) (*FACT* `template-fact`) [Method]

Source [activations], page 11 (Lisp file)

`variable-bindings` (*PATTERN* `simple-pattern`) (*FACT* `simple-fact`) [Method]

Source [activations], page 11 (Lisp file)

`watchers` [Generic Function]

Package [exil-env], page 24

Methods

`watchers` [Method]

Source [environment], page 12 (Lisp file)

`wme` *OBJECT* [Generic Function]

Package [exil-rete], page 19

Methods

`wme` (*TOKEN* `token`) [Method]
automatically generated reader method

Source [tokens], page 7 (Lisp file)

4.2.5 Classes

`alpha-memory-node` [Class]

Package [exil-rete], page 19

Source [rete-alpha-part], page 8 (Lisp file)

Direct superclasses

- [memory-node], page 74 (class)
- [alpha-node], page 67 (class)

Direct methods

- [find/create-neg-node], page 58 (method)
- [find/create-join-node], page 57 (method)
- [inactivate], page 60 (method)
- [activate], page 53 (method)

`alpha-node` [Class]

Package [exil-rete], page 19

Source [rete-alpha-part], page 8 (Lisp file)

Direct superclasses
[node], page 74 (class)

Direct subclasses
- [alpha-test-node], page 68 (class)
- [simple-fact-alpha-node], page 75 (class)
- [template-fact-alpha-node], page 76 (class)
- [alpha-subtop-node], page 68 (class)
- [alpha-top-node], page 69 (class)
- [alpha-memory-node], page 67 (class)

Direct methods
[find-test-node], page 57 (method)

`alpha-subtop-node` [Class]

Package [exil-rete], page 19

Source [rete-alpha-part], page 8 (Lisp file)

Direct superclasses
[alpha-node], page 67 (class)

Direct subclasses
- [simple-fact-subtop-node], page 75 (class)
- [template-fact-subtop-node], page 76 (class)

Direct methods
[activate], page 54 (method)

`alpha-test-node` [Class]

Package [exil-rete], page 19

Source [rete-alpha-part], page 8 (Lisp file)

Direct superclasses
[alpha-node], page 67 (class)

Direct subclasses
- [simple-fact-test-node], page 76 (class)
- [template-fact-test-node], page 76 (class)

Direct methods
- [inactivate], page 60 (method)
- [activate], page 54 (method)
- [activate-memory], page 54 (method)
- [activate-children], page 54 (method)
- [test], page 65 (method)
- `print-object`
- [node-equal-p], page 62 (method)
- `memory`
- [memory], page 61 (method)
- [value], page 67 (method)
- [tested-field], page 65 (method)

Direct slots

`tested-field` [Slot]

Initargs `:tested-field`

Initform `(error "tested-field slot has to be specified")`

Readers [tested-field], page 65 (generic function)

`desired-value` [Slot]

Initargs `:value`

Initform `(error "desired-value slot has to be specified")`

Readers [value], page 66 (generic function)

`alpha-memory` [Slot]

Initargs `:memory`

Readers [memory], page 61 (generic function)

Writers [(setf memory)], page 61 (generic function)

`alpha-top-node` [Class]

Package [exil-rete], page 19

Source [rete-alpha-part], page 8 (Lisp file)

Direct superclasses

[alpha-node], page 67 (class)

Direct methods

- [inactivate], page 60 (method)
- [activate], page 53 (method)
- `initialize-instance`
- [get/initialize-network], page 59 (method)
- [initialize-network], page 60 (method)
- `get-network`
- [get-network], page 59 (method)
- [simple-fact-key-name], page 64 (method)
- `networks`
- [networks], page 62 (method)

Direct slots

`dataflow-networks` [Slot]

Initform `(make-hash-table)`

Readers [networks], page 62 (generic function)

Writers [(setf networks)], page 62 (generic function)

`simple-fact-key-name` [Slot]

Initform `(gensym "simple-fact")`

Readers [simple-fact-key-name], page 64 (generic function)

`beta-join-node` [Class]

Package [exil-rete], page 19

Source [rete-beta-part], page 9 (Lisp file)

Direct superclasses

[beta-node], page 71 (class)

Direct subclasses

[beta-negative-node], page 71 (class)

Direct methods

- [node-equal-p], page 62 (method)
- [activate], page 53 (method)
- [activate], page 53 (method)
- [beta-memory], page 55 (method)
- `initialize-instance`
- `tests`
- [tests], page 65 (method)
- [alpha-memory], page 55 (method)

Direct slots

`alpha-memory` [Slot]

Initargs `:alpha-memory`

Initform `(error "alpha-memory slot has to be specified")`

Readers [alpha-memory], page 55 (generic function)

`tests` [Slot]

Initargs `:tests`

Readers [tests], page 65 (generic function)

Writers [(setf tests)], page 65 (generic function)

`beta-memory-node` [Class]

Package [exil-rete], page 19

Source [rete-beta-part], page 9 (Lisp file)

Direct superclasses

- [memory-node], page 74 (class)
- [beta-node], page 71 (class)

Direct subclasses

[beta-top-node], page 71 (class)

Direct methods

- [find/create-neg-node], page 58 (method)
- [find/create-join-node], page 57 (method)
- `print-object`
- [delete-production], page 57 (method)
- [add-production], page 55 (method)
- [inactivate], page 59 (method)

- [inactivate], page 59 (method)
- [broken-match], page 55 (method)
- [activate], page 53 (method)
- [complete-match], page 56 (method)
- productions
- [productions], page 64 (method)

Direct slots

productions [Slot]

Readers [productions], page 64 (generic function)

Writers [(setf productions)], page 64 (generic function)

beta-negative-node [Class]

Package [exil-rete], page 19

Source [rete-beta-part], page 9 (Lisp file)

Direct superclasses

- [memory-node], page 74 (class)
- [beta-join-node], page 70 (class)

Direct methods

- [inactivate], page 59 (method)
- [activate], page 53 (method)
- [activate], page 53 (method)
- [get-bad-wmes], page 58 (method)

beta-node [Class]

Package [exil-rete], page 19

Source [rete-beta-part], page 9 (Lisp file)

Direct superclasses

[node], page 74 (class)

Direct subclasses

- [beta-memory-node], page 70 (class)
- [beta-join-node], page 70 (class)

Direct methods

- parent
- [parent], page 63 (method)

Direct slots

parent [Slot]

Initargs :parent

Readers [parent], page 63 (generic function)

Writers [(setf parent)], page 63 (generic function)

beta-top-node [Class]

Package [exil-rete], page 19

Source [rete-beta-part], page 9 (Lisp file)

Direct superclasses
[beta-memory-node], page 70 (class)

Direct slots

`items` [Slot]
Initform `(list (make-instance 'exil-rete::empty-token))`

`described-object` [Class]
Package [exil-rete], page 19
Source [rete-generic-node], page 7 (Lisp file)
Direct superclasses
`standard-object`
Direct subclasses
[node], page 74 (class)
Direct methods
- `print-object`
- `description`
- [description], page 57 (method)

Direct slots

`description` [Slot]
Initargs `:description`
Initform `""`
Readers [description], page 57 (generic function)
Writers [(setf description)], page 57 (generic function)

`empty-token` [Class]
Package [exil-rete], page 19
Source [tokens], page 7 (Lisp file)
Direct superclasses
[token], page 78 (class)
Direct methods
[token-equal-p], page 43 (method)
Direct slots

`wme` [Slot]

`exil-environment` [Class]
Package [exil-env], page 24
Source [environment], page 12 (Lisp file)
Direct superclasses
`standard-object`
Direct slots

`facts` [Slot]

`fact-groups` [Slot]

`templates` [Slot]

Initform `(make-hash-table :test 'equalp)`

`rules` [Slot]

Initform `(make-hash-table :test 'equalp)`

`rete` [Slot]

Initform `(exil-rete:make-rete)`

`agenda` [Slot]

`strategies` [Slot]

Initform
```
`((exil-env::default ,@#'exil-env::depth-strategy)
  (exil-env::depth-strategy ,@#'exil-env::depth-strategy)
  (exil-env::breadth-strategy ,@#'exil-env::breadth-strategy)
  (exil-env::simplicity-strategy ,@#'exil-env::simplicity-strategy)
  (exil-env::complexity-strategy ,@#'exil-env::complexity-strategy))
```

`current-strategy-name` [Slot]

Initform `'exil-env::default`

`watchers` [Slot]

Initform `'((:facts) (:rules) (:activations))`

`match` [Class]

Package [exil-env], page 24

Source [matches], page 11 (Lisp file)

Direct superclasses

`standard-object`

Direct methods

- [simpler-than], page 64 (method)
- [newer-than], page 62 (method)
- [activate-rule], page 35 (method)
- `print-object`
- [match-equal-p], page 61 (method)
- [timestamp], page 66 (method)
- [match-token], page 61 (method)
- [match-rule], page 61 (method)

Direct slots

`rule` [Slot]

Initargs `:rule`

Initform `(error "match rule has to be specified")`

Readers [match-rule], page 61 (generic function)

`token` [Slot]

Initargs `:token`

Initform `(error "match token has to be specified")`

Readers [match-token], page 61 (generic function)

`timestamp` [Slot]

Initargs `:timestamp`

Initform `(get-internal-real-time)`

Readers [timestamp], page 66 (generic function)

`memory-node` [Class]

Package [exil-rete], page 19

Source [rete-generic-node], page 7 (Lisp file)

Direct superclasses

[node], page 74 (class)

Direct subclasses

- [alpha-memory-node], page 67 (class)
- [beta-memory-node], page 70 (class)
- [beta-negative-node], page 71 (class)

Direct methods

- [add-item], page 54 (method)
- `items`
- [items], page 60 (method)

Direct slots

`items` [Slot]

Readers [items], page 60 (generic function)

Writers [(setf items)], page 60 (generic function)

`node` [Class]

Package [exil-rete], page 19

Source [rete-generic-node], page 7 (Lisp file)

Direct superclasses

[described-object], page 72 (class)

Direct subclasses

- [memory-node], page 74 (class)
- [alpha-node], page 67 (class)
- [beta-node], page 71 (class)

Direct methods

- [inactivate], page 60 (method)
- [inactivate-children], page 60 (method)
- [activate-children], page 54 (method)
- [add-children], page 54 (method)
- [add-child], page 54 (method)
- [node-equal-p], page 63 (method)
- `children`
- [children], page 56 (method)

Direct slots

`children` [Slot]

Readers [children], page 55 (generic function)

Writers [(setf children)], page 55 (generic function)

`rete` [Class]

Package [exil-rete], page 19

Source [rete-net-creation], page 10 (Lisp file)

Direct superclasses

`standard-object`

Direct methods

- `beta-top-node`
- [beta-top-node], page 55 (method)
- [alpha-top-node], page 55 (method)

Direct slots

`alpha-top-node` [Slot]

Initform `(make-instance 'exil-rete::alpha-top-node)`

Readers [alpha-top-node], page 55 (generic function)

`beta-top-node` [Slot]

Initform `(make-instance 'exil-rete::beta-top-node)`

Readers [beta-top-node], page 55 (generic function)

Writers [(setf beta-top-node)], page 55 (generic function)

`simple-fact-alpha-node` [Class]

Package [exil-rete], page 19

Source [rete-alpha-part], page 8 (Lisp file)

Direct superclasses

[alpha-node], page 67 (class)

Direct subclasses

- [simple-fact-test-node], page 76 (class)
- [simple-fact-subtop-node], page 75 (class)

Direct methods

[find/create-test-node], page 58 (method)

`simple-fact-subtop-node` [Class]

Package [exil-rete], page 19

Source [rete-alpha-part], page 8 (Lisp file)

Direct superclasses

- [simple-fact-alpha-node], page 75 (class)
- [alpha-subtop-node], page 68 (class)

Direct methods

[create-alpha-net%], page 56 (method)

`simple-fact-test-node` [Class]

Package [exil-rete], page 19

Source [rete-alpha-part], page 8 (Lisp file)

Direct superclasses

- [simple-fact-alpha-node], page 75 (class)
- [alpha-test-node], page 68 (class)

Direct methods

[test], page 65 (method)

`template-fact-alpha-node` [Class]

Package [exil-rete], page 19

Source [rete-alpha-part], page 8 (Lisp file)

Direct superclasses

[alpha-node], page 67 (class)

Direct subclasses

- [template-fact-test-node], page 76 (class)
- [template-fact-subtop-node], page 76 (class)

Direct methods

[find/create-test-node], page 58 (method)

`template-fact-subtop-node` [Class]

Package [exil-rete], page 19

Source [rete-alpha-part], page 8 (Lisp file)

Direct superclasses

- [template-fact-alpha-node], page 76 (class)
- [alpha-subtop-node], page 68 (class)

Direct methods

[create-alpha-net%], page 56 (method)

`template-fact-test-node` [Class]

Package [exil-rete], page 19

Source [rete-alpha-part], page 8 (Lisp file)

Direct superclasses

- [template-fact-alpha-node], page 76 (class)
- [alpha-test-node], page 68 (class)

Direct methods

[test], page 65 (method)

`template-object` [Class]

Package [exil-core], page 16

Source [templates], page 4 (Lisp file)

Direct superclasses

`standard-object`

Direct subclasses

- [template-fact], page 48 (class)

- [template-pattern], page 48 (class)

Direct methods

- [atom-position], page 36 (method)
- [find-atom], page 39 (method)
- print-object
- [tmpl-object-equal-p], page 66 (method)
- tmpl-object-slot-value
- [has-slot-p], page 39 (method)
- [tmpl-object-slot-value], page 66 (method)
- [slots], page 42 (method)
- [tmpl-name], page 43 (method)

Direct slots

template-name [Slot]

Initargs :tmpl-name

Readers [tmpl-name], page 43 (generic function)

slot-default [Slot]

Allocation :class

slots [Slot]

Initargs :slots

Readers [slots], page 42 (generic function)

test [Class]

Package [exil-rete], page 19

Source [rete-beta-part], page 9 (Lisp file)

Direct superclasses

standard-object

Direct methods

- [perform-join-test], page 63 (method)
- [test-equal-p], page 65 (method)
- print-object
- [previous-field], page 63 (method)
- [previous-condition], page 63 (method)
- [current-field], page 56 (method)

Direct slots

current-field-to-test [Slot]

Initargs :current-field

Initform (error "current-field slot has to be specified")

Readers [current-field], page 56 (generic function)

previous-condition-number [Slot]

tells, how many conditions back i must go

Initargs :previous-condition

Initform `0`

Readers [previous-condition], page 63 (generic function)

`previous-field-to-test` [Slot]

Initargs `:previous-field`

Initform `(error "previous-field slot has to be specified")`

Readers [previous-field], page 63 (generic function)

`token` [Class]

Package [exil-rete], page 19

Source [tokens], page 7 (Lisp file)

Direct superclasses

`standard-object`

Direct subclasses

[empty-token], page 72 (class)

Direct methods

- [activate], page 53 (method)
- [get-bad-wmes], page 58 (method)
- [activate], page 53 (method)
- [perform-join-tests], page 63 (method)
- [perform-join-test], page 63 (method)
- [inactivate], page 59 (method)
- [broken-match], page 55 (method)
- [activate], page 53 (method)
- [complete-match], page 56 (method)
- [token->list], page 43 (method)
- `print-object`
- [includes-p], page 60 (method)
- [token-equal-p], page 44 (method)
- [includes-p], page 60 (method)
- [previous-wme], page 64 (method)
- `negative-wmes`
- [negative-wmes], page 62 (method)
- [wme], page 67 (method)
- [parent], page 63 (method)

Direct slots

`parent` [Slot]

Initargs `:parent`

Readers [parent], page 63 (generic function)

`wme` [Slot]

Initargs `:wme`

Initform `(error "wme slot has to be specified")`

Readers [wme], page 67 (generic function)

`negative-wmes` [Slot]

Readers [`negative-wmes`], page 62 (generic function)

Writers [`(setf negative-wmes)`], page 62 (generic function)

Appendix A Indexes

A.1 Concepts

A.2 Functions

G

H

I

M

N

P

R

S

T

U

V

W

A.3 Variables

*

A

B

C

D

F

I

M

N

P

R

S

T

W

A.4 Data types

A

B

C

D

E

F

M

N

P

R

S

T