

# The exil Reference Manual

---

EXpert system In Lisp, version 0.01

Jakub Kalab <[jakubkalab@gmail.com](mailto: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

<b>1</b>	<b>System .....</b>	<b>1</b>
<b>2</b>	<b>Files .....</b>	<b>3</b>
2.1	Lisp.....	3
2.1.1	exil.asd.....	3
2.1.2	packages.lisp.....	3
2.1.3	utils.lisp .....	3
2.1.4	templates.lisp.....	4
2.1.5	facts.lisp.....	5
2.1.6	patterns.lisp.....	6
2.1.7	rules.lisp .....	6
2.1.8	tokens.lisp.....	7
2.1.9	rete-generic-node.lisp.....	7
2.1.10	rete-alpha-part.lisp.....	8
2.1.11	rete-beta-part.lisp.....	9
2.1.12	rete-net-creation.lisp.....	10
2.1.13	matches.lisp.....	11
2.1.14	activations.lisp.....	11
2.1.15	strategies.lisp.....	11
2.1.16	environment.lisp.....	12
2.1.17	export.lisp.....	13
<b>3</b>	<b>Packages .....</b>	<b>15</b>
3.1	exil-system.....	15
3.2	exil-utils.....	15
3.3	exil-core .....	16
3.4	exil-rete .....	19
3.5	exil-env .....	24
3.6	exil.....	26
3.7	exil-user .....	27
<b>4</b>	<b>Definitions .....</b>	<b>29</b>
4.1	Exported definitions.....	29
4.1.1	Macros.....	29
4.1.2	Functions .....	31
4.1.3	Generic functions.....	35
4.1.4	Classes .....	45
4.2	Internal definitions .....	49
4.2.1	Special variables.....	49
4.2.2	Macros.....	49
4.2.3	Functions .....	50
4.2.4	Generic functions.....	53
4.2.5	Classes .....	67

<b>Appendix A</b>	<b>Indexes</b>	<b>81</b>
A.1	Concepts	81
A.2	Functions	82
A.3	Variables	88
A.4	Data types	90

# 1 System

**Name**        `exil`

**Author**     Jakub Kalab <[jakubkalab@gmail.com](mailto:jakubkalab@gmail.com)>

**Maintainer**  
              Jakub Kalab <[jakubkalab@gmail.com](mailto:jakubkalab@gmail.com)>

**License**     BSD

**Description**  
              EXpert system In Lisp

**Long Description**

**Version**     0.1

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

**Components**

- [[packages](#)], page 3 (Lisp file)
- [[utils](#)], page 3 (Lisp file)
- [[templates](#)], page 4 (Lisp file)
- [[facts](#)], page 5 (Lisp file)
- [[patterns](#)], page 6 (Lisp file)
- [[rules](#)], page 6 (Lisp file)
- [[tokens](#)], page 7 (Lisp file)
- [[rete-generic-node](#)], page 7 (Lisp file)
- [[rete-alpha-part](#)], page 8 (Lisp file)
- [[rete-beta-part](#)], page 9 (Lisp file)
- [[rete-net-creation](#)], page 10 (Lisp file)
- [[matches](#)], page 11 (Lisp file)
- [[activations](#)], page 11 (Lisp file)
- [[strategies](#)], page 11 (Lisp file)
- [[environment](#)], page 12 (Lisp file)
- [[export](#)], page 13 (Lisp file)



## 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)

- [\[exil-equal-p\]](#), 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 (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)

#### 2.1.4 templates.lisp

##### Dependency

`utils`

**Parent**     [\[exil\]](#), page 1 (system)

**Location**   `templates.lisp`

##### Exported Definitions

- [\[atom-position\]](#), page 36 (method)
- [\[find-atom\]](#), page 39 (method)
- [\[has-slot-p\]](#), page 39 (method)
- [\[make-template\]](#), page 33 (function)
- [\[name\]](#), page 40 (method)
- [\[slots\]](#), page 42 (method)
- [\[slots\]](#), page 42 (method)



- `[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)

- `[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 (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)

Packages are listed by definition order.

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

- asdf
- common-lisp

## Source [packages], page 3 (Lisp file)

### Used By List

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

- [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)

- `[conditions]`, page 36 (generic function)
- `[conditions]`, page 37 (method)
- `[constant-test]`, page 32 (function)
- `[copy-fact]`, page 37 (generic function)
- `[copy-fact]`, page 37 (method)
- `[fact]`, page 38 (generic function)
- `[fact]`, page 38 (method)
- `[fact]`, page 45 (class)
- `[fact-description]`, page 38 (generic function)
- `[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 (generic function)
- `[find-atom]`, page 39 (method)
- `[find-atom]`, page 39 (method)
- `[find-atom]`, page 39 (method)
- `[has-slot-p]`, page 39 (generic function)
- `[has-slot-p]`, page 39 (method)
- `[make-fact]`, page 33 (function)
- `[make-pattern]`, page 33 (function)
- `[make-rule]`, page 39 (generic function)
- `[make-rule]`, page 39 (method)
- `[make-template]`, page 33 (function)
- `[match-var]`, page 40 (generic function)
- `[match-var]`, page 40 (method)
- `[(setf match-var)]`, page 40 (method)
- `[(setf match-var)]`, page 40 (generic function)
- `[name]`, page 40 (generic function)
- `[name]`, page 40 (method)
- `[name]`, page 40 (method)
- `[negated-p]`, page 40 (generic function)
- `[negated-p]`, page 40 (method)
- `[(setf negated-p)]`, page 40 (method)
- `[(setf negated-p)]`, page 40 (generic function)
- `[pattern]`, page 40 (generic function)
- `[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)
- [\[rule\]](#), page 46 (class)
- [\[rule-equal-p\]](#), page 42 (generic function)
- [\[rule-equal-p\]](#), page 42 (method)
- [\[simple-fact\]](#), page 47 (class)
- [\[simple-pattern\]](#), page 47 (class)
- [\[slots\]](#), page 42 (generic function)
- [\[slots\]](#), page 42 (method)
- [\[slots\]](#), page 42 (method)
- [\[template\]](#), page 48 (class)
- [\[template-fact\]](#), page 48 (class)
- [\[template-pattern\]](#), page 48 (class)
- [\[tmpl-fact-slot-value\]](#), page 43 (generic function)
- [\[tmpl-fact-slot-value\]](#), page 43 (method)
- [\[\(setf tmpl-fact-slot-value\)\]](#), page 43 (method)
- [\[\(setf tmpl-fact-slot-value\)\]](#), page 43 (generic function)
- [\[tmpl-name\]](#), page 43 (generic function)
- [\[tmpl-name\]](#), page 43 (method)
- [\[var-or-equal-p\]](#), page 35 (function)
- [\[variable-p\]](#), page 35 (function)

### Internal Definitions

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

### 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)

- `[add-children]`, page 54 (generic function)
- `[add-children]`, page 54 (method)
- `[add-item]`, page 54 (generic function)
- `[add-item]`, page 54 (method)
- `[add-production]`, page 54 (generic function)
- `[add-production]`, page 55 (method)
- `[alpha-memory]`, page 55 (generic function)
- `[alpha-memory]`, page 55 (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 55 (generic function)
- `[alpha-top-node]`, page 55 (method)
- `[alpha-top-node]`, page 69 (class)
- `[beta-join-node]`, page 70 (class)
- `[beta-memory]`, page 55 (generic function)
- `[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 55 (generic function)
- `[beta-top-node]`, page 55 (method)
- `[(setf beta-top-node)]`, page 55 (method)
- `[(setf beta-top-node)]`, page 55 (generic function)
- `[beta-top-node]`, page 71 (class)
- `[broken-match]`, page 55 (generic function)
- `[broken-match]`, page 55 (method)
- `[children]`, page 55 (generic function)
- `[children]`, page 56 (method)
- `[(setf children)]`, page 56 (method)
- `[(setf children)]`, page 55 (generic function)
- `[complete-match]`, page 56 (generic function)
- `[complete-match]`, page 56 (method)
- `[create-alpha-net]`, page 56 (generic function)
- `[create-alpha-net]`, page 56 (method)
- `[create-alpha-net]`, page 56 (method)
- `[create-alpha-net%]`, page 56 (generic function)
- `[create-alpha-net%]`, page 56 (method)
- `[create-alpha-net%]`, page 56 (method)
- `[current-field]`, page 56 (generic function)
- `[current-field]`, page 56 (method)
- `[delete-production]`, page 57 (generic function)



- `[delete-production]`, page 57 (method)
- `[described-object]`, page 72 (class)
- `[description]`, page 57 (generic function)
- `[description]`, page 57 (method)
- `[(setf description)]`, page 57 (method)
- `[(setf description)]`, page 57 (generic function)
- `[empty-token]`, page 72 (class)
- `[find-atom-in-cond-list%]`, page 51 (function)
- `[find-test-node]`, page 57 (generic function)
- `[find-test-node]`, page 57 (method)
- `[find/create-join-node]`, page 57 (generic function)
- `[find/create-join-node]`, page 57 (method)
- `[find/create-neg-node]`, page 57 (generic function)
- `[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 (generic function)
- `[find/create-test-node%]`, page 58 (method)
- `[get-bad-wmes]`, page 58 (generic function)
- `[get-bad-wmes]`, page 58 (method)
- `[get-intercondition-tests%]`, page 58 (generic function)
- `[get-intercondition-tests%]`, page 58 (method)
- `[get-intercondition-tests%]`, page 58 (method)
- `[get-intracondition-tests%]`, page 58 (generic function)
- `[get-intracondition-tests%]`, page 59 (method)
- `[get-intracondition-tests%]`, page 59 (method)
- `[get-join-tests-from-condition]`, page 59 (generic function)
- `[get-join-tests-from-condition]`, page 59 (method)
- `[get-network]`, page 59 (generic function)
- `[get-network]`, page 59 (method)
- `[(setf get-network)]`, page 59 (method)
- `[(setf get-network)]`, page 59 (generic function)
- `[get/initialize-network]`, page 59 (generic function)
- `[get/initialize-network]`, page 59 (method)
- `[inactivate]`, page 59 (generic function)
- `[inactivate]`, page 59 (method)
- `[inactivate]`, page 59 (method)
- `[inactivate]`, page 59 (method)
- `[inactivate]`, page 60 (method)
- `[inactivate]`, page 60 (method)
- `[inactivate]`, page 60 (method)
- `[inactivate]`, page 60 (method)

- `[inactivate-children]`, page 60 (generic function)
- `[inactivate-children]`, page 60 (method)
- `[includes-p]`, page 60 (generic function)
- `[includes-p]`, page 60 (method)
- `[includes-p]`, page 60 (method)
- `[initialize-network]`, page 60 (generic function)
- `[initialize-network]`, page 60 (method)
- `[items]`, page 60 (generic function)
- `[items]`, page 60 (method)
- `[(setf items)]`, page 61 (method)
- `[(setf items)]`, page 60 (generic function)
- `[make-test]`, page 51 (function)
- `[memory]`, page 61 (generic function)
- `[memory]`, page 61 (method)
- `[(setf memory)]`, page 61 (method)
- `[(setf memory)]`, page 61 (generic function)
- `[memory-node]`, page 74 (class)
- `[negative-wmes]`, page 62 (generic function)
- `[negative-wmes]`, page 62 (method)
- `[(setf negative-wmes)]`, page 62 (method)
- `[(setf negative-wmes)]`, page 62 (generic function)
- `[networks]`, page 62 (generic function)
- `[networks]`, page 62 (method)
- `[(setf networks)]`, page 62 (method)
- `[(setf networks)]`, page 62 (generic function)
- `[node]`, page 74 (class)
- `[node-equal-p]`, page 62 (generic function)
- `[node-equal-p]`, page 62 (method)
- `[node-equal-p]`, page 62 (method)
- `[node-equal-p]`, page 62 (method)
- `[node-equal-p]`, page 63 (method)
- `[parent]`, page 63 (generic function)
- `[parent]`, page 63 (method)
- `[(setf parent)]`, page 63 (method)
- `[parent]`, page 63 (method)
- `[(setf parent)]`, page 63 (generic function)
- `[perform-join-test]`, page 63 (generic function)
- `[perform-join-test]`, page 63 (method)
- `[perform-join-tests]`, page 63 (generic function)
- `[perform-join-tests]`, page 63 (method)
- `[previous-condition]`, page 63 (generic function)
- `[previous-condition]`, page 63 (method)
- `[previous-field]`, page 63 (generic function)

- `[previous-field]`, page 63 (method)
- `[previous-wme]`, page 64 (generic function)
- `[previous-wme]`, page 64 (method)
- `[productions]`, page 64 (generic function)
- `[productions]`, page 64 (method)
- `[(setf productions)]`, page 64 (method)
- `[(setf productions)]`, page 64 (generic function)
- `[rete]`, page 75 (class)
- `[simple-fact-alpha-node]`, page 75 (class)
- `[simple-fact-key-name]`, page 64 (generic function)
- `[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)
- `[test]`, page 77 (class)
- `[test-equal-p]`, page 65 (generic function)
- `[test-equal-p]`, page 65 (method)
- `[tested-field]`, page 65 (generic function)
- `[tested-field]`, page 65 (method)
- `[tests]`, page 65 (generic function)
- `[tests]`, page 65 (method)
- `[(setf tests)]`, page 65 (method)
- `[(setf tests)]`, page 65 (generic function)
- `[tests-equal-p]`, page 65 (generic function)
- `[tests-equal-p]`, page 66 (method)
- `[token]`, page 66 (generic function)
- `[token]`, page 66 (method)
- `[token]`, page 78 (class)
- `[value]`, page 66 (generic function)
- `[value]`, page 67 (method)
- `[wme]`, page 67 (generic function)
- `[wme]`, page 67 (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)

- `[select-activation]`, page 42 (generic function)
- `[select-activation]`, page 42 (method)
- `[set-strategy]`, page 42 (generic function)
- `[set-strategy]`, page 42 (method)
- `[set-watcher]`, page 42 (generic function)
- `[set-watcher]`, page 42 (method)
- `[templates]`, page 43 (generic function)
- `[templates]`, page 43 (method)
- `[unset-watcher]`, page 44 (generic function)
- `[unset-watcher]`, page 44 (method)
- `[unwatch-all]`, page 44 (generic function)
- `[unwatch-all]`, page 44 (method)
- `[watch-all]`, page 44 (generic function)
- `[watch-all]`, page 44 (method)
- `[watched-p]`, page 44 (generic function)
- `[watched-p]`, page 44 (method)

#### Internal Definitions

- `[*current-environment*]`, page 49 (special variable)
- `[*environments*]`, page 49 (special variable)
- `[breadth-strategy]`, page 50 (function)
- `[completely-reset-environment]`, page 50 (function)
- `[complexity-strategy]`, page 51 (function)
- `[current-strategy]`, page 56 (generic function)
- `[current-strategy]`, page 57 (method)
- `[current-strategy-name]`, page 57 (generic function)
- `[current-strategy-name]`, page 57 (method)
- `[defenv]`, page 49 (macro)
- `[depth-strategy]`, page 51 (function)
- `[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)
- `[get-variable-bindings]`, page 51 (function)
- `[is-watcher]`, page 60 (generic function)
- `[is-watcher]`, page 60 (method)
- `[make-match]`, page 51 (function)
- `[match]`, page 73 (class)
- `[match-equal-p]`, page 61 (generic function)
- `[match-equal-p]`, page 61 (method)
- `[match-rule]`, page 61 (generic function)
- `[match-rule]`, page 61 (method)
- `[match-token]`, page 61 (generic function)

- [\[match-token\]](#), 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

<b>assert</b> <i>&amp;rest FACT-SPECS</i>	[Macro]
Add fact into working memory	
<b>Package</b> <i>[exil]</i> , page 26	
<b>Source</b> <i>[export]</i> , page 13 (Lisp file)	
<b>deffacts</b> <i>NAME &amp;body FACT-DESCRIPTIONS</i>	[Macro]
Create group of facts to be asserted after (reset)	
<b>Package</b> <i>[exil]</i> , page 26	
<b>Source</b> <i>[export]</i> , page 13 (Lisp file)	
<b>defrule</b> <i>NAME &amp;body RULE</i>	[Macro]
Define rule	
<b>Package</b> <i>[exil]</i> , page 26	
<b>Source</b> <i>[export]</i> , page 13 (Lisp file)	
<b>defstrategy</b> <i>NAME FUNCTION</i>	[Macro]
Define strategy	
<b>Package</b> <i>[exil]</i> , page 26	
<b>Source</b> <i>[export]</i> , page 13 (Lisp file)	
<b>deftemplate</b> <i>NAME &amp;body SLOTS</i>	[Macro]
<b>Package</b> <i>[exil]</i> , page 26	
<b>Source</b> <i>[export]</i> , page 13 (Lisp file)	
<b>diff-delete</b> <i>ITEM SEQUENCE &amp;key TEST KEY</i>	[Macro]
like delete, but as a second value returns list of deleted items	
<b>Package</b> <i>[exil-utils]</i> , page 15	
<b>Source</b> <i>[utils]</i> , page 3 (Lisp file)	
<b>doplist</b> ( <i>KEY VAL PLIST &amp;optional RETVAL</i> ) <i>&amp;body BODY</i>	[Macro]
<b>Package</b> <i>[exil-utils]</i> , page 15	
<b>Source</b> <i>[utils]</i> , page 3 (Lisp file)	
<b>ext-delete</b> <i>ITEM PLACE &amp;key TEST KEY</i>	[Macro]
like delete, but as a second value returns, whether the list was actually altered	
<b>Package</b> <i>[exil-utils]</i> , page 15	
<b>Source</b> <i>[utils]</i> , page 3 (Lisp file)	

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

<b>assoc-value</b> <i>THE-KEY ALIST &amp;key KEY TEST</i>	[Function]
get value from assoc-list according to the key	
<b>Package</b> [exil-utils], page 15	
<b>Source</b> [utils], page 3 (Lisp file)	
<b>(setf assoc-value)</b> <i>VALUE THE-KEY ALIST &amp;key KEY TEST</i>	[Function]
set value in assoc-list according to the key	
<b>Package</b> [exil-utils], page 15	
<b>Source</b> [utils], page 3 (Lisp file)	
<b>class-slot-value</b> <i>CLASS-NAME SLOT-NAME</i>	[Function]
get a class-slot value from class-name	
<b>Package</b> [exil-utils], page 15	
<b>Source</b> [utils], page 3 (Lisp file)	
<b>clear</b>	[Function]
Delete all facts	
<b>Package</b> [exil], page 26	
<b>Source</b> [export], page 13 (Lisp file)	
<b>constant-test</b> <i>DESIRED-VALUE REAL-VALUE</i>	[Function]
<b>Package</b> [exil-core], page 16	
<b>Source</b> [patterns], page 6 (Lisp file)	
<b>cpl-assoc-val</b> <i>KEY CPL-LIST</i>	[Function]
<b>(setf cpl-assoc-val)</b> <i>NEW-VAL KEY CPL-LIST</i>	[Function]
get value from couple list according to the key	
<b>Package</b> [exil-utils], page 15	
<b>Source</b> [utils], page 3 (Lisp file)	
<b>every-couple</b> <i>PREDICATE LIST</i>	[Function]
<b>Package</b> [exil-utils], page 15	
<b>Source</b> [utils], page 3 (Lisp file)	
<b>facts</b> <i>&amp;optional START-INDEX END-INDEX AT-MOST</i>	[Function]
<b>Package</b> [exil], page 26	
<b>Source</b> [export], page 13 (Lisp file)	
<b>find-fact</b> <i>FACT</i>	[Function]
<b>Package</b> [exil-env], page 24	
<b>Source</b> [environment], page 12 (Lisp file)	
<b>find-rule</b> <i>NAME</i>	[Function]
<b>Package</b> [exil-env], page 24	
<b>Source</b> [environment], page 12 (Lisp file)	

<b>from-keyword</b> <i>KEY</i> <b>&amp;optional</b> <i>PACKAGE</i>	[Function]
get symbol from key	
<b>Package</b> [ <a href="#">exil-utils</a> ], page 15	
<b>Source</b> [ <a href="#">utils</a> ], page 3 (Lisp file)	
<b>halt</b>	[Function]
Stop the inference engine	
<b>Package</b> [ <a href="#">exil</a> ], page 26	
<b>Source</b> [ <a href="#">export</a> ], page 13 (Lisp file)	
<b>intern</b> <i>STRING</i> <b>&amp;optional</b> <i>PACKAGE</i>	[Function]
create symbol from string	
<b>Package</b> [ <a href="#">exil-utils</a> ], page 15	
<b>Source</b> [ <a href="#">utils</a> ], page 3 (Lisp file)	
<b>make-fact</b> <i>FACT-SPEC</i>	[Function]
<b>Package</b> [ <a href="#">exil-core</a> ], page 16	
<b>Source</b> [ <a href="#">facts</a> ], page 5 (Lisp file)	
<b>make-pattern</b> <i>SPECIFICATION</i> <b>&amp;key</b> <i>MATCH-VAR</i>	[Function]
<b>Package</b> [ <a href="#">exil-core</a> ], page 16	
<b>Source</b> [ <a href="#">patterns</a> ], page 6 (Lisp file)	
<b>make-rete</b>	[Function]
<b>Package</b> [ <a href="#">exil-rete</a> ], page 19	
<b>Source</b> [ <a href="#">rete-net-creation</a> ], page 10 (Lisp file)	
<b>make-template</b> <i>NAME SLOTS</i>	[Function]
<b>Package</b> [ <a href="#">exil-core</a> ], page 16	
<b>Source</b> [ <a href="#">templates</a> ], page 4 (Lisp file)	
<b>modify-fact</b> <i>FACT MOD-LIST</i>	[Function]
<b>Package</b> [ <a href="#">exil-env</a> ], page 24	
<b>Source</b> [ <a href="#">environment</a> ], page 12 (Lisp file)	
<b>plistp</b> <i>LIST</i>	[Function]
<b>Package</b> [ <a href="#">exil-utils</a> ], page 15	
<b>Source</b> [ <a href="#">utils</a> ], page 3 (Lisp file)	
<b>rem-fact</b> <i>FACT</i>	[Function]
<b>Package</b> [ <a href="#">exil-env</a> ], page 24	
<b>Source</b> [ <a href="#">environment</a> ], page 12 (Lisp file)	
<b>rem-fact-group</b> <i>NAME</i>	[Function]
<b>Package</b> [ <a href="#">exil-env</a> ], page 24	
<b>Source</b> [ <a href="#">environment</a> ], page 12 (Lisp file)	

<b>rem-rule</b> <i>RULE</i>	[Function]
<b>Package</b> [ <i>exil-env</i> ], page 24	
<b>Source</b> [ <i>environment</i> ], page 12 (Lisp file)	
<b>reset</b>	[Function]
Clear all facts and add all fact groups	
<b>Package</b> [ <i>exil</i> ], page 26	
<b>Source</b> [ <i>export</i> ], page 13 (Lisp file)	
<b>reset-environment</b>	[Function]
<b>Package</b> [ <i>exil-env</i> ], page 24	
<b>Source</b> [ <i>environment</i> ], page 12 (Lisp file)	
<b>reset-facts</b>	[Function]
<b>Package</b> [ <i>exil-env</i> ], page 24	
<b>Source</b> [ <i>environment</i> ], page 12 (Lisp file)	
<b>retract-all</b>	[Function]
<b>Package</b> [ <i>exil</i> ], page 26	
<b>Source</b> [ <i>export</i> ], page 13 (Lisp file)	
<b>run</b>	[Function]
Run the infenece engine	
<b>Package</b> [ <i>exil</i> ], page 26	
<b>Source</b> [ <i>export</i> ], page 13 (Lisp file)	
<b>select</b> <i>LIST INDICES</i>	[Function]
get a list of values from list according to the list of indices	
<b>Package</b> [ <i>exil-utils</i> ], page 15	
<b>Source</b> [ <i>utils</i> ], page 3 (Lisp file)	
<b>step</b>	[Function]
Run inference engine for one turn	
<b>Package</b> [ <i>exil</i> ], page 26	
<b>Source</b> [ <i>export</i> ], page 13 (Lisp file)	
<b>string-append &amp;rest STRINGS</b>	[Function]
append two strings	
<b>Package</b> [ <i>exil-utils</i> ], page 15	
<b>Source</b> [ <i>utils</i> ], page 3 (Lisp file)	
<b>subsets</b> <i>LIST</i>	[Function]
get a list of all subsets of given list	
<b>Package</b> [ <i>exil-utils</i> ], page 15	
<b>Source</b> [ <i>utils</i> ], 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	<i>PRODUCTION TOKEN</i>	[Method]
Source	[environment], page 12 (Lisp file)	
add-strategy	<i>NAME FUNCTION</i>	[Generic Function]
Package	[exil-env], page 24	
Methods		
add-strategy	<i>NAME FUNCTION</i>	[Method]
Source	[environment], page 12 (Lisp file)	
add-wme	<i>FACT &amp;optional RETE</i>	[Generic Function]
Package	[exil-rete], page 19	
Methods		
add-wme	<i>(FACT fact) &amp;optional RETE</i>	[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	<i>OBJECT1 OBJECT2</i>	[Generic Function]
equality predicate for fact atoms		
Package	[exil-core], page 16	
Source	[patterns], page 6 (Lisp file)	
Methods		
atom-equal-p	<i>OBJECT1 OBJECT2</i>	[Method]
Source	[patterns], page 6 (Lisp file)	
atom-position	<i>OBJECT ATOM</i>	[Generic Function]
Package	[exil-core], page 16	
Methods		
atom-position	<i>(PATTERN simple-pattern) ATOM</i>	[Method]
Source	[patterns], page 6 (Lisp file)	
atom-position	<i>(FACT simple-fact) ATOM</i>	[Method]
Source	[facts], page 5 (Lisp file)	
atom-position	<i>(OBJECT template-object) ATOM</i>	[Method]
get the atom position in template-object slots		
Source	[templates], page 4 (Lisp file)	
conditions	<i>OBJECT</i>	[Generic Function]
Package	[exil-core], page 16	
Methods		



<b>conditions</b> ( <i>RULE rule</i> )	[Method]
automatically generated reader method	
<b>Source</b> <a href="#">[rules]</a> , page 6 (Lisp file)	
<b>copy-fact</b> <i>FACT</i>	[Generic Function]
<b>Package</b> <a href="#">[exil-core]</a> , page 16	
<b>Methods</b>	
<b>copy-fact</b> ( <i>FACT fact</i> )	[Method]
<b>Source</b> <a href="#">[facts]</a> , page 5 (Lisp file)	
<b>exil-equal-p</b> <i>OBJ1 OBJ2</i>	[Generic Function]
ExiL default equality predicate	
<b>Package</b> <a href="#">[exil-utils]</a> , page 15	
<b>Source</b> <a href="#">[utils]</a> , page 3 (Lisp file)	
<b>Methods</b>	
<b>exil-equal-p</b> ( <i>OBJ1 cons</i> ) ( <i>OBJ2 cons</i> )	[Method]
<b>Source</b> <a href="#">[utils]</a> , page 3 (Lisp file)	
<b>exil-equal-p</b> ( <i>OBJ1 number</i> ) ( <i>OBJ2 number</i> )	[Method]
<b>Source</b> <a href="#">[utils]</a> , page 3 (Lisp file)	
<b>exil-equal-p</b> ( <i>OBJ1 symbol</i> ) ( <i>OBJ2 symbol</i> )	[Method]
<b>Source</b> <a href="#">[utils]</a> , page 3 (Lisp file)	
<b>exil-equal-p</b> ( <i>OBJ1 string</i> ) ( <i>OBJ2 string</i> )	[Method]
<b>Source</b> <a href="#">[utils]</a> , page 3 (Lisp file)	
<b>exil-equal-p</b> <i>OBJ1 OBJ2</i>	[Method]
<b>Source</b> <a href="#">[utils]</a> , page 3 (Lisp file)	
<b>exil-equal-p</b> ( <i>OBJ1 null</i> ) ( <i>OBJ2 null</i> )	[Method]
<b>Source</b> <a href="#">[utils]</a> , page 3 (Lisp file)	
<b>exil-weak-equal-p</b> <i>OBJ1 OBJ2</i>	[Generic Function]
ExiL default weak equality predicate	
<b>Package</b> <a href="#">[exil-utils]</a> , page 15	
<b>Source</b> <a href="#">[utils]</a> , page 3 (Lisp file)	
<b>Methods</b>	
<b>exil-weak-equal-p</b> ( <i>OBJ1 cons</i> ) ( <i>OBJ2 cons</i> )	[Method]
<b>Source</b> <a href="#">[utils]</a> , page 3 (Lisp file)	
<b>exil-weak-equal-p</b> ( <i>OBJ1 symbol</i> ) ( <i>OBJ2 symbol</i> )	[Method]
<b>Source</b> <a href="#">[utils]</a> , page 3 (Lisp file)	
<b>exil-weak-equal-p</b> <i>OBJ1 OBJ2</i>	[Method]
<b>Source</b> <a href="#">[utils]</a> , page 3 (Lisp file)	

<code>fact</code>	<i>OBJECT</i>	[Generic Function]
<b>Package</b>	<a href="#">[exil-core]</a> , page 16	
<b>Methods</b>		
	<code>fact</code> ( <i>SIMPLE-FACT</i> <code>simple-fact</code> )	[Method]
	automatically generated reader method	
<b>Source</b>	<a href="#">[facts]</a> , page 5 (Lisp file)	
<code>fact-description</code>	<i>FACT</i>	[Generic Function]
<b>Package</b>	<a href="#">[exil-core]</a> , page 16	
<b>Methods</b>		
	<code>fact-description</code> ( <i>FACT</i> <code>template-fact</code> )	[Method]
<b>Source</b>	<a href="#">[facts]</a> , page 5 (Lisp file)	
	<code>fact-description</code> ( <i>FACT</i> <code>simple-fact</code> )	[Method]
<b>Source</b>	<a href="#">[facts]</a> , page 5 (Lisp file)	
<code>fact-equal-p</code>	<i>FACT1 FACT2</i>	[Generic Function]
<b>Package</b>	<a href="#">[exil-core]</a> , page 16	
<b>Source</b>	<a href="#">[facts]</a> , page 5 (Lisp file)	
<b>Methods</b>		
	<code>fact-equal-p</code> ( <i>FACT1</i> <code>template-fact</code> ) ( <i>FACT2</i> <code>template-fact</code> )	[Method]
<b>Source</b>	<a href="#">[facts]</a> , page 5 (Lisp file)	
	<code>fact-equal-p</code> ( <i>FACT1</i> <code>simple-fact</code> ) ( <i>FACT2</i> <code>simple-fact</code> )	[Method]
<b>Source</b>	<a href="#">[facts]</a> , page 5 (Lisp file)	
	<code>fact-equal-p</code> <i>FACT1 FACT2</i>	[Method]
<b>Source</b>	<a href="#">[facts]</a> , page 5 (Lisp file)	
<code>fact-groups</code>		[Generic Function]
<b>Package</b>	<a href="#">[exil-env]</a> , page 24	
<b>Methods</b>		
	<code>fact-groups</code>	[Method]
<b>Source</b>	<a href="#">[environment]</a> , page 12 (Lisp file)	
<code>fact-slot</code>	<i>FACT SLOT-SPEC</i>	[Generic Function]
	returns fact's slot specified by slot-spec	
<b>Package</b>	<a href="#">[exil-core]</a> , page 16	
<b>Source</b>	<a href="#">[facts]</a> , page 5 (Lisp file)	
<b>Methods</b>		
	<code>fact-slot</code> ( <i>FACT</i> <code>simple-fact</code> ) ( <i>SLOT-SPEC</i> <code>integer</code> )	[Method]
<b>Source</b>	<a href="#">[facts]</a> , page 5 (Lisp file)	

	<code>fact-slot</code> ( <i>FACT</i> template-fact) ( <i>SLOT-SPEC</i> symbol)	[Method]
	Source <code>[facts]</code> , page 5 (Lisp file)	
<code>facts</code>		[Generic Function]
Package	<code>[exil-env]</code> , page 24	
Methods		
	<code>facts</code>	[Method]
	Source <code>[environment]</code> , page 12 (Lisp file)	
<code>find-atom</code> <i>OBJECT</i> <i>ATOM</i>		[Generic Function]
Package	<code>[exil-core]</code> , page 16	
Methods		
	<code>find-atom</code> ( <i>PATTERN</i> simple-pattern) <i>ATOM</i>	[Method]
	Source <code>[patterns]</code> , page 6 (Lisp file)	
	<code>find-atom</code> ( <i>FACT</i> simple-fact) <i>ATOM</i>	[Method]
	Source <code>[facts]</code> , page 5 (Lisp file)	
	<code>find-atom</code> ( <i>OBJECT</i> template-object) <i>ATOM</i>	[Method]
	find the given atom in template-object slots	
	Source <code>[templates]</code> , page 4 (Lisp file)	
<code>find-template</code> <i>NAME</i>		[Generic Function]
Package	<code>[exil-env]</code> , page 24	
Methods		
	<code>find-template</code> <i>NAME</i>	[Method]
	Source <code>[environment]</code> , page 12 (Lisp file)	
<code>has-slot-p</code> <i>OBJECT</i> <i>SLOT-NAME</i>		[Generic Function]
Package	<code>[exil-core]</code> , page 16	
Methods		
	<code>has-slot-p</code> ( <i>OBJECT</i> template-object) <i>SLOT-NAME</i>	[Method]
	Source <code>[templates]</code> , page 4 (Lisp file)	
<code>hash-&gt;list</code> <i>HASH</i>		[Generic Function]
Package	<code>[exil-utils]</code> , page 15	
Methods		
	<code>hash-&gt;list</code> ( <i>HASH</i> hash-table)	[Method]
	Source <code>[utils]</code> , page 3 (Lisp file)	
<code>make-rule</code> <i>NAME</i> <i>CONDITIONS</i> <i>ACTIVATIONS</i>		[Generic Function]
Package	<code>[exil-core]</code> , page 16	
Methods		
	<code>make-rule</code> <i>NAME</i> <i>CONDITIONS</i> <i>ACTIVATIONS</i>	[Method]
	Source <code>[rules]</code> , page 6 (Lisp file)	

<code>match-var</code> <i>OBJECT</i>	[Generic Function]
<code>(setf match-var) NEW-VALUE OBJECT</code>	[Generic Function]
<b>Package</b> <code>[exil-core]</code> , page 16	
<b>Methods</b>	
<code>match-var (PATTERN pattern)</code> automatically generated reader method	[Method]
<b>Source</b> <code>[patterns]</code> , page 6 (Lisp file)	
<code>(setf match-var) NEW-VALUE (PATTERN pattern)</code> automatically generated writer method	[Method]
<b>Source</b> <code>[patterns]</code> , page 6 (Lisp file)	
<code>name</code> <i>OBJECT</i>	[Generic Function]
<b>Package</b> <code>[exil-core]</code> , page 16	
<b>Methods</b>	
<code>name (RULE rule)</code> automatically generated reader method	[Method]
<b>Source</b> <code>[rules]</code> , page 6 (Lisp file)	
<code>name (TEMPLATE template)</code> automatically generated reader method	[Method]
<b>Source</b> <code>[templates]</code> , page 4 (Lisp file)	
<code>negated-p</code> <i>OBJECT</i>	[Generic Function]
<code>(setf negated-p) NEW-VALUE OBJECT</code>	[Generic Function]
<b>Package</b> <code>[exil-core]</code> , page 16	
<b>Methods</b>	
<code>negated-p (PATTERN pattern)</code> automatically generated reader method	[Method]
<b>Source</b> <code>[patterns]</code> , page 6 (Lisp file)	
<code>(setf negated-p) NEW-VALUE (PATTERN pattern)</code> automatically generated writer method	[Method]
<b>Source</b> <code>[patterns]</code> , page 6 (Lisp file)	
<code>new-production</code> <i>RULE &amp;optional RETE</i>	[Generic Function]
<b>Package</b> <code>[exil-rete]</code> , page 19	
<b>Methods</b>	
<code>new-production (RULE rule) &amp;optional RETE</code> Source <code>[rete-net-creation]</code> , page 10 (Lisp file)	[Method]
<code>pattern</code> <i>OBJECT</i>	[Generic Function]
<b>Package</b> <code>[exil-core]</code> , page 16	
<b>Methods</b>	

<p><code>pattern</code> (<i>SIMPLE-PATTERN</i> <code>simple-pattern</code>)  automatically generated reader method</p> <p><b>Source</b>     <a href="#">[patterns]</a>, page 6 (Lisp file)</p>		[Method]
<p><code>pattern-equal-p</code> <i>PATTERN1 PATTERN2</i></p> <p><b>Package</b>     <a href="#">[exil-core]</a>, page 16</p> <p><b>Source</b>     <a href="#">[patterns]</a>, page 6 (Lisp file)</p> <p><b>Methods</b></p>		[Generic Function]
<p><code>pattern-equal-p</code> (<i>PATTERN1</i> <code>template-pattern</code>)  (<i>PATTERN2</i> <code>template-pattern</code>)</p> <p><b>Source</b>     <a href="#">[patterns]</a>, page 6 (Lisp file)</p>		[Method]
<p><code>pattern-equal-p</code> (<i>PATTERN1</i> <code>simple-pattern</code>)  (<i>PATTERN2</i> <code>simple-pattern</code>)</p> <p><b>Source</b>     <a href="#">[patterns]</a>, page 6 (Lisp file)</p>		[Method]
<p><code>pattern-equal-p</code> (<i>PATTERN1</i> <code>pattern</code>) (<i>PATTERN2</i>  <code>pattern</code>)</p> <p><b>Source</b>     <a href="#">[patterns]</a>, page 6 (Lisp file)</p>		[Method]
<p><code>rem-wme</code> <i>FACT &amp;optional RETE</i></p> <p><b>Package</b>     <a href="#">[exil-rete]</a>, page 19</p> <p><b>Methods</b></p>		[Generic Function]
<p><code>rem-wme</code> (<i>FACT</i> <code>fact</code>) <b>&amp;optional</b> <i>RETE</i></p> <p><b>Source</b>     <a href="#">[rete-net-creation]</a>, page 10 (Lisp file)</p>		[Method]
<p><code>remove-match</code> <i>PRODUCTION TOKEN</i></p> <p><b>Package</b>     <a href="#">[exil-env]</a>, page 24</p> <p><b>Methods</b></p>		[Generic Function]
<p><code>remove-match</code> <i>PRODUCTION TOKEN</i></p> <p><b>Source</b>     <a href="#">[environment]</a>, page 12 (Lisp file)</p>		[Method]
<p><code>remove-production</code> <i>RULE &amp;optional RETE</i></p> <p><b>Package</b>     <a href="#">[exil-rete]</a>, page 19</p> <p><b>Methods</b></p>		[Generic Function]
<p><code>remove-production</code> (<i>RULE</i> <code>rule</code>) <b>&amp;optional</b> <i>RETE</i></p> <p><b>Source</b>     <a href="#">[rete-net-creation]</a>, page 10 (Lisp file)</p>		[Method]
<p><code>rete</code></p> <p><b>Package</b>     <a href="#">[exil-env]</a>, page 24</p> <p><b>Methods</b></p>		[Generic Function]
<p><code>rete</code></p> <p><b>Source</b>     <a href="#">[environment]</a>, page 12 (Lisp file)</p>		[Method]

<code>rule-equal-p</code>	<i>RULE1 RULE2</i>	[Generic Function]
Package	[ <i>exil-core</i> ], page 16	
Methods		
	<code>rule-equal-p</code> ( <i>RULE1</i> rule) ( <i>RULE2</i> rule)	[Method]
	Source [rules], page 6 (Lisp file)	
<code>rules</code>		[Generic Function]
Package	[ <i>exil-env</i> ], page 24	
Methods		
	<code>rules</code>	[Method]
	Source [environment], page 12 (Lisp file)	
<code>select-activation</code>		[Generic Function]
Package	[ <i>exil-env</i> ], page 24	
Methods		
	<code>select-activation</code>	[Method]
	Source [environment], page 12 (Lisp file)	
<code>set-strategy</code>	&optional <i>NAME</i>	[Generic Function]
Package	[ <i>exil-env</i> ], page 24	
Methods		
	<code>set-strategy</code> &optional <i>NAME</i>	[Method]
	Source [environment], page 12 (Lisp file)	
<code>set-watcher</code>	<i>WATCHER</i>	[Generic Function]
Package	[ <i>exil-env</i> ], page 24	
Methods		
	<code>set-watcher</code> <i>WATCHER</i>	[Method]
	Source [environment], page 12 (Lisp file)	
<code>slots</code>	<i>OBJECT</i>	[Generic Function]
Package	[ <i>exil-core</i> ], page 16	
Methods		
	<code>slots</code> ( <i>TEMPLATE-OBJECT</i> template-object)	[Method]
	automatically generated reader method	
	Source [templates], page 4 (Lisp file)	
	<code>slots</code> ( <i>TEMPLATE</i> template)	[Method]
	automatically generated reader method	
	Source [templates], page 4 (Lisp file)	
<code>symbol-name</code>	<i>SYMBOL</i>	[Generic Function]
	For symbol return its name, for string just return itself	
Package	[ <i>exil-utils</i> ], page 15	
Source	[utils], page 3 (Lisp file)	
Methods		

symbol-name	( <i>SYMBOL</i> symbol)	[Method]
Source	[utils], page 3 (Lisp file)	
symbol-name	( <i>STRING</i> 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	<i>FACT</i> <i>SLOT-NAME</i>	[Generic Function]
(setf tmpl-fact-slot-value)	<i>VAL</i> <i>FACT</i> <i>SLOT-NAME</i>	[Generic Function]
Package	[exil-core], page 16	
Methods		
tmpl-fact-slot-value	( <i>FACT</i> template-fact) <i>SLOT-NAME</i>	[Method]
(setf tmpl-fact-slot-value)	<i>VAL</i> ( <i>FACT</i> template-fact) <i>SLOT-NAME</i>	[Method]
Source	[facts], page 5 (Lisp file)	
tmpl-name	<i>OBJECT</i>	[Generic Function]
Package	[exil-core], page 16	
Methods		
tmpl-name	( <i>TEMPLATE-OBJECT</i> template-object) automatically generated reader method	[Method]
Source	[templates], page 4 (Lisp file)	
token->list	<i>TOKEN</i>	[Generic Function]
Package	[exil-rete], page 19	
Methods		
token->list	( <i>TOKEN</i> token)	[Method]
Source	[tokens], page 7 (Lisp file)	
token-equal-p	<i>TOKEN1</i> <i>TOKEN2</i>	[Generic Function]
token equality predicate		
Package	[exil-rete], page 19	
Source	[tokens], page 7 (Lisp file)	
Methods		
token-equal-p	<i>TOKEN1</i> <i>TOKEN2</i>	[Method]
Source	[tokens], page 7 (Lisp file)	
token-equal-p	( <i>TOKEN1</i> empty-token) ( <i>TOKEN2</i> 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**

<code>fact</code>		[Slot]
<b>Initargs</b>	<code>:fact</code>	
<b>Initform</b>	<code>(error "fact slot must be specified")</code>	
<b>Readers</b>	<code>[fact]</code> , 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**

<code>pattern</code>		[Slot]
<b>Initargs</b>	<code>:pattern</code>	
<b>Initform</b>	<code>(error "pattern slot must be specified")</code>	
<b>Readers</b>	<code>[pattern]</code> , 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**

<code>slot-default</code>	[Slot]
<code>Allocation :class</code>	
<code>Initform 'exil-core::?</code>	

## 4.2 Internal definitions

### 4.2.1 Special variables

<code>*clips-mode*</code>	[Special Variable]
---------------------------	--------------------

<b>Package</b>	<code>[exil]</code> , page 26
<b>Source</b>	<code>[export]</code> , page 13 (Lisp file)

<code>*current-environment*</code>	[Special Variable]
------------------------------------	--------------------

<b>Package</b>	<code>[exil-env]</code> , page 24
<b>Source</b>	<code>[environment]</code> , page 12 (Lisp file)

<code>*environments*</code>	[Special Variable]
-----------------------------	--------------------

<b>Package</b>	<code>[exil-env]</code> , page 24
<b>Source</b>	<code>[environment]</code> , page 12 (Lisp file)

<code>*exil-running*</code>	[Special Variable]
-----------------------------	--------------------

<b>Package</b>	<code>[exil]</code> , page 26
<b>Source</b>	<code>[export]</code> , page 13 (Lisp file)

### 4.2.2 Macros

<code>defenv NAME &amp;key REDEFINE</code>	[Macro]
--	---------

<b>Package</b>	<code>[exil-env]</code> , page 24
<b>Source</b>	<code>[environment]</code> , page 12 (Lisp file)

<code>exil-env-accessor SLOT-NAME</code>	[Macro]
--	---------

<b>Package</b>	<code>[exil-env]</code> , page 24
<b>Source</b>	<code>[environment]</code> , page 12 (Lisp file)

<code>exil-env-accessors &amp;rest SLOT-NAMES</code>	[Macro]
--	---------

<b>Package</b>	<code>[exil-env]</code> , page 24
<b>Source</b>	<code>[environment]</code> , 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)

<code>complexity-strategy</code>	<i>AGENDA</i>	[Function]
Package	[ <i>exil-env</i> ], page 24	
Source	[ <i>strategies</i> ], page 11 (Lisp file)	
<code>depth-strategy</code>	<i>AGENDA</i>	[Function]
Package	[ <i>exil-env</i> ], page 24	
Source	[ <i>strategies</i> ], page 11 (Lisp file)	
<code>extract-conditions%</code>	<i>COND-LIST</i>	[Function]
Package	[ <i>exil</i> ], page 26	
Source	[ <i>export</i> ], page 13 (Lisp file)	
<code>find-atom-in-cond-list%</code>	<i>ATOM COND-LIST</i>	[Function]
Package	[ <i>exil-rete</i> ], page 19	
Source	[ <i>rete-net-creation</i> ], page 10 (Lisp file)	
<code>get-variable-bindings</code>	<i>PATTERN-LIST FACT-LIST</i>	[Function]
Package	[ <i>exil-env</i> ], page 24	
Source	[ <i>activations</i> ], page 11 (Lisp file)	
<code>make-match</code>	<i>RULE TOKEN &amp;optional TIMESTAMP</i>	[Function]
Package	[ <i>exil-env</i> ], page 24	
Source	[ <i>matches</i> ], page 11 (Lisp file)	
<code>make-test</code>	<i>CURRENT-FIELD PREVIOUS-CONDITION PREVIOUS-FIELD</i>	[Function]
Package	[ <i>exil-rete</i> ], page 19	
Source	[ <i>rete-beta-part</i> ], page 9 (Lisp file)	
<code>make-tmpl-fact</code>	<i>FACT-SPEC</i>	[Function]
Package	[ <i>exil-core</i> ], page 16	
Source	[ <i>facts</i> ], page 5 (Lisp file)	
<code>make-tmpl-obj-clips</code>	<i>OBJECT-TYPE TEMPLATE SLOT-SPECS</i>	[Function]
Package	[ <i>exil-core</i> ], page 16	
Source	[ <i>templates</i> ], page 4 (Lisp file)	
<code>make-tmpl-obj-nonclips</code>	<i>OBJECT-TYPE TEMPLATE SLOT-SPECS</i>	[Function]
Package	[ <i>exil-core</i> ], page 16	
Source	[ <i>templates</i> ], page 4 (Lisp file)	
<code>make-tmpl-object</code>	<i>SPECIFICATION OBJECT-TYPE</i> creates template-object of given type from its specification	[Function]
Package	[ <i>exil-core</i> ], page 16	
Source	[ <i>templates</i> ], page 4 (Lisp file)	
<code>make-tmpl-pattern</code>	<i>PATTERN-SPEC &amp;optional NEGATED MATCH-VAR</i>	[Function]
Package	[ <i>exil-core</i> ], page 16	
Source	[ <i>patterns</i> ], page 6 (Lisp file)	

<code>my-position</code>	<i>ATOM LIST</i>	[Function]
Package	[exil], page 26	
Source	[export], page 13 (Lisp file)	
<code>nonclips-mod-list-p</code>	<i>MOD-LIST</i>	[Function]
Package	[exil], page 26	
Source	[export], page 13 (Lisp file)	
<code>nonclips-slot-&gt;slot-des%</code>	<i>SLOT-SPEC</i>	[Function]
Package	[exil], page 26	
Source	[export], page 13 (Lisp file)	
<code>nonclips-slot-spec-p</code>	<i>SLOT-SPEC</i>	[Function]
Package	[exil], page 26	
Source	[export], page 13 (Lisp file)	
<code>ppdefrule%</code>	<i>NAME</i>	[Function]
Package	[exil], page 26	
Source	[export], page 13 (Lisp file)	
<code>retract%</code>	<i>FACT-SPECS</i>	[Function]
Package	[exil], page 26	
Source	[export], page 13 (Lisp file)	
<code>set-clips-mode</code>	<i>VAL</i>	[Function]
Package	[exil], page 26	
Source	[export], page 13 (Lisp file)	
<code>simplicity-strategy</code>	<i>AGENDA</i>	[Function]
Package	[exil-env], page 24	
Source	[strategies], page 11 (Lisp file)	
<code>slot-&gt;slot-designator%</code>	<i>SLOT-SPEC</i>	[Function]
Package	[exil], page 26	
Source	[export], page 13 (Lisp file)	
<code>slot-spec-p</code>	<i>SLOT-SPEC</i>	[Function]
Package	[exil], page 26	
Source	[export], page 13 (Lisp file)	
<code>slots-&gt;slot-designators%</code>	<i>SLOTS</i>	[Function]
Package	[exil], page 26	
Source	[export], page 13 (Lisp file)	
<code>substitute-variables</code>	<i>ACTIVATIONS-WITH-VARS VAR-BIND-LIST</i>	[Function]
Package	[exil-env], page 24	
Source	[activations], page 11 (Lisp file)	



<code>tmpl-fact-specification-p</code>	<i>FACT-SPEC</i>	[Function]
<b>Package</b>	<a href="#">[exil-core]</a> , page 16	
<b>Source</b>	<a href="#">[facts]</a> , page 5 (Lisp file)	
<code>tmpl-object-specification-p</code>	<i>SPECIFICATION</i>	[Function]
	is this a template-object specification?	
<b>Package</b>	<a href="#">[exil-core]</a> , page 16	
<b>Source</b>	<a href="#">[templates]</a> , page 4 (Lisp file)	
<code>tmpl-pattern-specification-p</code>	<i>SPECIFICATION</i>	[Function]
<b>Package</b>	<a href="#">[exil-core]</a> , page 16	
<b>Source</b>	<a href="#">[patterns]</a> , page 6 (Lisp file)	
<code>tmpl-slot-spec-p</code>	<i>SPECIFICATION</i>	[Function]
<b>Package</b>	<a href="#">[exil-core]</a> , page 16	
<b>Source</b>	<a href="#">[templates]</a> , page 4 (Lisp file)	
<code>to-mod-spec-list</code>	<i>MOD-LIST</i>	[Function]
<b>Package</b>	<a href="#">[exil]</a> , page 26	
<b>Source</b>	<a href="#">[export]</a> , page 13 (Lisp file)	

#### 4.2.4 Generic functions

<code>activate</code>	<i>NODE OBJECT</i>	[Generic Function]
	handles various node activations	
<b>Package</b>	<a href="#">[exil-rete]</a> , page 19	
<b>Source</b>	<a href="#">[rete-generic-node]</a> , page 7 (Lisp file)	
<b>Methods</b>		
	<code>activate (NODE beta-negative-node) (WME fact)</code>	[Method]
	<b>Source</b> <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)	
	<code>activate (NODE beta-negative-node) (TOKEN token)</code>	[Method]
	<b>Source</b> <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)	
	<code>activate (NODE beta-join-node) (WME fact)</code>	[Method]
	<b>Source</b> <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)	
	<code>activate (NODE beta-join-node) (TOKEN token)</code>	[Method]
	<b>Source</b> <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)	
	<code>activate (NODE beta-memory-node) (TOKEN token)</code>	[Method]
	<b>Source</b> <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)	
	<code>activate (NODE alpha-memory-node) (WME fact)</code>	[Method]
	<b>Source</b> <a href="#">[rete-alpha-part]</a> , page 8 (Lisp file)	
	<code>activate (NODE alpha-top-node) (WME fact)</code>	[Method]
	<b>Source</b> <a href="#">[rete-alpha-part]</a> , page 8 (Lisp file)	

activate	( <i>NODE</i> alpha-subtop-node) ( <i>WME fact</i> )	[Method]
Source	[rete-alpha-part], page 8 (Lisp file)	
activate	( <i>NODE</i> alpha-test-node) ( <i>WME fact</i> )	[Method]
Source	[rete-alpha-part], page 8 (Lisp file)	
activate-children	<i>NODE OBJECT</i>	[Generic Function]
Package	[exil-rete], page 19	
Source	[rete-generic-node], page 7 (Lisp file)	
Methods		
activate-children	( <i>NODE</i> alpha-test-node) ( <i>WME fact</i> )	[Method]
Source	[rete-alpha-part], page 8 (Lisp file)	
activate-children	( <i>NODE</i> node) <i>OBJECT</i>	[Method]
Source	[rete-generic-node], page 7 (Lisp file)	
activate-memory	<i>NODE WME</i>	[Generic Function]
Package	[exil-rete], page 19	
Methods		
activate-memory	( <i>NODE</i> alpha-test-node) ( <i>WME fact</i> )	[Method]
Source	[rete-alpha-part], page 8 (Lisp file)	
add-child	<i>NODE CHILD</i>	[Generic Function]
Package	[exil-rete], page 19	
Methods		
add-child	( <i>NODE</i> node) ( <i>CHILD</i> node)	[Method]
Source	[rete-generic-node], page 7 (Lisp file)	
add-children	<i>NODE CHILDREN</i>	[Generic Function]
Package	[exil-rete], page 19	
Methods		
add-children	( <i>NODE</i> node) ( <i>CHILDREN</i> list)	[Method]
Source	[rete-generic-node], page 7 (Lisp file)	
add-item	<i>NODE ITEM &amp;optional EQUALITY-PREDICATE</i>	[Generic Function]
Package	[exil-rete], page 19	
Methods		
add-item	( <i>NODE</i> memory-node) <i>ITEM &amp;optional EQUALITY-PREDICATE</i>	[Method]
Source	[rete-generic-node], page 7 (Lisp file)	
add-production	<i>NODE PRODUCTION</i>	[Generic Function]
Package	[exil-rete], page 19	
Methods		

add-production ( <i>NODE</i> beta-memory-node) ( <i>PRODUCTION</i> rule)		[Method]
<b>Source</b> <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)		
alpha-memory <i>OBJECT</i>		[Generic Function]
<b>Package</b> <a href="#">[exil-rete]</a> , page 19		
<b>Methods</b>		
alpha-memory ( <i>BETA-JOIN-NODE</i> beta-join-node) automatically generated reader method		[Method]
<b>Source</b> <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)		
alpha-top-node <i>OBJECT</i>		[Generic Function]
<b>Package</b> <a href="#">[exil-rete]</a> , page 19		
<b>Methods</b>		
alpha-top-node ( <i>RETE</i> rete) automatically generated reader method		[Method]
<b>Source</b> <a href="#">[rete-net-creation]</a> , page 10 (Lisp file)		
beta-memory <i>NODE</i>		[Generic Function]
<b>Package</b> <a href="#">[exil-rete]</a> , page 19		
<b>Methods</b>		
beta-memory ( <i>NODE</i> beta-join-node)		[Method]
<b>Source</b> <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)		
beta-top-node <i>OBJECT</i>		[Generic Function]
(setf beta-top-node) <i>NEW-VALUE OBJECT</i>		[Generic Function]
<b>Package</b> <a href="#">[exil-rete]</a> , page 19		
<b>Methods</b>		
beta-top-node ( <i>RETE</i> rete) automatically generated reader method		[Method]
<b>Source</b> <a href="#">[rete-net-creation]</a> , page 10 (Lisp file)		
(setf beta-top-node) <i>NEW-VALUE</i> ( <i>RETE</i> rete) automatically generated writer method		[Method]
<b>Source</b> <a href="#">[rete-net-creation]</a> , page 10 (Lisp file)		
broken-match <i>NODE TOKEN</i>		[Generic Function]
<b>Package</b> <a href="#">[exil-rete]</a> , page 19		
<b>Methods</b>		
broken-match ( <i>NODE</i> beta-memory-node) ( <i>TOKEN</i> token)		[Method]
<b>Source</b> <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)		
children <i>OBJECT</i>		[Generic Function]
(setf children) <i>NEW-VALUE OBJECT</i>		[Generic Function]
<b>Package</b> <a href="#">[exil-rete]</a> , page 19		
<b>Methods</b>		

<code>children</code>	<i>(NODE node)</i>	[Method]
	automatically generated reader method	
<b>Source</b>	[rete-generic-node], page 7 (Lisp file)	
<code>(setf children)</code>	<i>NEW-VALUE (NODE node)</i>	[Method]
	automatically generated writer method	
<b>Source</b>	[rete-generic-node], page 7 (Lisp file)	
<code>complete-match</code>	<i>NODE TOKEN</i>	[Generic Function]
<b>Package</b>	[exil-rete], page 19	
<b>Methods</b>		
	<code>complete-match</code> <i>(NODE beta-memory-node) (TOKEN token)</i>	[Method]
<b>Source</b>	[rete-beta-part], page 9 (Lisp file)	
<code>create-alpha-net</code>	<i>PATTERN &amp;optional RETE</i>	[Generic Function]
<b>Package</b>	[exil-rete], page 19	
<b>Methods</b>		
	<code>create-alpha-net</code> <i>(PATTERN template-pattern) &amp;optional RETE</i>	[Method]
<b>Source</b>	[rete-net-creation], page 10 (Lisp file)	
	<code>create-alpha-net</code> <i>(PATTERN simple-pattern) &amp;optional RETE</i>	[Method]
<b>Source</b>	[rete-net-creation], page 10 (Lisp file)	
<code>create-alpha-net%</code>	<i>PATTERN ROOT</i>	[Generic Function]
<b>Package</b>	[exil-rete], page 19	
<b>Methods</b>		
	<code>create-alpha-net%</code> <i>(PATTERN template-pattern) (ROOT template-fact-subtop-node)</i>	[Method]
<b>Source</b>	[rete-net-creation], page 10 (Lisp file)	
	<code>create-alpha-net%</code> <i>(PATTERN simple-pattern) (ROOT simple-fact-subtop-node)</i>	[Method]
<b>Source</b>	[rete-net-creation], page 10 (Lisp file)	
<code>current-field</code>	<i>OBJECT</i>	[Generic Function]
<b>Package</b>	[exil-rete], page 19	
<b>Methods</b>		
	<code>current-field</code> <i>(TEST test)</i>	[Method]
	automatically generated reader method	
<b>Source</b>	[rete-beta-part], page 9 (Lisp file)	
<code>current-strategy</code>		[Generic Function]
<b>Package</b>	[exil-env], page 24	
<b>Methods</b>		

<code>current-strategy</code>	[Method]
<b>Source</b> <a href="#">[environment]</a> , page 12 (Lisp file)	
<code>current-strategy-name</code>	[Generic Function]
<b>Package</b> <a href="#">[exil-env]</a> , page 24	
<b>Methods</b>	
<code>current-strategy-name</code>	[Method]
<b>Source</b> <a href="#">[environment]</a> , page 12 (Lisp file)	
<code>delete-production</code> <i>NODE PRODUCTION</i>	[Generic Function]
<b>Package</b> <a href="#">[exil-rete]</a> , page 19	
<b>Methods</b>	
<code>delete-production</code> ( <i>NODE</i> beta-memory-node) ( <i>PRODUCTION</i> rule)	[Method]
<b>Source</b> <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)	
<code>description</code> <i>OBJECT</i>	[Generic Function]
( <code>setf</code> <code>description</code> ) <i>NEW-VALUE OBJECT</i>	[Generic Function]
<b>Package</b> <a href="#">[exil-rete]</a> , page 19	
<b>Methods</b>	
<code>description</code> ( <i>DESCRIBED-OBJECT</i> described-object)	[Method]
automatically generated reader method	
<b>Source</b> <a href="#">[rete-generic-node]</a> , page 7 (Lisp file)	
( <code>setf</code> <code>description</code> ) <i>NEW-VALUE</i> ( <i>DESCRIBED-OBJECT</i> described-object)	[Method]
automatically generated writer method	
<b>Source</b> <a href="#">[rete-generic-node]</a> , page 7 (Lisp file)	
<code>find-test-node</code> <i>PARENT FIELD VALUE</i>	[Generic Function]
<b>Package</b> <a href="#">[exil-rete]</a> , page 19	
<b>Methods</b>	
<code>find-test-node</code> ( <i>PARENT</i> alpha-node) <i>FIELD VALUE</i>	[Method]
<b>Source</b> <a href="#">[rete-net-creation]</a> , page 10 (Lisp file)	
<code>find/create-join-node</code> <i>PARENT TESTS A-MEMORY</i>	[Generic Function]
<b>Package</b> <a href="#">[exil-rete]</a> , page 19	
<b>Methods</b>	
<code>find/create-join-node</code> ( <i>PARENT</i> beta-memory-node) ( <i>TESTS</i> list) ( <i>A-MEMORY</i> alpha-memory-node)	[Method]
<b>Source</b> <a href="#">[rete-net-creation]</a> , page 10 (Lisp file)	
<code>find/create-neg-node</code> <i>PARENT TESTS A-MEMORY</i>	[Generic Function]
<b>Package</b> <a href="#">[exil-rete]</a> , page 19	
<b>Methods</b>	

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

**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* [Generic Function]  
                           *NEW-NODE-TYPE*

**Package**    [\[exil-rete\]](#), page 19

**Methods**

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

**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**

<code>get-intracondition-tests%</code> ( <i>CONDITION</i> template-pattern)	[Method]
Source <a href="#">[rete-net-creation]</a> , page 10 (Lisp file)	
<code>get-intracondition-tests%</code> ( <i>CONDITION</i> simple-pattern)	[Method]
Source <a href="#">[rete-net-creation]</a> , page 10 (Lisp file)	
<code>get-join-tests-from-condition</code> <i>CONDITION PREV-CONDS</i>	[Generic Function]
Package <a href="#">[exil-rete]</a> , page 19	
Methods	
<code>get-join-tests-from-condition</code> ( <i>CONDITION</i> pattern) ( <i>PREV-CONDS</i> list)	[Method]
Source <a href="#">[rete-net-creation]</a> , page 10 (Lisp file)	
<code>get-network</code> <i>NODE</i> &optional <i>TEMPLATE-NAME</i>	[Generic Function]
(setf <code>get-network</code> ) <i>VALUE</i> <i>NODE</i> &optional <i>TEMPLATE-NAME</i>	[Generic Function]
Package <a href="#">[exil-rete]</a> , page 19	
Methods	
<code>get-network</code> ( <i>NODE</i> alpha-top-node) &optional <i>TEMPLATE-NAME</i>	[Method]
(setf <code>get-network</code> ) <i>VALUE</i> ( <i>NODE</i> alpha-top-node) &optional <i>TEMPLATE-NAME</i>	[Method]
Source <a href="#">[rete-alpha-part]</a> , page 8 (Lisp file)	
<code>get/initialize-network</code> <i>NODE</i> &optional <i>TEMPLATE-NAME</i>	[Generic Function]
Package <a href="#">[exil-rete]</a> , page 19	
Methods	
<code>get/initialize-network</code> ( <i>NODE</i> alpha-top-node) &optional <i>TEMPLATE-NAME</i>	[Method]
Source <a href="#">[rete-alpha-part]</a> , page 8 (Lisp file)	
<code>inactivate</code> <i>NODE</i> <i>OBJECT</i>	[Generic Function]
Package <a href="#">[exil-rete]</a> , page 19	
Source <a href="#">[rete-generic-node]</a> , page 7 (Lisp file)	
Methods	
<code>inactivate</code> ( <i>NODE</i> beta-negative-node) ( <i>WME</i> fact)	[Method]
Source <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)	
<code>inactivate</code> ( <i>NODE</i> beta-memory-node) ( <i>TOKEN</i> token) before	[Method]
Source <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)	
<code>inactivate</code> ( <i>NODE</i> beta-memory-node) ( <i>FACT</i> fact) before	[Method]
Source <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)	

<code>inactivate</code>	<i>(NODE alpha-memory-node) (WME fact)</i>	[Method]
<b>Source</b>	<a href="#">[rete-alpha-part]</a> , page 8 (Lisp file)	
<code>inactivate</code>	<i>(NODE alpha-top-node) (WME fact)</i>	[Method]
<b>Source</b>	<a href="#">[rete-alpha-part]</a> , page 8 (Lisp file)	
<code>inactivate</code>	<i>(NODE alpha-test-node) (WME fact) after</i>	[Method]
<b>Source</b>	<a href="#">[rete-alpha-part]</a> , page 8 (Lisp file)	
<code>inactivate</code>	<i>(NODE node) OBJECT</i>	[Method]
<b>Source</b>	<a href="#">[rete-generic-node]</a> , page 7 (Lisp file)	
<code>inactivate-children</code>	<i>NODE OBJECT</i>	[Generic Function]
<b>Package</b>	<a href="#">[exil-rete]</a> , page 19	
<b>Methods</b>		
<code>inactivate-children</code>	<i>(NODE node) OBJECT</i>	[Method]
<b>Source</b>	<a href="#">[rete-generic-node]</a> , page 7 (Lisp file)	
<code>includes-p</code>	<i>FACT TOKEN</i>	[Generic Function]
<b>Package</b>	<a href="#">[exil-rete]</a> , page 19	
<b>Methods</b>		
<code>includes-p</code>	<i>(INCLUDED-TOKEN token) (TOKEN token)</i>	[Method]
<b>Source</b>	<a href="#">[tokens]</a> , page 7 (Lisp file)	
<code>includes-p</code>	<i>(FACT fact) (TOKEN token)</i>	[Method]
<b>Source</b>	<a href="#">[tokens]</a> , page 7 (Lisp file)	
<code>initialize-network</code>	<i>NODE &amp;optional TEMPLATE-NAME</i>	[Generic Function]
<b>Package</b>	<a href="#">[exil-rete]</a> , page 19	
<b>Methods</b>		
<code>initialize-network</code>	<i>(NODE alpha-top-node) &amp;optional TEMPLATE-NAME</i>	[Method]
<b>Source</b>	<a href="#">[rete-alpha-part]</a> , page 8 (Lisp file)	
<code>is-watcher</code>	<i>WATCHER</i>	[Generic Function]
<b>Package</b>	<a href="#">[exil-env]</a> , page 24	
<b>Methods</b>		
<code>is-watcher</code>	<i>(WATCHER symbol)</i>	[Method]
<b>Source</b>	<a href="#">[environment]</a> , page 12 (Lisp file)	
<code>items</code>	<i>OBJECT</i>	[Generic Function]
<code>(setf items)</code>	<i>NEW-VALUE OBJECT</i>	[Generic Function]
<b>Package</b>	<a href="#">[exil-rete]</a> , page 19	
<b>Methods</b>		
<code>items</code>	<i>(MEMORY-NODE memory-node)</i>	[Method]
	automatically generated reader method	
<b>Source</b>	<a href="#">[rete-generic-node]</a> , page 7 (Lisp file)	



(setf items) <i>NEW-VALUE</i> ( <i>MEMORY-NODE</i> memory-node)		[Method]
automatically generated writer method		
<b>Source</b> [rete-generic-node], page 7 (Lisp file)		
match-equal-p <i>MATCH1 MATCH2</i>		[Generic Function]
<b>Package</b> [exil-env], page 24		
<b>Methods</b>		
match-equal-p ( <i>MATCH1</i> match) ( <i>MATCH2</i> match)		[Method]
<b>Source</b> [matches], page 11 (Lisp file)		
match-rule <i>OBJECT</i>		[Generic Function]
<b>Package</b> [exil-env], page 24		
<b>Methods</b>		
match-rule ( <i>MATCH</i> match)		[Method]
automatically generated reader method		
<b>Source</b> [matches], page 11 (Lisp file)		
match-token <i>OBJECT</i>		[Generic Function]
<b>Package</b> [exil-env], page 24		
<b>Methods</b>		
match-token ( <i>MATCH</i> match)		[Method]
automatically generated reader method		
<b>Source</b> [matches], page 11 (Lisp file)		
memory <i>OBJECT</i>		[Generic Function]
(setf memory) <i>NEW-VALUE OBJECT</i>		[Generic Function]
<b>Package</b> [exil-rete], page 19		
<b>Methods</b>		
memory ( <i>ALPHA-TEST-NODE</i> alpha-test-node)		[Method]
automatically generated reader method		
<b>Source</b> [rete-alpha-part], page 8 (Lisp file)		
(setf memory) <i>NEW-VALUE</i> ( <i>ALPHA-TEST-NODE</i> alpha-test-node)		[Method]
automatically generated writer method		
<b>Source</b> [rete-alpha-part], page 8 (Lisp file)		
modify% <i>FACT-SPEC MOD-LIST</i>		[Generic Function]
<b>Package</b> [exil], page 26		
<b>Methods</b>		
modify% ( <i>FACT-SPEC</i> integer) <i>MOD-LIST</i>		[Method]
<b>Source</b> [export], page 13 (Lisp file)		
modify% ( <i>FACT-SPEC</i> list) <i>MOD-LIST</i>		[Method]
<b>Source</b> [export], page 13 (Lisp file)		

<code>negative-wmes</code>	<i>OBJECT</i>	[Generic Function]
<code>(setf negative-wmes)</code>	<i>NEW-VALUE OBJECT</i>	[Generic Function]
<b>Package</b>	<code>[exil-rete]</code> , page 19	
<b>Methods</b>		
<code>negative-wmes</code>	<i>(TOKEN token)</i>	[Method]
	automatically generated reader method	
<b>Source</b>	<code>[tokens]</code> , page 7 (Lisp file)	
<code>(setf negative-wmes)</code>	<i>NEW-VALUE (TOKEN token)</i>	[Method]
	automatically generated writer method	
<b>Source</b>	<code>[tokens]</code> , page 7 (Lisp file)	
<code>networks</code>	<i>OBJECT</i>	[Generic Function]
<code>(setf networks)</code>	<i>NEW-VALUE OBJECT</i>	[Generic Function]
<b>Package</b>	<code>[exil-rete]</code> , page 19	
<b>Methods</b>		
<code>networks</code>	<i>(ALPHA-TOP-NODE alpha-top-node)</i>	[Method]
	automatically generated reader method	
<b>Source</b>	<code>[rete-alpha-part]</code> , page 8 (Lisp file)	
<code>(setf networks)</code>	<i>NEW-VALUE (ALPHA-TOP-NODE alpha-top-node)</i>	[Method]
	automatically generated writer method	
<b>Source</b>	<code>[rete-alpha-part]</code> , page 8 (Lisp file)	
<code>newer-than</code>	<i>MATCH1 MATCH2</i>	[Generic Function]
<b>Package</b>	<code>[exil-env]</code> , page 24	
<b>Methods</b>		
<code>newer-than</code>	<i>(MATCH1 match) (MATCH2 match)</i>	[Method]
<b>Source</b>	<code>[strategies]</code> , page 11 (Lisp file)	
<code>node-equal-p</code>	<i>NODE1 NODE2</i>	[Generic Function]
<b>Package</b>	<code>[exil-rete]</code> , page 19	
<b>Source</b>	<code>[rete-generic-node]</code> , page 7 (Lisp file)	
<b>Methods</b>		
<code>node-equal-p</code>	<i>(NODE1 beta-join-node) (NODE2 beta-join-node)</i>	[Method]
<b>Source</b>	<code>[rete-beta-part]</code> , page 9 (Lisp file)	
<code>node-equal-p</code>	<i>(NODE1 alpha-test-node) (NODE2 alpha-test-node)</i>	[Method]
<b>Source</b>	<code>[rete-alpha-part]</code> , page 8 (Lisp file)	
<code>node-equal-p</code>	<i>(NODE1 (eq1 nil)) (NODE2 (eq1 nil))</i>	[Method]
<b>Source</b>	<code>[rete-generic-node]</code> , page 7 (Lisp file)	

node-equal-p ( <i>NODE1</i> node) ( <i>NODE2</i> node)	[Method]
Source <a href="#">[rete-generic-node]</a> , page 7 (Lisp file)	
parent <i>OBJECT</i>	[Generic Function]
(setf parent) <i>NEW-VALUE OBJECT</i>	[Generic Function]
Package <a href="#">[exil-rete]</a> , page 19	
Methods	
parent ( <i>BETA-NODE</i> beta-node)	[Method]
automatically generated reader method	
Source <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)	
(setf parent) <i>NEW-VALUE</i> ( <i>BETA-NODE</i> beta-node)	[Method]
automatically generated writer method	
Source <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)	
parent ( <i>TOKEN</i> token)	[Method]
automatically generated reader method	
Source <a href="#">[tokens]</a> , page 7 (Lisp file)	
perform-join-test <i>TEST TOKEN WME</i>	[Generic Function]
Package <a href="#">[exil-rete]</a> , page 19	
Methods	
perform-join-test ( <i>TEST</i> test) ( <i>TOKEN</i> token) ( <i>WME</i> fact)	[Method]
Source <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)	
perform-join-tests <i>TESTS TOKEN WME</i>	[Generic Function]
Package <a href="#">[exil-rete]</a> , page 19	
Methods	
perform-join-tests ( <i>TESTS</i> list) ( <i>TOKEN</i> token) ( <i>WME</i> fact)	[Method]
Source <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)	
previous-condition <i>OBJECT</i>	[Generic Function]
Package <a href="#">[exil-rete]</a> , page 19	
Methods	
previous-condition ( <i>TEST</i> test)	[Method]
tells, how many conditions back i must go	
Source <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)	
previous-field <i>OBJECT</i>	[Generic Function]
Package <a href="#">[exil-rete]</a> , page 19	
Methods	
previous-field ( <i>TEST</i> test)	[Method]
automatically generated reader method	
Source <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)	

<code>previous-wme</code>	<i>TOKEN</i> & <b>optional</b> <i>N</i>	[Generic Function]
<b>Package</b>	<a href="#">[exil-rete]</a> , page 19	
<b>Methods</b>		
	<code>previous-wme</code> ( <i>TOKEN</i> token) & <b>optional</b> <i>N</i>	[Method]
	gives wme from token n wmes back	
	<b>Source</b>	<a href="#">[tokens]</a> , page 7 (Lisp file)
<code>productions</code>	<i>OBJECT</i>	[Generic Function]
<code>(setf productions)</code>	<i>NEW-VALUE OBJECT</i>	[Generic Function]
<b>Package</b>	<a href="#">[exil-rete]</a> , page 19	
<b>Methods</b>		
	<code>productions</code> ( <i>BETA-MEMORY-NODE</i> beta-memory-node)	[Method]
	automatically generated reader method	
	<b>Source</b>	<a href="#">[rete-beta-part]</a> , page 9 (Lisp file)
	<code>(setf productions)</code> <i>NEW-VALUE</i> ( <i>BETA-MEMORY-NODE</i> beta-memory-node)	[Method]
	automatically generated writer method	
	<b>Source</b>	<a href="#">[rete-beta-part]</a> , page 9 (Lisp file)
<code>remove-matches</code>	<i>RULE</i>	[Generic Function]
<b>Package</b>	<a href="#">[exil-env]</a> , page 24	
<b>Methods</b>		
	<code>remove-matches</code> <i>RULE</i>	[Method]
	<b>Source</b>	<a href="#">[environment]</a> , page 12 (Lisp file)
<code>simple-fact-key-name</code>	<i>OBJECT</i>	[Generic Function]
<b>Package</b>	<a href="#">[exil-rete]</a> , page 19	
<b>Methods</b>		
	<code>simple-fact-key-name</code> ( <i>ALPHA-TOP-NODE</i> alpha-top-node)	[Method]
	automatically generated reader method	
	<b>Source</b>	<a href="#">[rete-alpha-part]</a> , page 8 (Lisp file)
<code>simpler-than</code>	<i>RULE1 RULE2</i>	[Generic Function]
<b>Package</b>	<a href="#">[exil-env]</a> , page 24	
<b>Methods</b>		
	<code>simpler-than</code> ( <i>MATCH1</i> match) ( <i>MATCH2</i> match)	[Method]
	<b>Source</b>	<a href="#">[strategies]</a> , page 11 (Lisp file)
	<code>simpler-than</code> ( <i>RULE1</i> rule) ( <i>RULE2</i> rule)	[Method]
	<b>Source</b>	<a href="#">[strategies]</a> , page 11 (Lisp file)
<code>strategies</code>		[Generic Function]
<b>Package</b>	<a href="#">[exil-env]</a> , page 24	
<b>Methods</b>		

<b>strategies</b>	[Method]
<b>Source</b> <a href="#">[environment]</a> , page 12 (Lisp file)	
<b>test</b> <i>NODE WME</i>	[Generic Function]
provides testing part of alpha-test-node activation	
<b>Package</b> <a href="#">[exil-rete]</a> , page 19	
<b>Source</b> <a href="#">[rete-alpha-part]</a> , page 8 (Lisp file)	
<b>Methods</b>	
<b>test</b> ( <i>NODE</i> template-fact-test-node) ( <i>WME</i> template-fact)	[Method]
<b>Source</b> <a href="#">[rete-alpha-part]</a> , page 8 (Lisp file)	
<b>test</b> ( <i>NODE</i> simple-fact-test-node) ( <i>WME</i> simple-fact)	[Method]
<b>Source</b> <a href="#">[rete-alpha-part]</a> , page 8 (Lisp file)	
<b>test</b> ( <i>NODE</i> alpha-test-node) ( <i>WME</i> fact)	[Method]
<b>Source</b> <a href="#">[rete-alpha-part]</a> , page 8 (Lisp file)	
<b>test-equal-p</b> <i>TEST1 TEST2</i>	[Generic Function]
<b>Package</b> <a href="#">[exil-rete]</a> , page 19	
<b>Methods</b>	
<b>test-equal-p</b> ( <i>TEST1</i> test) ( <i>TEST2</i> test)	[Method]
<b>Source</b> <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)	
<b>tested-field</b> <i>OBJECT</i>	[Generic Function]
<b>Package</b> <a href="#">[exil-rete]</a> , page 19	
<b>Methods</b>	
<b>tested-field</b> ( <i>ALPHA-TEST-NODE</i> alpha-test-node)	[Method]
automatically generated reader method	
<b>Source</b> <a href="#">[rete-alpha-part]</a> , page 8 (Lisp file)	
<b>tests</b> <i>OBJECT</i>	[Generic Function]
(setf tests) <i>NEW-VALUE OBJECT</i>	[Generic Function]
<b>Package</b> <a href="#">[exil-rete]</a> , page 19	
<b>Methods</b>	
<b>tests</b> ( <i>BETA-JOIN-NODE</i> beta-join-node)	[Method]
automatically generated reader method	
<b>Source</b> <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)	
(setf tests) <i>NEW-VALUE</i> ( <i>BETA-JOIN-NODE</i> beta-join-node)	[Method]
automatically generated writer method	
<b>Source</b> <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)	
<b>tests-equal-p</b> <i>TEST-LIST1 TEST-LIST2</i>	[Generic Function]
<b>Package</b> <a href="#">[exil-rete]</a> , page 19	
<b>Methods</b>	

<code>tests-equal-p</code> ( <i>TEST-LIST1</i> list) ( <i>TEST-LIST2</i> list)		[Method]
Source <a href="#">[rete-beta-part]</a> , page 9 (Lisp file)		
<code>timestamp</code> <i>OBJECT</i>		[Generic Function]
Package <a href="#">[exil-env]</a> , page 24		
Methods		
<code>timestamp</code> ( <i>MATCH</i> match)		[Method]
automatically generated reader method		
Source <a href="#">[matches]</a> , page 11 (Lisp file)		
<code>tmpl-object-equal-p</code> <i>OBJECT1 OBJECT2</i>		[Generic Function]
Package <a href="#">[exil-core]</a> , page 16		
Methods		
<code>tmpl-object-equal-p</code> ( <i>OBJECT1</i> template-object) ( <i>OBJECT2</i> template-object)		[Method]
template-object equality predicate		
Source <a href="#">[templates]</a> , page 4 (Lisp file)		
<code>tmpl-object-slot-value</code> <i>OBJECT SLOT-NAME</i>		[Generic Function]
( <code>setf</code> <code>tmpl-object-slot-value</code> ) <i>VAL OBJECT SLOT-NAME</i>		[Generic Function]
Package <a href="#">[exil-core]</a> , page 16		
Methods		
<code>tmpl-object-slot-value</code> ( <i>OBJECT</i> template-object) <i>SLOT-NAME</i>		[Method]
(setf <code>tmpl-object-slot-value</code> ) <i>VAL (OBJECT</i> template-object) <i>SLOT-NAME</i>		[Method]
get the template-object slot value according to the slot name		
Source <a href="#">[templates]</a> , page 4 (Lisp file)		
<code>tmpl-pattern-slot-value</code> <i>PATTERN SLOT-NAME</i>		[Generic Function]
Package <a href="#">[exil-core]</a> , page 16		
Methods		
<code>tmpl-pattern-slot-value</code> ( <i>PATTERN</i> template-pattern) <i>SLOT-NAME</i>		[Method]
Source <a href="#">[patterns]</a> , page 6 (Lisp file)		
<code>token</code> <i>WME &amp;optional PARENT</i>		[Generic Function]
Package <a href="#">[exil-rete]</a> , page 19		
Methods		
<code>token</code> <i>WME &amp;optional PARENT</i>		[Method]
Source <a href="#">[tokens]</a> , page 7 (Lisp file)		
<code>value</code> <i>OBJECT</i>		[Generic Function]
Package <a href="#">[exil-rete]</a> , page 19		
Methods		

<code>value</code>	<code>(<i>ALPHA-TEST-NODE</i> <i>alpha-test-node</i>)</code>	[Method]
	automatically generated reader method	
<b>Source</b>	<code>[rete-alpha-part]</code> , page 8 (Lisp file)	
<code>variable-bindings</code>	<i>PATTERN</i> <i>FACT</i>	[Generic Function]
<b>Package</b>	<code>[exil-env]</code> , page 24	
<b>Methods</b>		
	<code>variable-bindings</code> ( <i>PATTERN</i> <i>template-pattern</i> ) ( <i>FACT</i> <i>template-fact</i> )	[Method]
<b>Source</b>	<code>[activations]</code> , page 11 (Lisp file)	
	<code>variable-bindings</code> ( <i>PATTERN</i> <i>simple-pattern</i> ) ( <i>FACT</i> <i>simple-fact</i> )	[Method]
<b>Source</b>	<code>[activations]</code> , page 11 (Lisp file)	
<code>watchers</code>		[Generic Function]
<b>Package</b>	<code>[exil-env]</code> , page 24	
<b>Methods</b>		
	<code>watchers</code>	[Method]
<b>Source</b>	<code>[environment]</code> , page 12 (Lisp file)	
<code>wme</code>	<i>OBJECT</i>	[Generic Function]
<b>Package</b>	<code>[exil-rete]</code> , page 19	
<b>Methods</b>		
	<code>wme</code> ( <i>TOKEN</i> <i>token</i> )	[Method]
	automatically generated reader method	
<b>Source</b>	<code>[tokens]</code> , page 7 (Lisp file)	

#### 4.2.5 Classes

<code>alpha-memory-node</code>		[Class]
<b>Package</b>	<code>[exil-rete]</code> , page 19	
<b>Source</b>	<code>[rete-alpha-part]</code> , page 8 (Lisp file)	
<b>Direct superclasses</b>		
	<ul style="list-style-type: none"> <li><code>[memory-node]</code>, page 74 (class)</li> <li><code>[alpha-node]</code>, page 67 (class)</li> </ul>	
<b>Direct methods</b>		
	<ul style="list-style-type: none"> <li><code>[find/create-neg-node]</code>, page 58 (method)</li> <li><code>[find/create-join-node]</code>, page 57 (method)</li> <li><code>[inactivate]</code>, page 60 (method)</li> <li><code>[activate]</code>, page 53 (method)</li> </ul>	
<code>alpha-node</code>		[Class]
<b>Package</b>	<code>[exil-rete]</code> , page 19	
<b>Source</b>	<code>[rete-alpha-part]</code> , 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)

<b>Direct superclasses</b>	
	<a href="#">[beta-memory-node]</a> , page 70 (class)
<b>Direct slots</b>	
items	[Slot]
Initform	(list (make-instance 'exil-rete::empty-token))
described-object	[Class]
Package	<a href="#">[exil-rete]</a> , page 19
Source	<a href="#">[rete-generic-node]</a> , page 7 (Lisp file)
<b>Direct superclasses</b>	
	standard-object
<b>Direct subclasses</b>	
	<a href="#">[node]</a> , page 74 (class)
<b>Direct methods</b>	
	<ul style="list-style-type: none"> <li>• print-object</li> <li>• description</li> <li>• <a href="#">[description]</a>, page 57 (method)</li> </ul>
<b>Direct slots</b>	
description	[Slot]
Initargs	:description
Initform	""
Readers	<a href="#">[description]</a> , page 57 (generic function)
Writers	<a href="#">[(setf description)]</a> , page 57 (generic function)
empty-token	[Class]
Package	<a href="#">[exil-rete]</a> , page 19
Source	<a href="#">[tokens]</a> , page 7 (Lisp file)
<b>Direct superclasses</b>	
	<a href="#">[token]</a> , page 78 (class)
<b>Direct methods</b>	
	<a href="#">[token-equal-p]</a> , page 43 (method)
<b>Direct slots</b>	
wme	[Slot]
exil-environment	[Class]
Package	<a href="#">[exil-env]</a> , page 24
Source	<a href="#">[environment]</a> , page 12 (Lisp file)
<b>Direct superclasses</b>	
	standard-object
<b>Direct slots</b>	
facts	[Slot]
fact-groups	[Slot]

templates	[Slot]
<b>Initform</b>	(make-hash-table :test 'equalp)
rules	[Slot]
<b>Initform</b>	(make-hash-table :test 'equalp)
rete	[Slot]
<b>Initform</b>	(exil-rete:make-rete)
agenda	[Slot]
strategies	[Slot]
<b>Initform</b>	‘((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]
<b>Initform</b>	'exil-env::default
watchers	[Slot]
<b>Initform</b>	'(:facts) (:rules) (:activations))
match	[Class]
<b>Package</b>	[exil-env], page 24
<b>Source</b>	[matches], page 11 (Lisp file)
<b>Direct superclasses</b>	standard-object
<b>Direct methods</b>	<ul style="list-style-type: none"> <li>• [simpler-than], page 64 (method)</li> <li>• [newer-than], page 62 (method)</li> <li>• [activate-rule], page 35 (method)</li> <li>• print-object</li> <li>• [match-equal-p], page 61 (method)</li> <li>• [timestamp], page 66 (method)</li> <li>• [match-token], page 61 (method)</li> <li>• [match-rule], page 61 (method)</li> </ul>
<b>Direct slots</b>	
rule	[Slot]
<b>Initargs</b>	:rule
<b>Initform</b>	(error "match rule has to be specified")
<b>Readers</b>	[match-rule], page 61 (generic function)
token	[Slot]
<b>Initargs</b>	:token
<b>Initform</b>	(error "match token has to be specified")
<b>Readers</b>	[match-token], page 61 (generic function)

timestamp	[Slot]
<b>Initargs</b> :timestamp	
<b>Initform</b> (get-internal-real-time)	
<b>Readers</b> [timestamp], page 66 (generic function)	
memory-node	[Class]
<b>Package</b> [exil-rete], page 19	
<b>Source</b> [rete-generic-node], page 7 (Lisp file)	
<b>Direct superclasses</b> [node], page 74 (class)	
<b>Direct subclasses</b> <ul style="list-style-type: none"> <li>• [alpha-memory-node], page 67 (class)</li> <li>• [beta-memory-node], page 70 (class)</li> <li>• [beta-negative-node], page 71 (class)</li> </ul>	
<b>Direct methods</b> <ul style="list-style-type: none"> <li>• [add-item], page 54 (method)</li> <li>• items</li> <li>• [items], page 60 (method)</li> </ul>	
<b>Direct slots</b>	
items	[Slot]
<b>Readers</b> [items], page 60 (generic function)	
<b>Writers</b> [(setf items)], page 60 (generic function)	
node	[Class]
<b>Package</b> [exil-rete], page 19	
<b>Source</b> [rete-generic-node], page 7 (Lisp file)	
<b>Direct superclasses</b> [described-object], page 72 (class)	
<b>Direct subclasses</b> <ul style="list-style-type: none"> <li>• [memory-node], page 74 (class)</li> <li>• [alpha-node], page 67 (class)</li> <li>• [beta-node], page 71 (class)</li> </ul>	
<b>Direct methods</b> <ul style="list-style-type: none"> <li>• [inactivate], page 60 (method)</li> <li>• [inactivate-children], page 60 (method)</li> <li>• [activate-children], page 54 (method)</li> <li>• [add-children], page 54 (method)</li> <li>• [add-child], page 54 (method)</li> <li>• [node-equal-p], page 63 (method)</li> <li>• children</li> <li>• [children], page 56 (method)</li> </ul>	

<b>Direct slots</b>	
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)
<b>Direct superclasses</b>	
standard-object	
<b>Direct methods</b>	
<ul style="list-style-type: none"> <li>• beta-top-node</li> <li>• [beta-top-node], page 55 (method)</li> <li>• [alpha-top-node], page 55 (method)</li> </ul>	
<b>Direct slots</b>	
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)
<b>Direct superclasses</b>	
[alpha-node], page 67 (class)	
<b>Direct subclasses</b>	
<ul style="list-style-type: none"> <li>• [simple-fact-test-node], page 76 (class)</li> <li>• [simple-fact-subtop-node], page 75 (class)</li> </ul>	
<b>Direct methods</b>	
[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)
<b>Direct superclasses</b>	
<ul style="list-style-type: none"> <li>• [simple-fact-alpha-node], page 75 (class)</li> <li>• [alpha-subtop-node], page 68 (class)</li> </ul>	
<b>Direct methods</b>	
[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**

<code>template-name</code>	[Slot]
<b>Initargs</b> <code>:tmpl-name</code>	
<b>Readers</b> <code>[tmpl-name]</code> , page 43 (generic function)	
<code>slot-default</code>	[Slot]
<b>Allocation</b> <code>:class</code>	
<code>slots</code>	[Slot]
<b>Initargs</b> <code>:slots</code>	
<b>Readers</b> <code>[slots]</code> , 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**

<code>current-field-to-test</code>	[Slot]
<b>Initargs</b> <code>:current-field</code>	
<b>Initform</b> (error "current-field slot has to be specified")	
<b>Readers</b> <code>[current-field]</code> , page 56 (generic function)	
<code>previous-condition-number</code>	[Slot]
tells, how many conditions back i must go	
<b>Initargs</b> <code>:previous-condition</code>	

<b>Initform</b>	0	
<b>Readers</b>	[previous-condition], page 63 (generic function)	
previous-field-to-test		[Slot]
<b>Initargs</b>	:previous-field	
<b>Initform</b>	(error "previous-field slot has to be specified")	
<b>Readers</b>	[previous-field], page 63 (generic function)	
token		[Class]
<b>Package</b>	[exil-rete], page 19	
<b>Source</b>	[tokens], page 7 (Lisp file)	
<b>Direct superclasses</b>	standard-object	
<b>Direct subclasses</b>	[empty-token], page 72 (class)	
<b>Direct methods</b>	<ul style="list-style-type: none"> <li>• [activate], page 53 (method)</li> <li>• [get-bad-wmes], page 58 (method)</li> <li>• [activate], page 53 (method)</li> <li>• [perform-join-tests], page 63 (method)</li> <li>• [perform-join-test], page 63 (method)</li> <li>• [inactivate], page 59 (method)</li> <li>• [broken-match], page 55 (method)</li> <li>• [activate], page 53 (method)</li> <li>• [complete-match], page 56 (method)</li> <li>• [token-&gt;list], page 43 (method)</li> <li>• print-object</li> <li>• [includes-p], page 60 (method)</li> <li>• [token-equal-p], page 44 (method)</li> <li>• [includes-p], page 60 (method)</li> <li>• [previous-wme], page 64 (method)</li> <li>• negative-wmes</li> <li>• [negative-wmes], page 62 (method)</li> <li>• [wme], page 67 (method)</li> <li>• [parent], page 63 (method)</li> </ul>	
<b>Direct slots</b>		
parent		[Slot]
<b>Initargs</b>	:parent	
<b>Readers</b>	[parent], page 63 (generic function)	
wme		[Slot]
<b>Initargs</b>	:wme	
<b>Initform</b>	(error "wme slot has to be specified")	
<b>Readers</b>	[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

activations.lisp ..... 11

### E

environment.lisp ..... 12

exil.asd ..... 3

export.lisp ..... 13

### F

facts.lisp ..... 5

File, Lisp, activations.lisp ..... 11

File, Lisp, environment.lisp ..... 12

File, Lisp, exil.asd ..... 3

File, Lisp, export.lisp ..... 13

File, Lisp, facts.lisp ..... 5

File, Lisp, matches.lisp ..... 11

File, Lisp, packages.lisp ..... 3

File, Lisp, patterns.lisp ..... 6

File, Lisp, rete-alpha-part.lisp ..... 8

File, Lisp, rete-beta-part.lisp ..... 9

File, Lisp, rete-generic-node.lisp ..... 7

File, Lisp, rete-net-creation.lisp ..... 10

File, Lisp, rules.lisp ..... 6

File, Lisp, strategies.lisp ..... 11

File, Lisp, templates.lisp ..... 4

File, Lisp, tokens.lisp ..... 7

File, Lisp, utils.lisp ..... 3

### L

Lisp File, activations.lisp ..... 11

Lisp File, environment.lisp ..... 12

Lisp File, exil.asd ..... 3

Lisp File, export.lisp ..... 13

Lisp File, facts.lisp ..... 5

Lisp File, matches.lisp ..... 11

Lisp File, packages.lisp ..... 3

Lisp File, patterns.lisp ..... 6

Lisp File, rete-alpha-part.lisp ..... 8

Lisp File, rete-beta-part.lisp ..... 9

Lisp File, rete-generic-node.lisp ..... 7

Lisp File, rete-net-creation.lisp ..... 10

Lisp File, rules.lisp ..... 6

Lisp File, strategies.lisp ..... 11

Lisp File, templates.lisp ..... 4

Lisp File, tokens.lisp ..... 7

Lisp File, utils.lisp ..... 3

### M

matches.lisp ..... 11

### P

packages.lisp ..... 3

patterns.lisp ..... 6

### R

rete-alpha-part.lisp ..... 8

rete-beta-part.lisp ..... 9

rete-generic-node.lisp ..... 7

rete-net-creation.lisp ..... 10

rules.lisp ..... 6

### S

strategies.lisp ..... 11

### T

templates.lisp ..... 4

tokens.lisp ..... 7

### U

utils.lisp ..... 3

## A.2 Functions

(	
(setf assoc-value) .....	32
(setf beta-top-node) .....	55
(setf children) .....	55, 56
(setf cpl-assoc-val) .....	32
(setf description) .....	57
(setf get-network) .....	59
(setf items) .....	60, 61
(setf match-var) .....	40
(setf memory) .....	61
(setf negated-p) .....	40
(setf negative-wmes) .....	62
(setf networks) .....	62
(setf parent) .....	63
(setf productions) .....	64
(setf tests) .....	65
(setf tmpl-fact-slot-value) .....	43
(setf tmpl-object-slot-value) .....	66

## A

activate .....	53, 54
activate-children .....	54
activate-memory .....	54
activate-rule .....	35
activations .....	35
add-child .....	54
add-children .....	54
add-fact .....	31
add-fact-group .....	31
add-item .....	54
add-match .....	35, 36
add-production .....	54, 55
add-rule .....	31
add-strategy .....	36
add-template .....	31
add-wme .....	36
agenda .....	36
alistp .....	31
alpha-memory .....	55
alpha-top-node .....	55
assert .....	29
assert% .....	50
assert-group% .....	50
assoc-key .....	31
assoc-value .....	32
atom-equal-p .....	36
atom-position .....	36

## B

beta-memory .....	55
beta-top-node .....	55
breadth-strategy .....	50
broken-match .....	55

## C

children .....	55, 56
class-slot-value .....	32
clear .....	32

clips->nonclips-mod-list .....	50
clips-mod-list-p .....	50
clips-slot->slot-des% .....	50
clips-slot-spec-p .....	50
clips-tmpl-slot-spec-p .....	50
complete-match .....	56
completely-reset-environment .....	50
complexity-strategy .....	51
conditions .....	36, 37
constant-test .....	32
copy-fact .....	37
cpl-assoc-val .....	32
create-alpha-net .....	56
create-alpha-net% .....	56
current-field .....	56
current-strategy .....	56, 57
current-strategy-name .....	57

## D

defenv .....	49
deffacts .....	29
defrule .....	29
defstrategy .....	29
deftemplate .....	29
delete-production .....	57
depth-strategy .....	51
description .....	57
diff-delete .....	29
doplist .....	29

## E

every-couple .....	32
exil-env-accessor .....	49
exil-env-accessors .....	49
exil-env-reader .....	50
exil-env-writer .....	50
exil-equal-p .....	37
exil-weak-equal-p .....	37
ext-delete .....	29
ext-pushnew .....	30
extract-conditions% .....	51

## F

fact .....	38
fact-description .....	38
fact-equal-p .....	38
fact-groups .....	38
fact-slot .....	38, 39
facts .....	32, 39
find-atom .....	39
find-atom-in-cond-list% .....	51
find-fact .....	32
find-rule .....	32
find-template .....	39
find-test-node .....	57
find/create-join-node .....	57
find/create-neg-node .....	57, 58
find/create-test-node .....	58

find/create-test-node%	58
from-keyword	33
Function, (setf assoc-value)	32
Function, (setf cpl-assoc-val)	32
Function, add-fact	31
Function, add-fact-group	31
Function, add-rule	31
Function, add-template	31
Function, alistp	31
Function, assert%	50
Function, assert-group%	50
Function, assoc-key	31
Function, assoc-value	32
Function, breadth-strategy	50
Function, class-slot-value	32
Function, clear	32
Function, clips->nonclips-mod-list	50
Function, clips-mod-list-p	50
Function, clips-slot->slot-des%	50
Function, clips-slot-spec-p	50
Function, clips-tmpl-slot-spec-p	50
Function, completely-reset-environment	50
Function, complexity-strategy	51
Function, constant-test	32
Function, cpl-assoc-val	32
Function, depth-strategy	51
Function, every-couple	32
Function, extract-conditions%	51
Function, facts	32
Function, find-atom-in-cond-list%	51
Function, find-fact	32
Function, find-rule	32
Function, from-keyword	33
Function, get-variable-bindings	51
Function, halt	33
Function, intern	33
Function, make-fact	33
Function, make-match	51
Function, make-pattern	33
Function, make-rete	33
Function, make-template	33
Function, make-test	51
Function, make-tmpl-fact	51
Function, make-tmpl-obj-clips	51
Function, make-tmpl-obj-nonclips	51
Function, make-tmpl-object	51
Function, make-tmpl-pattern	51
Function, modify-fact	33
Function, my-position	52
Function, nonclips-mod-list-p	52
Function, nonclips-slot->slot-des%	52
Function, nonclips-slot-spec-p	52
Function, plistp	33
Function, ppdefrule%	52
Function, rem-fact	33
Function, rem-fact-group	33
Function, rem-rule	34
Function, reset	34
Function, reset-environment	34
Function, reset-facts	34
Function, retract%	52
Function, retract-all	34
Function, run	34
Function, select	34
Function, set-clips-mode	52

Function, simplicity-strategy	52
Function, slot->slot-designator%	52
Function, slot-spec-p	52
Function, slots->slot-designators%	52
Function, step	34
Function, string-append	34
Function, subsets	34
Function, substitute-variables	52
Function, symbol-append	35
Function, tmpl-fact-specification-p	53
Function, tmpl-object-specification-p	53
Function, tmpl-pattern-specification-p	53
Function, tmpl-slot-spec-p	53
Function, to-keyword	35
Function, to-list	35
Function, to-list-of-lists	35
Function, to-mod-spec-list	53
Function, var-or-equal-p	35
Function, variable-p	35

## G

Generic Function, (setf beta-top-node)	55
Generic Function, (setf children)	55
Generic Function, (setf description)	57
Generic Function, (setf get-network)	59
Generic Function, (setf items)	60
Generic Function, (setf match-var)	40
Generic Function, (setf memory)	61
Generic Function, (setf negated-p)	40
Generic Function, (setf negative-wmes)	62
Generic Function, (setf networks)	62
Generic Function, (setf parent)	63
Generic Function, (setf productions)	64
Generic Function, (setf tests)	65
Generic Function, (setf tmpl-fact-slot-value)	43
Generic Function, (setf tmpl-object-slot-value)	66
Generic Function, activate	53
Generic Function, activate-children	54
Generic Function, activate-memory	54
Generic Function, activate-rule	35
Generic Function, activations	35
Generic Function, add-child	54
Generic Function, add-children	54
Generic Function, add-item	54
Generic Function, add-match	35
Generic Function, add-production	54
Generic Function, add-strategy	36
Generic Function, add-wme	36
Generic Function, agenda	36
Generic Function, alpha-memory	55
Generic Function, alpha-top-node	55
Generic Function, atom-equal-p	36
Generic Function, atom-position	36
Generic Function, beta-memory	55
Generic Function, beta-top-node	55
Generic Function, broken-match	55
Generic Function, children	55
Generic Function, complete-match	56
Generic Function, conditions	36
Generic Function, copy-fact	37
Generic Function, create-alpha-net	56

Generic Function, <code>create-alpha-net%</code> .....	56	Generic Function, <code>rules</code> .....	42
Generic Function, <code>current-field</code> .....	56	Generic Function, <code>select-activation</code> .....	42
Generic Function, <code>current-strategy</code> .....	56	Generic Function, <code>set-strategy</code> .....	42
Generic Function, <code>current-strategy-name</code> .....	57	Generic Function, <code>set-watcher</code> .....	42
Generic Function, <code>delete-production</code> .....	57	Generic Function, <code>simple-fact-key-name</code> .....	64
Generic Function, <code>description</code> .....	57	Generic Function, <code>simpler-than</code> .....	64
Generic Function, <code>exil-equal-p</code> .....	37	Generic Function, <code>slots</code> .....	42
Generic Function, <code>exil-weak-equal-p</code> .....	37	Generic Function, <code>strategies</code> .....	64
Generic Function, <code>fact</code> .....	38	Generic Function, <code>symbol-name</code> .....	42
Generic Function, <code>fact-description</code> .....	38	Generic Function, <code>templates</code> .....	43
Generic Function, <code>fact-equal-p</code> .....	38	Generic Function, <code>test</code> .....	65
Generic Function, <code>fact-groups</code> .....	38	Generic Function, <code>test-equal-p</code> .....	65
Generic Function, <code>fact-slot</code> .....	38	Generic Function, <code>tested-field</code> .....	65
Generic Function, <code>facts</code> .....	39	Generic Function, <code>tested</code> .....	65
Generic Function, <code>find-atom</code> .....	39	Generic Function, <code>tests-equal-p</code> .....	65
Generic Function, <code>find-template</code> .....	39	Generic Function, <code>timestamp</code> .....	66
Generic Function, <code>find-test-node</code> .....	57	Generic Function, <code>tmpl-fact-slot-value</code> .....	43
Generic Function, <code>find/create-join-node</code> .....	57	Generic Function, <code>tmpl-name</code> .....	43
Generic Function, <code>find/create-neg-node</code> .....	57	Generic Function, <code>tmpl-object-equal-p</code> .....	66
Generic Function, <code>find/create-test-node</code> .....	58	Generic Function, <code>tmpl-object-slot-value</code> .....	66
Generic Function, <code>find/create-test-node%</code> .....	58	Generic Function, <code>tmpl-pattern-slot-value</code> .....	66
Generic Function, <code>get-bad-wmes</code> .....	58	Generic Function, <code>token</code> .....	66
Generic Function, <code>get-intercondition-tests%</code> ...	58	Generic Function, <code>token-&gt;list</code> .....	43
Generic Function, <code>get-intracondition-tests%</code> ...	58	Generic Function, <code>token-equal-p</code> .....	43
Generic Function, <code>get-join-tests-from-condition</code> .....	59	Generic Function, <code>unset-watcher</code> .....	44
Generic Function, <code>get-network</code> .....	59	Generic Function, <code>unwatch-all</code> .....	44
Generic Function, <code>get/initialize-network</code> .....	59	Generic Function, <code>value</code> .....	66
Generic Function, <code>has-slot-p</code> .....	39	Generic Function, <code>variable-bindings</code> .....	67
Generic Function, <code>hash-&gt;list</code> .....	39	Generic Function, <code>watch-all</code> .....	44
Generic Function, <code>inactivate</code> .....	59	Generic Function, <code>watched-p</code> .....	44
Generic Function, <code>inactivate-children</code> .....	60	Generic Function, <code>watchers</code> .....	67
Generic Function, <code>includes-p</code> .....	60	Generic Function, <code>weak-symbol-equal-p</code> .....	44
Generic Function, <code>initialize-network</code> .....	60	Generic Function, <code>wme</code> .....	67
Generic Function, <code>is-watcher</code> .....	60	<code>get-bad-wmes</code> .....	58
Generic Function, <code>items</code> .....	60	<code>get-intercondition-tests%</code> .....	58
Generic Function, <code>make-rule</code> .....	39	<code>get-intracondition-tests%</code> .....	58, 59
Generic Function, <code>match-equal-p</code> .....	61	<code>get-join-tests-from-condition</code> .....	59
Generic Function, <code>match-rule</code> .....	61	<code>get-network</code> .....	59
Generic Function, <code>match-token</code> .....	61	<code>get-variable-bindings</code> .....	51
Generic Function, <code>match-var</code> .....	40	<code>get/initialize-network</code> .....	59
Generic Function, <code>memory</code> .....	61		
Generic Function, <code>modify%</code> .....	61		
Generic Function, <code>name</code> .....	40		
Generic Function, <code>negated-p</code> .....	40		
Generic Function, <code>negative-wmes</code> .....	62		
Generic Function, <code>networks</code> .....	62		
Generic Function, <code>new-production</code> .....	40		
Generic Function, <code>newer-than</code> .....	62		
Generic Function, <code>node-equal-p</code> .....	62		
Generic Function, <code>parent</code> .....	63		
Generic Function, <code>pattern</code> .....	40		
Generic Function, <code>pattern-equal-p</code> .....	41		
Generic Function, <code>perform-join-test</code> .....	63		
Generic Function, <code>perform-join-tests</code> .....	63		
Generic Function, <code>previous-condition</code> .....	63		
Generic Function, <code>previous-field</code> .....	63		
Generic Function, <code>previous-wme</code> .....	64		
Generic Function, <code>productions</code> .....	64		
Generic Function, <code>rem-wme</code> .....	41		
Generic Function, <code>remove-match</code> .....	41		
Generic Function, <code>remove-matches</code> .....	64		
Generic Function, <code>remove-production</code> .....	41		
Generic Function, <code>rete</code> .....	41		
Generic Function, <code>rule-equal-p</code> .....	42		

## H

<code>halt</code> .....	33
<code>has-slot-p</code> .....	39
<code>hash-&gt;list</code> .....	39

## I

<code>inactivate</code> .....	59, 60
<code>inactivate-children</code> .....	60
<code>includes-p</code> .....	60
<code>initialize-network</code> .....	60
<code>intern</code> .....	33
<code>is-watcher</code> .....	60
<code>items</code> .....	60

## M

<code>mac-exp</code> .....	30
Macro, <code>assert</code> .....	29
Macro, <code>defenv</code> .....	49
Macro, <code>deffacts</code> .....	29
Macro, <code>defrule</code> .....	29



Macro, defstrategy .....	29	Method, add-production .....	55
Macro, deftemplate .....	29	Method, add-strategy .....	36
Macro, diff-delete .....	29	Method, add-wme .....	36
Macro, doplist .....	29	Method, agenda .....	36
Macro, exil-env-accessor .....	49	Method, alpha-memory .....	55
Macro, exil-env-accessors .....	49	Method, alpha-top-node .....	55
Macro, exil-env-reader .....	50	Method, atom-equal-p .....	36
Macro, exil-env-writer .....	50	Method, atom-position .....	36
Macro, ext-delete .....	29	Method, beta-memory .....	55
Macro, ext-pushnew .....	30	Method, beta-top-node .....	55
Macro, mac-exp .....	30	Method, broken-match .....	55
Macro, modify .....	30	Method, children .....	56
Macro, my-pushnew .....	30	Method, complete-match .....	56
Macro, ppdefrule .....	30	Method, conditions .....	37
Macro, push-end .....	30	Method, copy-fact .....	37
Macro, push-update .....	30	Method, create-alpha-net .....	56
Macro, pushnew-end .....	30	Method, create-alpha-net% .....	56
Macro, retract .....	30	Method, current-field .....	56
Macro, setenv .....	50	Method, current-strategy .....	57
Macro, setstrategy .....	30	Method, current-strategy-name .....	57
Macro, undefeffacts .....	31	Method, delete-production .....	57
Macro, undefrule .....	31	Method, description .....	57
Macro, unwatch .....	31	Method, exil-equal-p .....	37
Macro, watch .....	31	Method, exil-weak-equal-p .....	37
make-fact .....	33	Method, fact .....	38
make-match .....	51	Method, fact-description .....	38
make-pattern .....	33	Method, fact-equal-p .....	38
make-rete .....	33	Method, fact-groups .....	38
make-rule .....	39	Method, fact-slot .....	38, 39
make-template .....	33	Method, facts .....	39
make-test .....	51	Method, find-atom .....	39
make-tmpl-fact .....	51	Method, find-template .....	39
make-tmpl-obj-clips .....	51	Method, find-test-node .....	57
make-tmpl-obj-nonclips .....	51	Method, find/create-join-node .....	57
make-tmpl-object .....	51	Method, find/create-neg-node .....	58
make-tmpl-pattern .....	51	Method, find/create-test-node .....	58
match-equal-p .....	61	Method, find/create-test-node% .....	58
match-rule .....	61	Method, get-bad-wmes .....	58
match-token .....	61	Method, get-intercondition-tests% .....	58
match-var .....	40	Method, get-intracondition-tests% .....	59
memory .....	61	Method, get-join-tests-from-condition .....	59
Method, (setf beta-top-node) .....	55	Method, get-network .....	59
Method, (setf children) .....	56	Method, get/initialize-network .....	59
Method, (setf description) .....	57	Method, has-slot-p .....	39
Method, (setf get-network) .....	59	Method, hash->list .....	39
Method, (setf items) .....	61	Method, inactivate .....	59, 60
Method, (setf match-var) .....	40	Method, inactivate-children .....	60
Method, (setf memory) .....	61	Method, includes-p .....	60
Method, (setf negated-p) .....	40	Method, initialize-network .....	60
Method, (setf negative-wmes) .....	62	Method, is-watcher .....	60
Method, (setf networks) .....	62	Method, items .....	60
Method, (setf parent) .....	63	Method, make-rule .....	39
Method, (setf productions) .....	64	Method, match-equal-p .....	61
Method, (setf tests) .....	65	Method, match-rule .....	61
Method, (setf tmpl-fact-slot-value) .....	43	Method, match-token .....	61
Method, (setf tmpl-object-slot-value) .....	66	Method, match-var .....	40
Method, activate .....	53, 54	Method, memory .....	61
Method, activate-children .....	54	Method, modify% .....	61
Method, activate-memory .....	54	Method, name .....	40
Method, activate-rule .....	35	Method, negated-p .....	40
Method, activations .....	35	Method, negative-wmes .....	62
Method, add-child .....	54	Method, networks .....	62
Method, add-children .....	54	Method, new-production .....	40
Method, add-item .....	54	Method, newer-than .....	62
Method, add-match .....	36	Method, node-equal-p .....	62, 63

Method, parent.....	63
Method, pattern.....	41
Method, pattern-equal-p.....	41
Method, perform-join-test.....	63
Method, perform-join-tests.....	63
Method, previous-condition.....	63
Method, previous-field.....	63
Method, previous-wme.....	64
Method, productions.....	64
Method, rem-wme.....	41
Method, remove-match.....	41
Method, remove-matches.....	64
Method, remove-production.....	41
Method, rete.....	41
Method, rule-equal-p.....	42
Method, rules.....	42
Method, select-activation.....	42
Method, set-strategy.....	42
Method, set-watcher.....	42
Method, simple-fact-key-name.....	64
Method, simpler-than.....	64
Method, slots.....	42
Method, strategies.....	65
Method, symbol-name.....	43
Method, templates.....	43
Method, test.....	65
Method, test-equal-p.....	65
Method, tested-field.....	65
Method, tests.....	65
Method, tests-equal-p.....	66
Method, timestamp.....	66
Method, tpl-fact-slot-value.....	43
Method, tpl-name.....	43
Method, tpl-object-equal-p.....	66
Method, tpl-object-slot-value.....	66
Method, tpl-pattern-slot-value.....	66
Method, token.....	66
Method, token->list.....	43
Method, token-equal-p.....	43, 44
Method, unset-watcher.....	44
Method, unwatch-all.....	44
Method, value.....	67
Method, variable-bindings.....	67
Method, watch-all.....	44
Method, watched-p.....	44
Method, watchers.....	67
Method, weak-symbol-equal-p.....	44
Method, wme.....	67
modify.....	30
modify%.....	61
modify-fact.....	33
my-position.....	52
my-pushnew.....	30

## N

name.....	40
negated-p.....	40
negative-wmes.....	62
networks.....	62
new-production.....	40
newer-than.....	62
node-equal-p.....	62, 63
nonclips-mod-list-p.....	52

nonclips-slot->slot-des%.....	52
nonclips-slot-spec-p.....	52

## P

parent.....	63
pattern.....	40, 41
pattern-equal-p.....	41
perform-join-test.....	63
perform-join-tests.....	63
plislp.....	33
ppdefrule.....	30
ppdefrule%.....	52
previous-condition.....	63
previous-field.....	63
previous-wme.....	64
productions.....	64
push-end.....	30
push-update.....	30
pushnew-end.....	30

## R

rem-fact.....	33
rem-fact-group.....	33
rem-rule.....	34
rem-wme.....	41
remove-match.....	41
remove-matches.....	64
remove-production.....	41
reset.....	34
reset-environment.....	34
reset-facts.....	34
rete.....	41
retract.....	30
retract%.....	52
retract-all.....	34
rule-equal-p.....	42
rules.....	42
run.....	34

## S

select.....	34
select-activation.....	42
set-clips-mode.....	52
set-strategy.....	42
set-watcher.....	42
setenv.....	50
setstrategy.....	30
simple-fact-key-name.....	64
simpler-than.....	64
simplicity-strategy.....	52
slot->slot-designator%.....	52
slot-spec-p.....	52
slots.....	42
slots->slot-designators%.....	52
step.....	34
strategies.....	64, 65
string-append.....	34
subsets.....	34
substitute-variables.....	52
symbol-append.....	35
symbol-name.....	42, 43

**T**

templates .....	43
test .....	65
test-equal-p .....	65
tested-field .....	65
tests .....	65
tests-equal-p .....	65, 66
timestamp .....	66
tmpl-fact-slot-value .....	43
tmpl-fact-specification-p .....	53
tmpl-name .....	43
tmpl-object-equal-p .....	66
tmpl-object-slot-value .....	66
tmpl-object-specification-p .....	53
tmpl-pattern-slot-value .....	66
tmpl-pattern-specification-p .....	53
tmpl-slot-spec-p .....	53
to-keyword .....	35
to-list .....	35
to-list-of-lists .....	35
to-mod-spec-list .....	53
token .....	66
token->list .....	43
token-equal-p .....	43, 44

**U**

undeffacts .....	31
undefrule .....	31
unset-watcher .....	44
unwatch .....	31
unwatch-all .....	44

**V**

value .....	66, 67
var-or-equal-p .....	35
variable-bindings .....	67
variable-p .....	35

**W**

watch .....	31
watch-all .....	44
watched-p .....	44
watchers .....	67
weak-symbol-equal-p .....	44
wme .....	67



## A.3 Variables

### \*

*clips-mode*	49
*current-environment*	49
*environments*	49
*exil-running*	49

### A

activations	46
agenda	73
alpha-memory	69, 70
alpha-top-node	75

### B

beta-top-node	75
---------------	----

### C

children	75
conditions	46
current-field-to-test	77
current-strategy-name	73

### D

dataflow-networks	69
description	72
desired-value	69

### F

fact	47
fact-groups	72
facts	72

### I

items	72, 74
-------	--------

### M

match-variable	46
----------------	----

### N

name	46, 48
negated	46
negative-wmes	79

### P

parent	71, 78
pattern	47
previous-condition-number	77
previous-field-to-test	78
productions	71

### R

rete	73
rule	73
rules	73

### S

simple-fact-key-name	69
Slot, activations	46
Slot, agenda	73
Slot, alpha-memory	69, 70
Slot, alpha-top-node	75
Slot, beta-top-node	75
Slot, children	75
Slot, conditions	46
Slot, current-field-to-test	77
Slot, current-strategy-name	73
Slot, dataflow-networks	69
Slot, description	72
Slot, desired-value	69
Slot, fact	47
Slot, fact-groups	72
Slot, facts	72
Slot, items	72, 74
Slot, match-variable	46
Slot, name	46, 48
Slot, negated	46
Slot, negative-wmes	79
Slot, parent	71, 78
Slot, pattern	47
Slot, previous-condition-number	77
Slot, previous-field-to-test	78
Slot, productions	71
Slot, rete	73
Slot, rule	73
Slot, rules	73
Slot, simple-fact-key-name	69
Slot, slot-default	49, 77
Slot, slots	48, 77
Slot, strategies	73
Slot, template-name	77
Slot, templates	73
Slot, tested-field	69
Slot, tests	70
Slot, timestamp	74
Slot, token	73
Slot, watchers	73
Slot, wme	72, 78
slot-default	49, 77
slots	48, 77
Special Variable, *clips-mode*	49
Special Variable, *current-environment*	49
Special Variable, *environments*	49
Special Variable, *exil-running*	49
strategies	73

### T

template-name	77
templates	73
tested-field	69
tests	70
timestamp	74
token	73

### W

watchers	73
wme	72, 78

## A.4 Data types

### A

alpha-memory-node .....	67
alpha-node .....	67
alpha-subtop-node .....	68
alpha-test-node .....	68
alpha-top-node .....	69

### B

beta-join-node .....	70
beta-memory-node .....	70
beta-negative-node .....	71
beta-node .....	71
beta-top-node .....	71

### C

Class, alpha-memory-node .....	67
Class, alpha-node .....	67
Class, alpha-subtop-node .....	68
Class, alpha-test-node .....	68
Class, alpha-top-node .....	69
Class, beta-join-node .....	70
Class, beta-memory-node .....	70
Class, beta-negative-node .....	71
Class, beta-node .....	71
Class, beta-top-node .....	71
Class, described-object .....	72
Class, empty-token .....	72
Class, exil-environment .....	72
Class, fact .....	45
Class, match .....	73
Class, memory-node .....	74
Class, node .....	74
Class, pattern .....	45
Class, rete .....	75
Class, rule .....	46
Class, simple-fact .....	47
Class, simple-fact-alpha-node .....	75
Class, simple-fact-subtop-node .....	75
Class, simple-fact-test-node .....	76
Class, simple-pattern .....	47
Class, template .....	48
Class, template-fact .....	48
Class, template-fact-alpha-node .....	76
Class, template-fact-subtop-node .....	76
Class, template-fact-test-node .....	76
Class, template-object .....	76
Class, template-pattern .....	48
Class, test .....	77
Class, token .....	78

### D

described-object .....	72
------------------------	----

### E

empty-token .....	72
-------------------	----

exil .....	1, 26
exil-core .....	16
exil-env .....	24
exil-environment .....	72
exil-rete .....	19
exil-system .....	15
exil-user .....	27
exil-utils .....	15

### F

fact .....	45
------------	----

### M

match .....	73
memory-node .....	74

### N

node .....	74
------------	----

### P

Package, exil .....	26
Package, exil-core .....	16
Package, exil-env .....	24
Package, exil-rete .....	19
Package, exil-system .....	15
Package, exil-user .....	27
Package, exil-utils .....	15
pattern .....	45

### R

rete .....	75
rule .....	46

### S

simple-fact .....	47
simple-fact-alpha-node .....	75
simple-fact-subtop-node .....	75
simple-fact-test-node .....	76
simple-pattern .....	47
System, exil .....	1

### T

template .....	48
template-fact .....	48
template-fact-alpha-node .....	76
template-fact-subtop-node .....	76
template-fact-test-node .....	76
template-object .....	76
template-pattern .....	48
test .....	77
token .....	78