spell User's Manual

Table of Contents

1 Introduction **1**

2 External Protocols **2**

2.1 English Protocol 2
2.2 Word Protocols 3
2.3 Dictionary Protocol 14

Concept index **16**

Function and macro and variable and type index . **17**

Changelog **18**

1 Introduction

The spell library provides protocols and implementations of those protocols for automatic spell-checking and suggestion of likely corrections. In addition to the spelling of words, dictionary entries contain information about word types and properties of the words such as case, gender and number. At the moment, English is the only supported language.

2 External Protocols

This chapter describes the external protocols provided by the spell library.

2.1 English Protocol

This protocol provides the highest level of abstraction within the spell library. The protocol consists of a few functions that perform the most common operations with minimal setup effort.

`english-lookup` [spell] [Function]

string

Return a (possibly empty) list of entries in the English dictionary the spelling of which matches *string*.

The returned entries are instances of [Class spell|word], page 3. More than one entry in the dictionary can match *string* due to different word types or multiple combinations of properties within a word type that share a common spelling.

Note that clients which are interested only in the validity of *string* and not in the matching dictionary entries can treat the return value as a generalized Boolean.

Examples:

```
(spell:english-lookup "lithp")
⇒ NIL

(spell:english-lookup "lisp")
⇒ (#<EXPLICIT-BASE-VERB "lisp" person:ANY number:ANY ... infinitive:SELF {1004ED
   #<EXPLICIT-BASE-NOUN "lisp" number:SINGULAR case:NIL gender:NIL {1004ED3533}>)
```

`english-check-paragraph` [spell] [Function]

string

Check the paragraph of English text *string* for spelling errors and return a list of detected errors.

The return value of this function is a possibly empty list of entries of the form (*start-index* . *end-index*). The *start-index* and *end-index* within each entry designate a sub-string of *string* that corresponds to a misspelled word.

Example:

```
(spell:english-check-paragraph "In Polish, a horse is koń, and in German, it's
das Pferd.")
⇒ ((22 . 25) (47 . 50) (51 . 56))
```

The three pairs in the result correspond to sub-strings that are not valid English words:

```
(loop :with string = "In Polish, a horse is koń, and in German, it's
das Pferd."
      :for (start . end) :in (spell:english-check-paragraph string)
      :collect (subseq string start end))
⇒ ("koń" "das" "Pferd")
```

2.2 Word Protocols

Entries in spell dictionaries are conceptually pair: an (implicit) string that is the spelling and a word object that characterizes the word type and the type-specific properties of the word. Word objects are instance of subclasses of [Class spell|word], page 3. spell defines the following sub-classes of [Class spell|word], page 3, and associated readers:

```
[Class spell|word], page 3
+-[Class spell|adjective], page 3
+-[Class spell|adverb], page 4
+-[Class spell|conjunction], page 4
| `-[Class spell|subordinate], page 4
+-[Class spell|determiner], page 5
| +-[Class spell|article], page 5
| +-[Class spell|demonstrative-adjective], page 6
| +-[Class spell|interrogative-adjective], page 6
| +-[Class spell|possessive-adjective], page 6
| `-[Class spell|quantifier], page 7
+-[Class spell|interjection], page 7
+-[Class spell|noun], page 8
| +-[Class spell|noun-verb-contraction], page 8
| `-[Class spell|proper-noun], page 9
+-[Class spell|preposition], page 9
+-[Class spell|pronoun], page 10
| +-[Class spell|demonstrative-pronoun], page 10
| +-[Class spell|personal-pronoun], page 11
| +-[Class spell|possessive-pronoun], page 11
| `-[Class spell|reflexive-pronoun], page 12
`-[Class spell|verb], page 12
  +-[Class spell|noun-verb-contraction], page 8
  `-[Class spell|verb-verb-contraction], page 13
```

Word Class word

`word` `[spell]` [Class]

() Superclass for all word classes.

`base` `[spell]` [Generic Function]

word Return the base word of *word*.

For words of type noun, the base word is usually the nominative, singular variant of the word.

For words of type verb, the base word is usually the infinitive variant of the word.

Word Class adjective

`adjective` `[spell]` [Class]

([Class spell|word], page 3) Instances represent adjectives, that is a property of a noun together with the degree of that property.

`degree` `[spell]` [Generic Function]

word Return a keyword which indicates the degree for *word*.

Possible values are `:positive`, `:comparative` and `:superlative`.

Examples

zinkier
```
#<EXPLICIT-BASE-ADJECTIVE "zincy"
degree:COMPARATIVE {1003D28FF3}>
```

wroughter
```
#<EXPLICIT-BASE-ADJECTIVE "wrought"
degree:COMPARATIVE {1003CD6E33}>
```

well-wornest
```
#<EXPLICIT-BASE-ADJECTIVE "well-worn"
degree:SUPERLATIVE {1003BC5783}>
```

Word Class adverb

`adverb` `[spell]` [Class]

([Class spell|word], page 3) Instances represent adverb, that is a property of a verb.

Examples

unstoppably
```
#<EXPLICIT-BASE-ADVERB "unstoppably" {1003895EC3}>
```

twistingly
```
#<EXPLICIT-BASE-ADVERB "twistingly" {100378F923}>
```

together
```
#<EXPLICIT-BASE-ADVERB "together" {10035D3933}>
```

Word Class conjunction

`conjunction` `[spell]` [Class]

([Class spell|word], page 3) Direct instances represent conjunctions, that is words which connect sentences or clauses.

Examples

seeing
```
#<EXPLICIT-BASE-CONJUNCTION "seeing" {10021499C3}>
```

providing
```
#<EXPLICIT-BASE-CONJUNCTION "provided"
{1001B82F73}>
```

either
```
#<EXPLICIT-BASE-CONJUNCTION "either" {1006E22EF3}>
```

Word Class subordinate

`subordinate` `[spell]` [Class]

([Class spell|word], page 3, [Class spell|conjunction], page 4) Instances represent subordinates, that is conjunctions which connect a dependent and an independent clause.

Examples

whether `#<EXPLICIT-BASE-SUBORDINATE "whether" {1003C041E3}>`

that `#<EXPLICIT-BASE-SUBORDINATE "that" {10034E4F83}>`

if `#<EXPLICIT-BASE-SUBORDINATE "if" {1006E22F83}>`

Word Class determiner

`determiner` `[spell]` [Class]

([Class spell|word], page 3) Instances are determiners, that is words which determine one or more aspect, such as the grammatical number, of a noun. Also a superclass for several more specific determiner classes.

`number` `[spell]` [Generic Function]

word Return a keyword which indicates the number for entities related to *word*.

If *word* is a verb, the keyword indicates the number of agents who perform the action.

If *word* is a noun, the keyword indicates the number of objects or people the noun refers to. Similarly for pronouns.

If *word* is a determiner (or a more specific word sub type), the number refers to the associated noun.

Possible values are `:any`, `:singular` and `:plural`.

Word Class article

`article` `[spell]` [Class]

([Class spell|word], page 3, [Class spell|determiner], page 5) Instances are articles, that is determiners which specify whether a noun refers to a particular or an arbitrary entity.

`determinate` `[spell]` [Generic Function]

word Return a Boolean which indicates whether the noun associated with *word* refers to particular or an arbitrary entity.

Examples

the `#<EXPLICIT-BASE-ARTICLE "the" number:SINGULAR determinate:T {10035318E3}>`

a `#<EXPLICIT-BASE-ARTICLE "a" number:SINGULAR determinate:NIL {1006E22F33}>`

an `#<EXPLICIT-BASE-ARTICLE "an" number:SINGULAR determinate:NIL {1006E22F43}>`

Word Class demonstrative-adjective

`demonstrative-adjective` `[spell]` [Class]
([Class spell|word], page 3, [Class spell|determiner], page 5) Instances represent demonstrative adjectives, that is determiners which highlight the fact that an associated noun is identifiable from the context.

Examples

this
```
#<EXPLICIT-BASE-DEMONSTRATIVE-ADJECTIVE "this"
number:SINGULAR {1003549873}>
```

these
```
#<EXPLICIT-BASE-DEMONSTRATIVE-ADJECTIVE "these"
number:PLURAL {10035304D3}>
```

that
```
#<EXPLICIT-BASE-DEMONSTRATIVE-ADJECTIVE "that"
number:SINGULAR {1003502D33}>
```

Word Class interrogative-adjective

`interrogative-adjective` `[spell]` [Class]
([Class spell|word], page 3, [Class spell|determiner], page 5) Instances represent interrogative adjectives, that is adjectives

Examples

which
```
#<EXPLICIT-BASE-INTERROGATIVE-ADJECTIVE "which"
number:ANY {1003C05583}>
```

what
```
#<EXPLICIT-BASE-INTERROGATIVE-ADJECTIVE "what"
number:ANY {1003BFD033}>
```

Word Class possessive-adjective

`possessive-adjective` `[spell]` [Class]
([Class spell|word], page 3, [Class spell|determiner], page 5) Instances represent possessive adjectives, that is determiners which mark a noun as being possessed by another entity.

`refnumber` `[spell]` [Generic Function]
word Return a keyword which indicates the grammatical number of the possessing entity referenced by *word*.

Possible values are `:any`, `:singular` and `:plural`.

`person` `[spell]` [Generic Function]
word Return a keyword which indicates the grammatical person of entities associated with *word*.

Possible values are `:any`, `:first`, `:second`, `:third`.

`gender` `[spell]` [Generic Function]

word Return a keyword which indicates the gender of entities related to *word*.

Possible values are `nil`, `:masculine`, `:feminine` and `:neuter`.

Examples

its
```
#<EXPLICIT-BASE-POSSESSIVE-ADJECTIVE
"its" number:ANY gender:ANY person:THIRD
refnumber:SINGULAR {1006E23043}>
```

his
```
#<EXPLICIT-BASE-POSSESSIVE-ADJECTIVE "his"
number:ANY gender:MASCULINE person:THIRD
refnumber:SINGULAR {1006E23053}>
```

her
```
#<EXPLICIT-BASE-POSSESSIVE-ADJECTIVE "her"
number:ANY gender:FEMININE person:THIRD
refnumber:SINGULAR {1006E23063}>
```

Word Class quantifier

`quantifier` `[spell]` [Class]

([Class spell|word], page 3, [Class spell|determiner], page 5) Instances are quantifiers, that is determiners which specify count or amount of a noun.

Examples

enough
```
#<EXPLICIT-BASE-QUANTIFIER "enough" number:ANY
{1006E22E53}>
```

eleven
```
#<EXPLICIT-BASE-QUANTIFIER "eleven" number:PLURAL
{1006E22E63}>
```

either
```
#<EXPLICIT-BASE-QUANTIFIER "either"
number:SINGULAR {1006E22E73}>
```

Word Class interjection

`interjection` `[spell]` [Class]

([Class spell|word], page 3) Instances represent interjections, that is words which interrupt the sequence of words within a sentence or the sequence of sentences.

Examples

bye-bye
```
#<EXPLICIT-BASE-INTERJECTION "bye-bye"
{1006E22343}>
```

blimey
```
#<EXPLICIT-BASE-INTERJECTION "blimey"
{1006E22353}>
```

begorra `#<EXPLICIT-BASE-INTERJECTION "begorra" {1006E22363}>`

Word Class noun

`noun` `[spell]` [Class]
([Class spell|word], page 3) Direct instances represent generic nouns. Also a superclass for more specific noun classes.

The generic function [Generic-Function spell|gender], page 7, can be applied to words of this class.

`case` `[spell]` [Generic Function]
word Return a keyword which indicates the grammatical case of *word*.

Possible values are `:nominative`, `:genitive`, `:accusative` and `:nominative-or-accusative`.

The generic function [Generic-Function spell|number], page 5, can be applied to words of this class.

Examples

yellow-flags `#<EXPLICIT-BASE-NOUN "yellow-flag" number:PLURAL case:NIL gender:NIL {1003CFEAB3}>`

worrywarts `#<EXPLICIT-BASE-NOUN "worrywart" number:PLURAL case:NIL gender:NIL {1003CBA2D3}>`

woodrushes `#<EXPLICIT-BASE-NOUN "woodrush" number:PLURAL case:NIL gender:NIL {1003C9C663}>`

Word Class noun-verb-contraction

`noun-verb-contraction` `[spell]` [Class]
([Class spell|word], page 3, [Class spell|verb], page 12, [Class spell|noun], page 8) Instances are of type noun and also of type verb and are formed as a contraction of one noun and one verb.

Examples

'twould `#<EXPLICIT-BASE-NOUN-VERB-CONTRACTION "it would" person:THIRD number:SINGULAR tense:PRESENT-SIMPLE negative:NIL contraction:NIL strength:WEAK infinitive:NIL case:NIL gender:NEUTER {1006E223E3}>`

'twill
```
#<EXPLICIT-BASE-NOUN-VERB-CONTRACTION "it will"
person:THIRD number:SINGULAR tense:PRESENT-SIMPLE
negative:NIL contraction:NIL strength:WEAK
infinitive:NIL case:NIL gender:NEUTER
{1006E223F3}>
```

'twas
```
#<EXPLICIT-BASE-NOUN-VERB-CONTRACTION "it
was" person:THIRD number:SINGULAR tense:PAST
negative:NIL contraction:NIL strength:STRONG
infinitive:NIL case:NIL gender:NEUTER
{1006E22403}>
```

Word Class proper-noun

`proper-noun` [spell] [Class]

([Class spell|word], page 3, [Class spell|noun], page 8) Instances represent proper nouns, that is generally names.

Examples

Zaragozas
```
#<EXPLICIT-BASE-PROPER-NOUN "Zaragoza"
number:PLURAL case:NIL gender:NIL {1006E22C63}>
```

Trinidad
```
#<EXPLICIT-BASE-PROPER-NOUN "Trinidad"
number:SINGULAR case:NIL gender:NIL {1006E22C73}>
```

Sloughs
```
#<EXPLICIT-BASE-PROPER-NOUN "Slough" number:PLURAL
case:NIL gender:NIL {1006E22C83}>
```

Word Class preposition

`preposition` [spell] [Class]

([Class spell|word], page 3) Instances represent prepositions, that is words which indicate the spatial, temporal, logical, etc. relations between other words.

Examples

although
```
#<EXPLICIT-BASE-PREPOSITION "although"
{1006E211C3}>
```

alongside
```
#<EXPLICIT-BASE-PREPOSITION "alongside"
{1006E211D3}>
```

albeit
```
#<EXPLICIT-BASE-PREPOSITION "albeit" {1006E211E3}>
```

Word Class pronoun

`pronoun` `[spell]` [Class]

([Class spell|word], page 3) Direct instances represent generic pronouns. Also a superclass for several more specific pronoun classes.

`negative` `[spell]` [Generic Function]

word Return a Boolean which indicates whether the meaning of *word* is in some way negative.

If *word* is a verb, a true value indicates that the absence of the action represented by the verb.

If *word* is a pronoun, a true value indicates the complement of the set of entities for which the non negated pronoun would stand in.

The generic function [Generic-Function spell|case], page 8, can be applied to words of this class.

The generic function [Generic-Function spell|gender], page 7, can be applied to words of this class.

The generic function [Generic-Function spell|number], page 5, can be applied to words of this class.

The generic function [Generic-Function spell|person], page 6, can be applied to words of this class.

Examples

everyone
```
#<EXPLICIT-BASE-PRONOUN "everyone"
person:THIRD number:SINGULAR gender:NIL
case:NOMINATIVE-OR-ACCUSATIVE negative:NIL
{1006E21123}>
```

everybody
```
#<EXPLICIT-BASE-PRONOUN "everybody"
person:THIRD number:SINGULAR gender:NIL
case:NOMINATIVE-OR-ACCUSATIVE negative:NIL
{1006E21133}>
```

either
```
#<EXPLICIT-BASE-PRONOUN "either"
person:THIRD number:SINGULAR gender:NIL
case:NOMINATIVE-OR-ACCUSATIVE negative:NIL
{1006E21143}>
```

Word Class demonstrative-pronoun

`demonstrative-pronoun` `[spell]` [Class]

([Class spell|word], page 3, [Class spell|pronoun], page 10) Instances represent demonstrative pronouns, that is shorthand references which highlight an entity that is identifiable from the context.

Examples

there
```
#<EXPLICIT-BASE-DEMONSTRATIVE-PRONOUN "there"
person:NIL number:ANY gender:NIL case:NOMINATIVE
negative:NIL {100351B513}>
```

that
```
#<EXPLICIT-BASE-DEMONSTRATIVE-PRONOUN "that"
person:NIL number:ANY gender:NIL case:NOMINATIVE
negative:NIL {10035032F3}>
```

here
```
#<EXPLICIT-BASE-DEMONSTRATIVE-PRONOUN "here"
person:NIL number:ANY gender:NIL case:NOMINATIVE
negative:NIL {1006E22FA3}>
```

Word Class personal-pronoun

`personal-pronoun` `[spell]` [Class]

([Class spell|word], page 3, [Class spell|pronoun], page 10) Instances represent personal pronouns, that is shorthand references to a noun which designates a person.

Examples

he
```
#<EXPLICIT-BASE-PERSONAL-PRONOUN "he" person:THIRD
number:SINGULAR gender:MASCULINE case:NOMINATIVE
negative:NIL {1006E22E23}>
```

her
```
#<EXPLICIT-BASE-PERSONAL-PRONOUN "her"
person:THIRD number:SINGULAR gender:FEMININE
case:ACCUSATIVE negative:NIL {1006E22E33}>
```

I
```
#<EXPLICIT-BASE-PERSONAL-PRONOUN "I" person:FIRST
number:SINGULAR gender:NIL case:NOMINATIVE
negative:NIL {1006E22E43}>
```

Word Class possessive-pronoun

`possessive-pronoun` `[spell]` [Class]

([Class spell|word], page 3, [Class spell|pronoun], page 10) Instances represent possessive pronouns, that is shorthand references to an object that is possessed by an entity which is identifiable from the context.

The generic function [Generic-Function spell|refnumber], page 6, can be applied to words of this class.

Examples

theirs
```
#<EXPLICIT-BASE-POSSESSIVE-PRONOUN "theirs"
person:THIRD number:ANY gender:NIL case:NOMINATIVE
negative:NIL refnumber:PLURAL {1003507833}>
```

its
```
#<EXPLICIT-BASE-POSSESSIVE-PRONOUN "it"
person:THIRD number:ANY gender:NIL case:NOMINATIVE
negative:NIL refnumber:SINGULAR {1006E23013}>
```

his
```
#<EXPLICIT-BASE-POSSESSIVE-PRONOUN "his"
person:THIRD number:ANY gender:MASCULINE
case:NOMINATIVE negative:NIL refnumber:SINGULAR
{1006E23023}>
```

Word Class reflexive-pronoun

`reflexive-pronoun` `[spell]` [Class]
([Class spell | word], page 3, [Class spell | pronoun], page 10) Instances represent possessive pronouns, that is shorthand references that link the object back to the subject which is identifiable from the context.

Examples

yourselves
```
#<EXPLICIT-BASE-REFLEXIVE-PRONOUN "yourselves"
person:SECOND number:PLURAL gender:NIL
case:NOMINATIVE negative:NIL {1003D106C3}>
```

yourself
```
#<EXPLICIT-BASE-REFLEXIVE-PRONOUN "yourself"
person:SECOND number:SINGULAR gender:NIL
case:NOMINATIVE negative:NIL {1003D10653}>
```

thyself
```
#<EXPLICIT-BASE-REFLEXIVE-PRONOUN "thyself"
person:SECOND number:SINGULAR gender:NIL
case:NOMINATIVE negative:NIL {1003580A13}>
```

Word Class verb

`verb` `[spell]` [Class]
([Class spell | word], page 3) Direct instances represent generic verbs. Also a superclass for more specific verb classes.

`infinitive` `[spell]` [Generic Function]
word Return a keyword which indicates whether and which kind of infinitive *word* is.

Possible values are `nil`, `:self` and `:to`.

`strength` `[spell]` [Generic Function]
word Return a keyword which indicates the grammatical strength of *word*.

Only `verb` instances have the strength property which is either weak or strong. Weak means that the past tense of the verb is formed by adding a suffix while strong means that the past tense form of the verb has a different stem than the infinitive form.

Possible values are `:weak` and `:strong`.

`contraction` `[spell]` [Generic Function]
word Return a Boolean which indicates whether *word* is a contraction of two words.

For instances of the `verb` class a true value of the contraction property always goes together with a true value of the negative property. In other words these contractions consist of the base verb and the word not as in won t.

The generic function [Generic-Function spell|negative], page 10, can be applied to words of this class.

`tense` `[spell]` [Generic Function]
word Return a keyword which indicates the time of the action for *word*.

Possible values are `nil`, `:present-simple`, `:progressive`, `:past`, `:perfect-participle`, `:passive-perfect-participle`.

The generic function [Generic-Function spell|number], page 5, can be applied to words of this class.

The generic function [Generic-Function spell|person], page 6, can be applied to words of this class.

Examples

wannaed
```
#<EXPLICIT-BASE-VERB "wanna" person:ANY
number:ANY tense:PERFECT-PARTICIPLE negative:NIL
contraction:NIL strength:WEAK infinitive:NIL
{1003A6DCC3}>
```

visaing
```
#<EXPLICIT-BASE-VERB "visa" person:ANY number:ANY
tense:PROGRESSIVE negative:NIL contraction:NIL
strength:WEAK infinitive:NIL {10039FAF63}>
```

validating
```
#<EXPLICIT-BASE-VERB "validate" person:ANY
number:ANY tense:PROGRESSIVE negative:NIL
contraction:NIL strength:WEAK infinitive:NIL
{10038FF693}>
```

Word Class verb-verb-contraction

`verb-verb-contraction` `[spell]` [Class]
([Class spell|word], page 3, [Class spell|verb], page 12) Instances are verbs that are formed as a contraction of two verbs.

Examples

might've
```
#<EXPLICIT-BASE-VERB-VERB-CONTRACTION "might
have" person:ANY number:ANY tense:PRESENT-SIMPLE
negative:NIL contraction:NIL strength:WEAK
infinitive:SELF {1006E22F53}>
```

```
may've          #<EXPLICIT-BASE-VERB-VERB-CONTRACTION "may
                have" person:ANY number:ANY tense:PRESENT-SIMPLE
                negative:NIL contraction:NIL strength:WEAK
                infinitive:SELF {1006E22F63}>

could've        #<EXPLICIT-BASE-VERB-VERB-CONTRACTION "could
                have" person:ANY number:ANY tense:PRESENT-SIMPLE
                negative:NIL contraction:NIL strength:WEAK
                infinitive:SELF {1006E22F73}>
```

2.3 Dictionary Protocol

`entry-count` [spell] [Generic Function]

dictionary

Return the number of entries in *dictionary*.

Example

```
(spell:entry-count spell::*english-dictionary*)
⇒ 347488
```

`map-entries` [spell] [Generic Function]

function dictionary

Call *function* for each entry in *dictionary*.

The lambda-list of *function* has to be compatible with `(spelling word)` where *word* is the word object of the entry and *spelling* is the spelling of the entry as a string.

spelling is passed as a separate argument because it cannot be obtained from *word*.

Example:

```
(prog ((i 0))
  (spell:map-entries (lambda (spelling word)
                       (cond ((<= 7000 (incf i) 7005)
                              (format t "~D ~10A ~A~%" i spelling word))
                             ((> i 10005)
                              (return))))
                     spell::*english-dictionary*))
⊣ 7000 Highlander #<EXPLICIT-BASE-NOUN "Highlander" number:SINGULAR case:NIL gen
⊣ 7001 Highlands' #<EXPLICIT-BASE-PROPER-NOUN "Highland" number:PLURAL case:GENI
⊣ 7002 Highlands  #<EXPLICIT-BASE-PROPER-NOUN "Highland" number:PLURAL case:NIL
⊣ 7003 Highland   #<EXPLICIT-BASE-PROPER-NOUN "Highland" number:SINGULAR case:NI
⊣ 7004 Hilary's   #<EXPLICIT-BASE-PROPER-NOUN "Hilary" number:SINGULAR case:GENI
⊣ 7005 Hilarys'   #<EXPLICIT-BASE-PROPER-NOUN "Hilary" number:PLURAL case:GENITI
```

`lookup` [spell] [Generic Function]

string dictionary

Return a (possibly empty) list of entries in *dictionary* the spelling of which matches *string*.

Example:

```
(spell:lookup "bright" spell::*english-dictionary*)
⇒ (#<EXPLICIT-BASE-ADVERB "bright" {101B1CE6B3}>
   #<EXPLICIT-BASE-ADJECTIVE "bright" degree:POSITIVE {101B1CE663}>
   #<EXPLICIT-BASE-NOUN "bright" number:SINGULAR case:NIL gender:NIL {101B1CE633}
```

`insert` `[spell]` [Generic Function]

word string dictionary

Insert *word* into *dictionary* as an entry for *string*.

word must be an instance of [Class spell|word], page 3.

dictionary must be a mutable dictionary. A "raw" dictionary is mutable, a "compacted" dictionary is not mutable.

`load-dictionary` `[spell]` [Generic Function]

source `&key` into

Turn the word information in *source* into *into* or a new dictionary and return that dictionary.

source can be a stream, a string or pathname or some other sequence. If *source* is a stream, it must be an open character input stream. The word information is read from the stream. If *source* is a string a pathname it is a designator for a file that should be opened and read. If *source* is some other sequence, the sequence elements must be of the types discussed above. The word information from all source is combined into one dictionary in that case.

If supplied, *into* must be a mutable dictionary into which the word information should be loaded. The function returns *into* in that case. Otherwise the function creates a new dictionary, loads the word information into it and returns it.

Concept index

(Index is nonexistent)

Function and macro and variable and type index

A

adjective [spell] 3
adverb [spell] 4
article [spell] 5

B

base [spell] 3

C

case [spell] 8
conjunction [spell] 4
contraction [spell] 13

D

degree [spell] 4
demonstrative-adjective [spell] 6
demonstrative-pronoun [spell] 10
determinate [spell] 5
determiner [spell] 5

E

english-check-paragraph [spell] 2
english-lookup [spell] 2
entry-count [spell] 14

G

gender [spell] 7

I

infinitive [spell] 12
insert [spell] 15
interjection [spell] 7
interrogative-adjective [spell] 6

L

load-dictionary [spell] 15
lookup [spell] 14

M

map-entries [spell] 14

N

negative [spell] 10
noun [spell] 8
noun-verb-contraction [spell] 8
number [spell] 5

P

person [spell] 6
personal-pronoun [spell] 11
possessive-adjective [spell] 6
possessive-pronoun [spell] 11
preposition [spell] 9
pronoun [spell] 10
proper-noun [spell] 9

Q

quantifier [spell] 7

R

reflexive-pronoun [spell] 12
refnumber [spell] 6

S

strength [spell] 12
subordinate [spell] 4

T

tense [spell] 13

V

verb [spell] 12
verb-verb-contraction [spell] 13

W

word [spell] 3

Changelog

Release 0.3 (not yet released)

- Documentation is now available in the `documentation` directory.
- The new function `map-entries` calls a supplied function for each entry in a given dictionary.

Release 0.2 (2025-01-05)

- The `README.org` file now uses the org-mode format. The examples have been updated.
- The `spell/simple` system is now an alias for the `spell` system since the improved memory footprint makes the former unnecessary.
- Compilation and loading times as well as memory footprint have been significantly reduced.
- Dictionaries can now be built from multiple source files. Additional words are now loaded from `data/english-additions.txt`.

Release 0.1 (2024-12-20)

- Initial release with basic lookup for the English dictionary but slow compilation and loading and a big memory footprint.