CL-L10N: CL Localization Package

Copyright © 2004 Sean Ross. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The names of the authors and contributors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHORS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Table of Contents

1 Introduction 1

1.1 Supported Implementations 1

2 Getting Started 2

2.1 Downloading 2
2.2 Installing 2

3 API 3

3.1 Variables 3
3.2 Functions 3
3.3 Classes 6
3.4 Conditions 6

4 I18N 7

4.1 Internationalisation 7
4.2 API 8

5 Notes 9

5.1 Locale Designators 9
5.2 The Default Locale 9
5.3 Time Format Control Characters 9
5.4 Accessors to Locale Values. 10
5.5 Known Issues 11

6 Credits 12

7 Index 13

7.1 Function Index 13
7.2 Variable Index 13

1 Introduction

CL-L10N is a portable localization package for Common Lisp which is more or less modelled on the Allegro Common Lisp locale package. It currently supports various accessors (like locale-mon), number printing, money printing and time/date printing.

The CL-L10N Home Page is at `http://www.common-lisp.net/project/cl-l10n` where one can find details about mailing lists, cvs repositories and various releases.

Enjoy Sean.

1.1 Supported Implementations

- SBCL
- CMUCL
- CLISP
- Lispworks
- ECL
- Allegro CL

2 Getting Started

CL-L10N uses asdf as it's system definition tool and is required whenever you load the package. You will need to download it, or if you have sbcl `(require 'asdf)`

2.1 Downloading

- ASDF-INSTALL CL-L10N is available through asdf-install. If you are new to Common Lisp this is the suggested download method. With asdf-install loaded run `(asdf-install:install :cl-l10n)` This will download and install the package for you. Asdf-install will try to verify that the package signature is correct and that you trust the author. If the key is not found or the trust level is not sufficient a continuable error will be signalled. You can choose to ignore the error and continue to install the package. See the documentation of asdf-install for more details.
- DOWNLOAD

  The latest cl-l10n release will always be available from cl.net. Download and untar in an appropriate directory then symlink '`cl-l10n.asd`' to a directory on `asdf:*central-registry*` (see the documentation for asdf for details about setting up asdf).
- CVS

  If you feel the need to be on the bleeding edge you can use anonymous CVS access, see the Home Page for more details for accessing the archive. Once downloaded follow the symlink instructions above.

2.2 Installing

Once downloaded and symlinked you can load CL-L10N at anytime using `(asdf:oos 'asdf:load-op :cl-l10n)` This will compile CL-L10N the first time it is loaded.

Once installed run `(asdf:oos 'asdf:test-op :cl-l10n)` to test the package. If any tests fail please send an email to one of the mailing lists.

3 API

3.1 Variables

`*locale*` [Variable]

The default locale which will be used.

`*locale-path*` [Variable]

The default pathname where locale definition files can be found.

`*locales*` [Variable]

A hash table containing loaded locales keyed on locale name.

`*float-digits*` [Variable]

An integer value which determines the number of digits after the decimal point when all said digits are zero. This variable only has an effect when printing numbers as monetary printing gets this value from the locale.

3.2 Functions

`locale-name` *locale* [Function]

Returns the name of *locale*.

`locale` *name* **&key** (*use-cache* **t**) (*errorp* **t**) [Function]

Loads the locale designated by the locale-designator *name* which is expected to be found in `*locale-path*`. If use-cache is `nil` the locale will be forcibly reloaded from path otherwise the cached locale will be returned. If the locale cannot be found and errorp is not nil an error of type `locale-error` will be signalled.

`locale-value` *locale category-name key* [Function]

Returns the value of *key* in cagetory *category-name* found in the `locale` *locale*.

`load-all-locales` **&optional** (*path* ***locale-path***) [Function]

Load all locales found in pathname *path*.

`print-number` *number* **&key** (*stream* ***standard-output***) *no-ts no-dp locale *locale** [Function]

Prints *number* using locale *locale*. If *no-ts* is not nil no thousand seperators will be used when printing *number*. If *no-dp* is not nil the decimal seperator will be suppressed if *number* is not an integer.

`format-number` *stream arg no-dp no-ts* **&optional** (*locale* ***locale***) [Function]

format-number is intended to be used as an argument to the ~/ / format directive. Example (assuming *locale* is en_ZA)

```
(format t "~:/cl-l10n:format-number/" 1002932)
 prints ‘1,002,932‘
```

`print-money` *value* **&key** (*stream* ***standard-output***) *use-int-sym no-ts* (*locale* ***locale***) [Function]

Prints *value* as a monetary value using locale *locale*. If *no-ts* is not nil no thousand seperators will be used when printing *number*. If *use-int-sym* is not nil `locale-int-curr-symbol` will be used instead of the default `locale-currency-symbol`

`format-money` *stream arg use-int-sym no-ts* **&optional** (*locale* ***locale***) [Function]

Prints *value* as a monetary value using locale *locale*. format-money is intended to be used as the function to the ~/ / format directive Examples.

```
(format t "~/cl-l10n:format-money/" 188232.2322)
 prints ‘R188,232.23‘

;; and

(format t "~:/cl-l10n:format-money/" 188232.2322)
 prints ‘ZAR 188,232.23‘
```

`print-time` *ut* **&key** *show-date show-time* (*stream* ***standard-output***) (*locale* ***locale***) *fmt time-zone* [Function]

Prints the `universal-time` *ut* as a locale specific time to *stream*. Equivalent to `(format-time stream ut show-date show-time locale fmt time-zone)`.

`format-time` *stream ut show-date show-time* **&optional** (*locale* ***locale***) *fmt time-zone* [function]

Prints the `universal-time` *ut* as a locale specific time to *stream*. The format of the time printed is controlled by *show-time* and *show-date*.

`show-time and show-date are not nil`
: `locale-d-t-fmt`

`show-time and show-date are nil`
: `locale-t-fmt-ampm` or `locale-t-fmt` if `locale-t-fmt-ampm` has no apparent value.

`show-time is not nil and show-date is nil`
: locale-t-fmt

`show-date is not nil and show-time is nil`
: locale-d-fmt

If *fmt* is not nil then *show-date* and *show-time* are ignored and *fmt* is used as the format control string. See the Notes Section for the defined control characters which can be used.

Examples (assuming *locale* is “en_ZA” and a CL -2 Time Zone)

```
(format t "~:/cl-l10n:format-time/" 3192624000)
  prints ‘03/03/01’

(format t "~@/cl-l10n:format-time/" 3192624000)
```

```
  prints '18:00:00'

(format t "~:@/cl-l10n:format-time/" 3192624000)
  prints 'Sat 03 Mar 2001 18:00:00 +0200'

(format t "~v,v/cl-l10n:format-time/" "fr_FR" "%A" 3192624000)
  prints 'samedi'

(format t "~,v/cl-l10n:format-time/" "%A" 3192624000)
  prints 'Saturday'

; The Time Zone can be overriden with an extra v argument
(format t "~v,v,v/cl-l10n:format-time/" "en_ZA" "%A" -8 3192624000)
  print 'Sunday'
```

`format` *stream fmt-string* **&rest** *args* [Function]

Format is an unexported symbol in the cl-l10n package. It's use is to make formatting of dates, times, numbers and monetary values simpler. Shadow importing `cl-l10::format` into your package gives you a few new format directives. The new directives are ~U : Time and Date (universal-time), ~N : Numbers and ~M : Monetary values. All other format directives are unchanged and work as normal. These new directives are drop in replacements for the ~/cl-l10n:format-?/ calls.

```
;; These examples assume an en_ZA locale and a CL -2 Time Zone
(in-package :cl-user)

(shadowing-import 'cl-l10n::format)

(format t "~:U" 3192624000)
  prints '03/03/2001'

(format t "~,vU" "%A" 3192624000)
  prints 'Saturday'

(format t "~:N" 3192624000)
  prints '3,192,624,000'

(format t "~:M" 3192624000)
  prints 'ZAR 3,192,624,000.00'
```

`formatter` *fmt-string* [Macro]

Formatter is another unexported symbol in the cl-l10n package Shadow importing formatter gives support for the new format control directives.

`parse-number` *num-string* **&optional** (*locale* ***locale***) [Function]

Parses the string *num-string* into a number using *locale*.

`parse-time` *time-string* **&key** (*start* **0**) (*end* **(length time-string)**) [Function]
(*error-on-mismatch* **nil**) (*patterns* ***default-date-time-patterns***)
(*default-seconds* **nil**) (*default-minutes* **nil**) (*default-hours* **nil**) (*default-day* **nil**)
(*default-month* **nil**) (*default-year* **nil**) (*default-zone* **nil**) (*default-weekday* **nil**)
(*locale* ***locale***)

Tries very hard to make sense out of the argument time-string using locale and returns a single integer representing the universal time if successful. If not, it returns nil. If the :error-on-mismatch keyword is true, parse-time will signal an error instead of returning nil. Default values for each part of the time/date can be specified by the appropriate :default- keyword. These keywords can be given a numeric value or the keyword :current to set them to the current value. The default-default values are 00:00:00 on the current date, current time-zone.

Example, what date does the string "02/03/05" specify? parse-time will use the current locale or the locale-designator passed to it to determine the correct format for dates. In America (en_US) this date is the 3rd of February 2005, with an South African English (en_ZA) locale this date is the 2nd of March 2005 and with a Swedish locale (sv_SE) it's the 5th of March 2002.

Note. This is not my work but was done by Jim Healy and is a part of the CMUCL project, which has been modified to handle differt locales.

3.3 Classes

`locale` [Class]

Class Precedence: `standard-object` The class representing a loaded locale.

`category` [Class]

Class Precedence: `standard-object` The class representing a loaded category within a locale.

3.4 Conditions

`locale-error` [Condition]

Class Precedence: `error`

Root CL-L10N condition which will be signalled when an exceptional situation occurs.

`parser-error` [Condition]

Class Precedence: `error` Error which is signalled when an error occurs when parsing numbers or time strings.

4 I18N

4.1 Internationalisation

CL-L10N supports internationalised strings through the use of bundles. The process is currently extremely basic, and is bound to change in the future, but is flexible and does what is expected of it.

First you define a bundle using `make-instance`.

```
(defvar *my-bundle* (make-instance 'bundle))
```

Then you add resources to your bundle using either `add-resource` or `add-resources`.

```
(add-resources (bundle "af_")
  "showtime" "Dankie, die tyd is ~:@/cl-l10n:format-time/~%")

;; an empty string as the locale matcher becomes the default
(add-resources (bundle "")
  "showtime" "Thanks, the time is ~:@/cl-l10n:format-time/~%")
```

Then by using `gettext` you can lookup locale specific strings.

```
(defun timey () (format t (gettext "showtime" bundle) 3310880446))
(timey) ;; with locale en_ZA
 prints `Thanks, the time is Wed 01 Dec 2004 11:00:46 +0200`

(let ((*locale* (locale "af_ZA")))
  (timey))
 prints `Dankie, di tyd is Wo 01 Des 2004 11:00:46 +0200`
```

A useful trick is to define either a macro or reader macro wrapping gettext for your specific bundle eg.

```
(set-dispatch-macro-character
 #\# #\"
 #'(lambda (s c1 c2)
     (declare (ignore c2))
     (unread-char c1 s)
     `(cl-l10n:gettext ,(read s) bundle)))

;; or this

(defmacro _ (text)
  `(cl-l10n:gettext ,text bundle))
```

which would change the `timey` function to

```
(defun timey () (format t #"showtime" 3310880446))
;; or
(defun timey () (format t (_ "showtime") 3310880446))
```

4.2 API

`add-resource` *bundle from to locale-name* [Generic]

Adds an entry to *bundle* for *locale-name* mapping *from* to *to*. The *locale-name* does not have to be a full name like "en_US" but can be a partial match like "en_". Adding mappings for these two locale-names will result in the mapping for "en_US" being used when the locale is "en_US" and the mapping for "en_" being used when using any other english locale. Adding a mapping for an empty locale-name will become the default.

```
;; Add mapping for welcome for Afrikaans languages.
(add-resource *my-bundle* "welcome" "welkom" "af_")
```

`add-resources` (*bundle locale-name*) **&rest** *entries* [Macro]

Utility macro to group large amounts of entries into a single logical block for a locale.

```
(add-resources (bundle "af_")
  "hello" "hallo"
  "goodbye" "totsiens"
  "yes" "ja"
  "no "nee")

==

(add-resource bundle "hello" "hallo" "af_")
(add-resource bundle "goodbye" "totsiens" "af_")
(add-resource bundle "yes" "ja" "af_")
(add-resource bundle "no" "nee" "af_")
```

`gettext` *name bundle* **&optional** (**locale** ***locale***) [Function]

Looks for a mapping for *name* in *bundle*. If no mapping is found returns name.

5 Notes

5.1 Locale Designators

The locale argument to the various locale accessors and to the print functions is a locale designator. A locale designator is one of three things

- A locale object returned by `(locale name)`
- A string designating a locale, "en_ZA".
- A symbol eg. `:|en_ZA|`

5.2 The Default Locale

The default locale is found by looking at various environment variables. If the CL_LOCALE environment variable is set then this locale is loaded. Failing that the locale designated by the environment variable LC_CTYPE is loaded. If these two have failed then the POSIX locale is loaded as the default.

5.3 Time Format Control Characters

The following is a list of each legal control character in a time format string followed by a description of what is does.

- %% A percentage sign.
- %a locale's abbreviated weekday name (Sun..Sat)
- %A locale's full weekday name, variable length (Sunday..Saturday)
- %b locale's abbreviated month name (Jan..Dec)
- %B locale's full month name, variable length (January..December)
- %c locale's date and time (Sat Nov 04 12:02:33 EST 1989)
- %C century [00-99]
- %d day of month (01..31)
- %D date (mm/dd/yy)
- %e day of month, blank padded (1..31)
- %F same as %Y-%m-%d
- %g the 2-digit year corresponding to the %V week number
- %G the 4-digit year corresponding to the %V week number
- %h same as %b
- %H hour (00..23)
- %I hour (01..12)
- %j day of year (001..366)
- %k hour (0..23)
- %l hour (1..12)
- %m month (01..12)
- %M minute (00..59)

- %n a newline
- %N nanoseconds (Always 000000000)
- %p locale's upper case AM or PM indicator (blank in many locales)
- %P locale's lower case am or pm indicator (blank in many locales)
- %r time, 12-hour (hh:mm:ss [AP]M)
- %R time, 24-hour (hh:mm)
- %s seconds since '00:00:00 1970-01-01 UTC'
- %S second (00..60)
- %t a horizontal tab
- %T time, 24-hour (hh:mm:ss)
- %u day of week (1..7); 1 represents Monday
- %U week number of year with Sunday as first day of week (00..53)
- %V week number of year with Monday as first day of week (01..53)
- %w day of week (0..6); 0 represents Sunday
- %W week number of year with Monday as first day of week (00..53)
- %x locale's date representation (locale-d-fmt)
- %X locale's time representation (locale-t-fmt)
- %y last two digits of year (00..99)
- %Y year (1900...)
- %z RFC-2822 style numeric timezone (-0500)
- %Z RFC-2822 style numeric timezone (-0500)

5.4 Accessors to Locale Values.

There are a number of accessor functions to the various locale attributes defined. The functions are named by replacing underscores with hypens and prepending locale- to the name. The following is each defined accessor function in the format Category, Keyword and the accessor function for it.

- LC_MONETARY int_curr_symbol `locale-int-curr-symbol`
- LC_MONETARY currency_symbol `locale-currency-symbol`
- LC_MONETARY mon_decimal_point `locale-mon-decimal-point`
- LC_MONETARY mon_thousands_sep `locale-mon-thousands-sep`
- LC_MONETARY mon_grouping `locale-mon-grouping`
- LC_MONETARY positive_sign `locale-positive-sign`
- LC_MONETARY negative_sign `locale-negative-sign`
- LC_MONETARY int_frac_digits `locale-int-frac-digits`
- LC_MONETARY frac_digits `locale-frac-digits`
- LC_MONETARY p_cs_precedes `locale-p-cs-precedes`
- LC_MONETARY p_sep_by_space `locale-p-sep-by-space`
- LC_MONETARY n_cs_precedes `locale-n-cs-precedes`

- LC_MONETARY n_sep_by_space `locale-n-sep-by-space`
- LC_MONETARY p_sign_posn `locale-p-sign-posn`
- LC_MONETARY n_sign_posn `locale-n-sign-posn`
- LC_NUMERIC decimal_point `locale-decimal-point`
- LC_NUMERIC thousands_sep `locale-thousands-sep`
- LC_NUMERIC grouping `locale-grouping`
- LC_TIME abday `locale-abday`
- LC_TIME day `locale-day`
- LC_TIME abmon `locale-abmon`
- LC_TIME mon `locale-mon`
- LC_TIME d_t_fmt `locale-d-t-fmt`
- LC_TIME d_fmt `locale-d-fmt`
- LC_TIME t_fmt `locale-t-fmt`
- LC_TIME am_pm `locale-am-pm`
- LC_TIME t_fmt_ampm `locale-t-fmt-ampm`
- LC_TIME date_fmt `locale-date-fmt`
- LC_MESSAGES yesexpr `locale-yesexpr`
- LC_MESSAGES noexpr `locale-noexpr`
- LC_PAPER height `locale-height`
- LC_PAPER width `locale-width`
- LC_NAME name_fmt `locale-name-fmt`
- LC_NAME name_gen `locale-name-gen`
- LC_NAME name_mr `locale-name-mr`
- LC_NAME name_mrs `locale-name-mrs`
- LC_NAME name_miss `locale-name-miss`
- LC_NAME name_ms `locale-name-ms`
- LC_ADDRESS postal_fmt `locale-postal-fmt`
- LC_TELEPHONE tel_int_fmt `locale-tel-int-fmt`
- LC_MEASUREMENT measurement `locale-measurement`

5.5 Known Issues

- LC_COLLATE and LC_CTYPE categories in the locale files are currently ignored.
- Not all time format directives are supported (U, V and W are not implemented).

6 Credits

Thanks To

- Common-Lisp.net: For project hosting.

7 Index

7.1 Function Index

A

add-resource 8
add-resources 8

F

format 5
format-money 4
format-number 3
format-time 4
formatter 5

G

gettext 8

L

load-all-locales 3
locale 3
locale-name 3
locale-value 3

P

parse-number 5
parse-time 6
print-money 4
print-number 3
print-time 4

7.2 Variable Index

float-digits 3
locale 3
locale-path 3
locales 3