rfc822 egg

RFC822 message parsing
Extension for Chicken Scheme
Version 1.5

Shiro Kawai, ported to Chicken by Reed Sheridan

Table of Contents

1 About this egg

1.1 Version history

`1.5` Parses strings with #/ correctly

`1.4` rfc822-field->tokens fix

`1.3` rfc822-header->list now defined

`1.2` Fix bug in rfc822-field->tokens

`1.1` Hide internal procedure

`1.0` Initial release

1.2 Usage

Load this egg like so:

```
(require-extension rfc822)
```

2 Documentation

This is a complete port of Gauche's rfc.822 module, with the exception of the procedure `rfc822-date->date`.

2.1 Parsing a message header

`rfc822-header->list` [procedure]

`(rfc822-header->list iport &keyword strict? reader)`

Reads RFC822 format message from an input port iport, until it reaches the end of the message header. The header fields are unfolded, and broken into a list of the following format:

((name body) ...)

Name ... are the field names, and body ... are the corresponding field body, both as strings. Field names are converted to lower-case characters. Field bodies are not modified, except the folded line is concatenated, CRLFs removed. The order of fields are preserved.

By default, the parser works permissively. If EOF is encountered during parsing header, it is taken as the end of the message. And if a line that doesn't consist neither continuing (folded) line nor start a new header field, it is simply ignored. You can change this behavior by giving true value to the keyword argument strict?; then the parser raises an error for such a malformed header.

The keyword argument reader takes a procedure that reads a line from iport. Its default is read-line, which should be enough for most cases.

`rfc822-header-ref` [procedure]

`(rfc822-header-ref header-list field-name &optional default)`

An utility procedure to get a specific field from the parsed header list, which is returned by rfc822-header->list.

Field-name specifies the field name in a lowercase string. If the field with given name is in header-list, the procedure returns its value in a string. Otherwise, if default is given, it is returned, and if not, #f is returned.

2.2 Basic field parsers

Several procedures are provided to parse "structured" header fields of RFC2822 messages. These procedures deal with the body of a header field, i.e. if the header field is "To: Wandering Schemer <schemer@example.com>", they parse "Wandering Schemer <schemer@example.com>".

Most of procedures take an input port. Usually you first parse the entire header fields by rfc822-header->list, obtain the body of the header by rfc822-header-ref, then open an input string port for the body and use those procedures to parse them.

The reason for this complexity is because you need different tokenization schemes depending on the type of the field. Rfc2822 also allows comments to appear between tokens for most cases, so a simple-minded regexp won't do the job, since rfc2822 comment can be

nested and can't be represented by regular grammar. So, this layer of procedures are designed flexible enough to handle various syntaxes. For the standard header types, high-level parsers are also provided; see "specific field parsers" below.

`rfc822-next-token` [procedure]

```
(rfc822-next-token iport &optional tokenizer-specs)
```

A basic tokenizer. First it skips whitespaces and/or comments (CFWS) from iport, if any. Then reads one token according to tokenizer-specs. If iport reaches EOF before any token is read, EOF is returned.

Tokenizer-specs is a list of tokenizer spec, which is either a char-set or a cons of a char-set and a procedure.

After skipping CFWS, the procedure peeks a character at the head of iport, and checks it against the char-sets in tokenizer-specs one by one. If a char-set that contains the character belongs to is found, then a token is retrieved as follows: If the tokenizer spec is just a char-set, a sequence of characters that belong to the char-set consists a token. If it is a cons, the procedure is called with iport to read a token.

If the head character doesn't match any char-sets, the character is taken from iport and returned.

The default tokenizer-specs is as follows:

```
(list (cons (char-set #") rfc822-quoted-string)
      (cons *rfc822-atext-chars* rfc822-dot-atom))
```

Where rfc822-quoted-string and rfc822-dot-atom are tokenizer procedures described below, and *rfc822-atext-chars* is bound to a char-set of atext specified in rfc2822. This means rfc822-next-token retrieves a token either quoted-string or dot-atom specified in rfc2822 by default.

Using tokenizer-specs, you can customize how the header field is parsed. For example, if you want to retrieve a token that is either (1) a word constructed by uppercase characters, or (2) a quoted string, then you can call rfc822-next-token by this:

```
(rfc822-next-token iport
   `(,char-set:uppercase (,(char-set #") . ,rfc822-quoted-string)))
```

`rfc822-field->tokens` [procedure]

```
(rfc822-field->tokens field &optional tokenizer-specs)
```

A convenience procedure. Creates an input string port for a field body field, and calls rfc822-next-token repeatedly on it until it consumes all input, then returns a list of tokens. Tokenizer-specs is passed to rfc822-next-token.

`rfc822-skip-cfws` [procedure]

```
(rfc822-skip-cfws iport)
```

A utility procedure that consumes any comments and/or whitespace characters from iport, and returns the head character that is neither whitespace nor a comment. The returned character remains in iport.

`*rfc822-atext-chars*` [constant]

Bound to a char-set that is a valid constituent of atom.

`*rfc822-standard-tokenizers*` [constant]
Bound to the default tokenizer-specs.

`rfc822-atom` [procedure]
`(rfc822-atom iport)`

`rfc822-dot-atom` [procedure]
`(rfc822-dot-atom iport)`

`rfc822-quoted-string` [procedure]
`(rfc822-quoted-string iport)`

Tokenizers for atom, dot-atom and quoted-string, respectively. The double-quotes and escaping backslashes within quoted-string are removed by rfc822-quoted-string.

2.3 Specific field parsers

`rfc822-parse-date` [procedure]
`(rfc822-parse-date string)`

Takes RFC-822 type date string, and returns eight values:

year, month, day-of-month, hour, minutes, seconds, timezone, day-of-week.

Timezone is an offset from UT in minutes. Day-of-week is a day from sunday, and may be #f if that information is not available. Month is an integer between 1 and 12, inclusive. If the string is not parsable, all the elements are #f.

3 License

Index

*

R