NAME

Mail::Message::Field::Full − construct one smart line in a message header

INHERITANCE

```
Mail::Message::Field::Full
  is a Mail::Message::Field
  is a Mail::Reporter

Mail::Message::Field::Full is extended by
  Mail::Message::Field::Structured
  Mail::Message::Field::Unstructured
```

SYNOPSIS

```
# Getting to understand the complexity of a header field ...

my $fast = $msg->head->get('subject');
my $full = Mail::Message::Field::Full->from($fast);

my $full = $msg->head->get('subject')->study;  # same
my $full = $msg->head->study('subject');       # same
my $full = $msg->study('subject');             # same

# ... or build a complex header field yourself

my $f = Mail::Message::Field::Full->new('To');
my $f = Mail::Message::Field::Full->new('Subject: hi!');
my $f = Mail::Message::Field::Full->new(Subject => 'hi!');
```

DESCRIPTION

This is the *full* implementation of a header field: it has *full* understanding of all predefined header fields. These objects will be quite slow, because header fields can be very complex. Of course, this class delivers the optimal result, but for a quite large penalty in performance and memory consumption. Are you willing to accept?

This class supports the common header description from RFC2822 (formerly RFC822), the extensions with respect to character set encodings as specified in RFC2047, and the extensions on language specification and long parameter wrapping from RFC2231. If you do not need the latter two, then the Mail::Message::Field::Fast and Mail::Message::Field::Flex are enough for your application.

Extends ‘‘DESCRIPTION’’ in Mail::Message::Field.

OVERLOADED

Extends ‘‘OVERLOADED’’ in Mail::Message::Field.

overload: **“”**
: Inherited, see ‘‘OVERLOADED’’ in Mail::Message::Field

overload: **0+**
: Inherited, see ‘‘OVERLOADED’’ in Mail::Message::Field

overload: <=>
: Inherited, see ‘‘OVERLOADED’’ in Mail::Message::Field

overload: **bool**
: Inherited, see ‘‘OVERLOADED’’ in Mail::Message::Field

overload: **cmp**
: Inherited, see ‘‘OVERLOADED’’ in Mail::Message::Field

overload: **stringification**
: In string context, the decoded body is returned, as if **decodedBody()** would have been called.

METHODS

Extends ‘‘METHODS’’ in Mail::Message::Field.

Constructors

Extends ‘‘Constructors’’ in Mail::Message::Field.

`$obj`−>**clone**()

Inherited, see ‘‘Constructors’’ in Mail::Message::Field

Mail::Message::Field::Full−>**from**($field, `%options`)

Convert any `$field` (a Mail::Message::Field object) into a new Mail::Message::Field::Full object. This conversion is done the hard way: the string which is produced by the original object is parsed again. Usually, the string which is parsed is exactly the line (or lines) as found in the original input source, which is a good thing because Full fields are much more careful with the actual content.

`%options` are passed to the constructor (see **new()**). In any case, some extensions of this Full field class is returned. It depends on which field is created what kind of class we get.

example:

```
my $fast = $msg->head->get('subject');
my $full = Mail::Message::Field::Full->from($fast);

my $full = $msg->head->get('subject')->study;  # same
my $full = $msg->head->study('subject');       # same
my $full = $msg->get('subject');               # same
```

Mail::Message::Field::Full−>**new**($data)

Creating a new field object the correct way is a lot of work, because there is so much freedom in the RFCs, but at the same time so many restrictions. Most fields are implemented, but if you have your own field (and do no want to contribute it to MailBox), then simply call new on your own package.

You have the choice to instantiate the object as string or in prepared parts:

- **new** LINE, OPTIONS

  Pass a LINE as it could be found in a file: a (possibly folded) line which is terminated by a new-line.

- **new** NAME, [BODY], OPTIONS

  A set of values which shape the line.

The NAME is a wellformed header name (you may use **wellformedName()**) to be sure about the casing. The BODY is a string, one object, or an ref-array of objects. In case of objects, they must fit to the constructor of the field: the types which are accepted may differ. The optional ATTRIBUTE list contains Mail::Message::Field::Attribute objects. Finally, there are some OPTIONS.

```
−Option  --Defined in     --Default
 charset                    undef
 encoding                   'q'
 force                      false
 language                   undef
 log       Mail::Reporter   'WARNINGS'
 trace     Mail::Reporter   'WARNINGS'
```

charset => STRING

The body is specified in utf8, and must become 7−bits ascii to be transmited. Specify a charset to which the multi-byte utf8 is converted before it gets encoded. See **encode()**, which does the job.

encoding => ’q’|’Q’|’b’|’B’

Non-ascii characters are encoded using Quoted-Printable (’q’ or ’Q’) or Base64 (’b’ or ’B’) encoding.

force => BOOLEAN
Enforce encoding in the specified charset, even when it is not needed because the body does not contain any non-ascii characters.

language => STRING
The language used can be specified, however is rarely used my mail clients.

log => LEVEL
trace => LEVEL

example:

```
my $s = Mail::Message::Field::Full->new('Subject: Hello World');
my $s = Mail::Message::Field::Full->new('Subject', 'Hello World');

my @attrs   = (Mail::Message::Field::Attribute->new(...), ...);
my @options = (extra => 'the color blue');
my $t = Mail::Message::Field::Full->new(To => \@addrs, @attrs, @options);
```

The field

Extends ‘‘The field’’ in Mail::Message::Field.

`$obj`->**isStructured**()
Mail::Message::Field::Full->**isStructured**()
Inherited, see ‘‘The field’’ in Mail::Message::Field

`$obj`->**length**()
Inherited, see ‘‘The field’’ in Mail::Message::Field

`$obj`->**nrLines**()
Inherited, see ‘‘The field’’ in Mail::Message::Field

`$obj`->**print**([$fh])
Inherited, see ‘‘The field’’ in Mail::Message::Field

`$obj`->**size**()
Inherited, see ‘‘The field’’ in Mail::Message::Field

`$obj`->**string**([$wrap])
Inherited, see ‘‘The field’’ in Mail::Message::Field

`$obj`->**toDisclose**()
Inherited, see ‘‘The field’’ in Mail::Message::Field

Access to the name

Extends ‘‘Access to the name’’ in Mail::Message::Field.

`$obj`->**Name**()
Inherited, see ‘‘Access to the name’’ in Mail::Message::Field

`$obj`->**name**()
Inherited, see ‘‘Access to the name’’ in Mail::Message::Field

`$obj`->**wellformedName**([STRING])
Inherited, see ‘‘Access to the name’’ in Mail::Message::Field

Access to the body

Extends ‘‘Access to the body’’ in Mail::Message::Field.

`$obj`->**body**()
Inherited, see ‘‘Access to the body’’ in Mail::Message::Field

`$obj`->**decodedBody**(%options)
Returns the unfolded body of the field, where encodings are resolved. The returned line will still contain comments and such. The `%options` are passed to the decoder, see **decode()**.

BE WARNED: if the field is a structured field, the content may change syntax, because of encapsulated special characters. By default, the body is decoded as text, which results in a small difference within comments as well (read the RFC).

`$obj->`**folded**()
Inherited, see ''Access to the body'' in Mail::Message::Field

`$obj->`**foldedBody**([$body])
Inherited, see ''Access to the body'' in Mail::Message::Field

`$obj->`**stripCFWS**([STRING])
Mail::Message::Field::Full->**stripCFWS**([STRING])
Inherited, see ''Access to the body'' in Mail::Message::Field

`$obj->`**unfoldedBody**([$body, [$wrap]])
Inherited, see ''Access to the body'' in Mail::Message::Field

Access to the content
Extends ''Access to the content'' in Mail::Message::Field.

`$obj->`**addresses**()
Inherited, see ''Access to the content'' in Mail::Message::Field

`$obj->`**attribute**(`$name`, [$value])
Inherited, see ''Access to the content'' in Mail::Message::Field

`$obj->`**attributes**()
Inherited, see ''Access to the content'' in Mail::Message::Field

`$obj->`**beautify**()
For structured header fields, this removes the original encoding of the field's body (the format as it was offered to **parse()**), therefore the next request for the field will have to re-produce the read data clean and nice. For unstructured bodies, this method doesn't do a thing.

`$obj->`**comment**([STRING])
Inherited, see ''Access to the content'' in Mail::Message::Field

`$obj->`**createComment**(STRING, `%options`)
Mail::Message::Field::Full->**createComment**(STRING, `%options`)
Create a comment to become part in a field. Comments are automatically included within parenthesis. Matching pairs of parenthesis are permitted within the STRING. When a non-matching parenthesis are used, it is only permitted with an escape (a backslash) in front of them. These backslashes will be added automatically if needed (don't worry!). Backslashes will stay, except at the end, where it will be doubled.

The `%options` are `charset`, `language`, and `encoding` as always. The created comment is returned.

`$obj->`**createPhrase**(STRING, `%options`)
Mail::Message::Field::Full->**createPhrase**(STRING, `%options`)
A phrase is a text which plays a well defined role. This is the main difference with comments, which have do specified meaning. Some special characters in the phrase will cause it to be surrounded with double quotes: do not specify them yourself.

The `%options` are `charset`, `language`, and `encoding`, as always.

`$obj->`**study**()
Inherited, see ''Access to the content'' in Mail::Message::Field

`$obj->`**toDate**([$time])
Mail::Message::Field::Full->**toDate**([$time])
Inherited, see ''Access to the content'' in Mail::Message::Field

`$obj->`**toInt**()
Inherited, see ‘‘Access to the content’’ in Mail::Message::Field

Other methods
Extends ‘‘Other methods’’ in Mail::Message::Field.

`$obj->`**dateToTimestamp**(STRING)
Mail::Message::Field::Full->**dateToTimestamp**(STRING)
Inherited, see ‘‘Other methods’’ in Mail::Message::Field

Internals
Extends ‘‘Internals’’ in Mail::Message::Field.

`$obj->`**consume**(`$line` | <$name,<$body|$objects>>)
Inherited, see ‘‘Internals’’ in Mail::Message::Field

`$obj->`**decode**(STRING, `%options`)
Mail::Message::Field::Full->**decode**(STRING, `%options`)
Decode field encoded STRING to an utf8 string. The input STRING is part of a header field, and as such, may contain encoded words in `=?...?.?...?=` format defined by RFC2047. The STRING may contain multiple encoded parts, maybe using different character sets.

Be warned: you MUST first interpret the field into parts, like phrases and comments, and then decode each part separately, otherwise the decoded text may interfere with your markup characters.

Be warned: language information, which is defined in RFC2231, is ignored.

Encodings with unknown charsets are left untouched [requires v2.085, otherwise croaked]. Unknown characters within an charset are replaced by a ’?’.

```
-Option --Default
 is_text  1
```

is_text => BOOLEAN
Encoding on text is slightly more complicated than encoding structured data, because it contains blanks. Visible blanks have to be ignored between two encoded words in the text, but not when an encoded word follows or precedes an unencoded word. Phrases and comments are texts.

example:

```
print Mail::Message::Field::Full->decode('=?iso-8859-1?Q?J=F8rgen?=');
   # prints   JE<0slash>rgen
```

`$obj->`**defaultWrapLength**([$length])
Inherited, see ‘‘Internals’’ in Mail::Message::Field

`$obj->`**encode**(STRING, `%options`)
Encode the (possibly utf8 encoded) STRING to a string which is acceptable to the RFC2047 definition of a header: only containing us-ascii characters.

```
-Option  --Default
 charset   'us-ascii'
 encoding  'q'
 force     <flase>
 language  undef
 name      undef
```

charset => STRING
STRING is an utf8 string which has to be translated into any byte-wise character set for transport, because MIME-headers can only contain ascii characters.

encoding => ’q’|’Q’|’b’|’B’
The character encoding to be used. With `q` or `Q`, quoted-printable encoding will be used. With `b` or `B` , base64 encoding will be taken.

force => BOOLEAN
Encode the string, even when it only contains us-ascii characters. By default, this is off because it decreases readibility of the produced header fields.

language => STRING
RFC2231 defines how to specify language encodings in encoded words. The STRING is a strandard iso language name.

name => STRING
[3.002] When the name of the field is given, the first encoded line will be shorter.

`$obj`->**fold**(`$name`, `$body`, [$maxchars])
Mail::Message::Field::Full->**fold**(`$name`, `$body`, [$maxchars])
Inherited, see ''Internals'' in Mail::Message::Field

`$obj`->**setWrapLength**([$length])
Inherited, see ''Internals'' in Mail::Message::Field

`$obj`->**stringifyData**(STRING|ARRAY|$objects)
Inherited, see ''Internals'' in Mail::Message::Field

`$obj`->**unfold**(STRING)
Inherited, see ''Internals'' in Mail::Message::Field

Parsing

You probably do not want to call these parsing methods yourself: use the standard constructors (**new()**) and it will be done for you.

`$obj`->**consumeComment**(STRING)
Mail::Message::Field::Full->**consumeComment**(STRING)
Try to read a comment from the STRING. When successful, the comment without encapsulation parenthesis is returned, together with the rest of the string.

`$obj`->**consumeDotAtom**(STRING)
Returns three elemens: the atom-text, the rest string, and the concatenated comments. Both atom and comments can be undef.

`$obj`->**consumePhrase**(STRING)
Mail::Message::Field::Full->**consumePhrase**(STRING)
Take the STRING, and try to strip-off a valid phrase. In the obsolete phrase syntax, any sequence of words is accepted as phrase (as long as certain special characters are not used). RFC2822 is stricter: only one word or a quoted string is allowed. As always, the obsolete syntax is accepted, and the new syntax is produced.

This method returns two elements: the phrase (or undef) followed by the resulting string. The phrase will be removed from the optional quotes. Be warned that `""` will return an empty, valid phrase.

example:

```
my ($phrase, $rest) = $field->consumePhrase( q["hi!" <sales@example.com>] );
```

`$obj`->**parse**(STRING)
Get the detailed information from the STRING, and store the data found in the field object. The accepted input is very field type dependent. Unstructured fields do no parsing whatsoever.

`$obj`->**produceBody**()
Produce the text for the field, based on the information stored within the field object.

Usually, you wish the exact same line as was found in the input source of a message. But when you have created a field yourself, it should get formatted. You may call **beautify()** on a preformatted field to enforce a call to this method when the field is needed later.

Error handling

Extends ‘‘Error handling’’ in Mail::Message::Field.

`$obj`−>**AUTOLOAD**()
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj`−>**addReport**($object)
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj`−>**defaultTrace**([$level]|[$loglevel, `$tracelevel`]|[$level, `$callback`])
Mail::Message::Field::Full−>**defaultTrace**([$level]|[$loglevel, `$tracelevel`]|[$level, `$callback`])
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj`−>**errors**()
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj`−>**log**([$level, [$strings]])
Mail::Message::Field::Full−>**log**([$level, [$strings]])
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj`−>**logPriority**($level)
Mail::Message::Field::Full−>**logPriority**($level)
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj`−>**logSettings**()
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj`−>**notImplemented**()
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj`−>**report**([$level])
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj`−>**reportAll**([$level])
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj`−>**trace**([$level])
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj`−>**warnings**()
Inherited, see ‘‘Error handling’’ in Mail::Reporter

Cleanup

Extends ‘‘Cleanup’’ in Mail::Message::Field.

`$obj`−>**DESTROY**()
Inherited, see ‘‘Cleanup’’ in Mail::Reporter

DETAILS

Extends ‘‘DETAILS’’ in Mail::Message::Field.

DIAGNOSTICS

Warning: Field content is not numerical: `$content`
The numeric value of a field is requested (for instance the `Lines` or `Content−Length` fields should be numerical), however the data contains weird characters.

Warning: Illegal character in charset ’$charset’
The field is created with an utf8 string which only contains data from the specified character set. However, that character set can never be a valid name because it contains characters which are not permitted.

Warning: Illegal character in field name `$name`
A new field is being created which does contain characters not permitted by the RFCs. Using this field in messages may break other e−mail clients or transfer agents, and therefore mutulate or extinguish your message.

Warning: Illegal character in language '$lang'
The field is created with data which is specified to be in a certain language, however, the name of the language cannot be valid: it contains characters which are not permitted by the RFCs.

Warning: Illegal encoding '$encoding', used 'q'
The RFCs only permit base64 (`b` or `B`) or quoted-printable (`q` or `Q`) encoding. Other than these four options are illegal.

Error: Package `$package` does not implement `$method`.
Fatal error: the specific package (or one of its superclasses) does not implement this method where it should. This message means that some other related classes do implement this method however the class at hand does not. Probably you should investigate this and probably inform the author of the package.

SEE ALSO

This module is part of Mail-Message distribution version 3.012, built on February 11, 2022. Website: *http://perl.overmeer.net/CPAN/*

LICENSE