NAME

Mail::Message::Body::Encode − organize general message encodings

SYNOPSIS

```
my Mail::Message $msg = ...;
my $decoded = $msg->decoded;
my $encoded = $msg->encode(mime_type => 'image/gif',
    transfer_encoding => 'base64');

my $body = $msg->body;
my $decoded = $body->decoded;
my $encoded = $body->encode(transfer_encoding => '7bit');
```

DESCRIPTION

Manages the message's body encodings and decodings on request of the main program. This package adds functionality to the Mail::Message::Body class when the **decoded()** or **encode()** method is called.

Four types of encodings are handled (in the right order)

- eol encoding

  Various operating systems have different ideas about how to encode the line termination. UNIX uses a LF character, MacOS uses a CR, and Windows uses a CR/LF combination. Messages which are transported over Internet will always use the CRLF separator.

- transfer encoding

  Messages transmitted over Internet have to be plain ASCII. Complicated characters and binary files (like images and archives) must be encoded during transmission to an ASCII representation.

  The implementation of the required encoders and decoders is found in the Mail::Message::TransferEnc set of packages. The related manual page lists the transfer encodings which are supported.

- mime-type translation

  NOT IMPLEMENTED YET

- charset conversion

METHODS

Constructing a body

`$obj->`**check**()

Check the content of the body not to include illegal characters. Which characters are considered illegal depends on the encoding of this body.

A body is returned which is checked. This may be the body where this method is called upon, but also a new object, when serious changes had to be made. If the check could not be made, because the decoder is not defined, then `undef` is returned.

`$obj->`**encode**(%options)

Encode (translate) a Mail::Message::Body into a different format. See the DESCRIPTION above. Options which are not specified will not trigger conversions.

```
 -Option           --Default
  charset            PERL if text
  mime_type          undef
  result_type        <same as source>
  transfer_encoding  undef
```

charset => CODESET|'PERL'

If the CODESET is explicitly specified (for instance `iso-8859-10`, then the data is interpreted as raw bytes (blob), not as text. However, in case of `PERL`, it is considered to be an internal representation of characters (either latin1 or Perl's utf8 --not the same as utf-8--, you should not

know).

mime_type => STRING|FIELD
Convert into the specified mime type, which can be specified as STRING or FIELD. The FIELD is a Mail::Message::Field, and the STRING is converted in such object before use.

result_type => CLASS
The type of body to be created when the body is changed to fulfill the request on re-coding. Also the intermediate stages in the translation process (if needed) will use this type. CLASS must extend Mail::Message::Body.

transfer_encoding => STRING|FIELD

`$obj`->**encoded**()
Encode the body to a format what is acceptable to transmit or write to a folder file. This returns the body where this method was called upon when everything was already prepared, or a new encoded body otherwise. In either case, the body is checked.

`$obj`->**unify**($body)
Unify the type of the given `$body` objects with the type of the called body. `undef` is returned when unification is impossible. If the bodies have the same settings, the `$body` object is returned unchanged.

Examples:

```
my $bodytype = Mail::Message::Body::Lines;
my $html  = $bodytype->new(mime_type=>'text/html', data => []);
my $plain = $bodytype->new(mime_type=>'text/plain', ...);

my $unified = $html->unify($plain);
# $unified is the data of plain translated to html (if possible).
```

About the payload

`$obj`->**dispositionFilename**([$directory])
Various fields are searched for `filename` and `name` attributes. Without `$directory`, the name found will be returned unmodified.

When a `$directory` is given, a filename is composed. For security reasons, only the basename of the found name gets used and many potentially dangerous characters removed. If no name was found, or when the found name is already in use, then an unique name is generated.

Don't forget to read RFC6266 section 4.3 for the security aspects in your email application.

`$obj`->**isBinary**()
Returns true when the un-encoded message is binary data. This information is retrieved from knowledge provided by MIME::Types.

`$obj`->**isText**()
Returns true when the un-encoded message contains printable text.

Internals

`$obj`->**addTransferEncHandler**(`$name`, <$class|$object>)
Mail::Message::Body->**addTransferEncHandler**(`$name`, <$class|$object>)
Relate the NAMEd transfer encoding to an OBJECTs or object of the specified `$class`. In the latter case, an object of that `$class` will be created on the moment that one is needed to do encoding or decoding.

The `$class` or `$object` must extend Mail::Message::TransferEnc. It will replace existing class and object for this `$name`.

Why aren't you contributing this class to MailBox?

`$obj->`**getTransferEncHandler**($type)

Get the transfer encoder/decoder which is able to handle `$type`, or return undef if there is no such handler.

DIAGNOSTICS

Warning: Charset `$name` is not known

The encoding or decoding of a message body encounters a character set which is not understood by Perl's Encode module.

Warning: No decoder defined for transfer encoding `$name`.

The data (message body) is encoded in a way which is not currently understood, therefore no decoding (or recoding) can take place.

Warning: No encoder defined for transfer encoding `$name`.

The data (message body) has been decoded, but the required encoding is unknown. The decoded data is returned.

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