NAME

MIME::EncWords − deal with RFC 2047 encoded words (improved)

SYNOPSIS

MIME::EncWords is aimed to be another implimentation of MIME::Words so that it will achieve more exact conformance with RFC 2047 (formerly RFC 1522) specifications. Additionally, it contains some improvements. Following synopsis and descriptions are inherited from its inspirer, then added descriptions on improvements (**) or changes and clarifications (*).

Before reading further, you should see MIME::Tools to make sure that you understand where this module fits into the grand scheme of things. Go on, do it now. I'll wait.

Ready? Ok...

```
use MIME::EncWords qw(:all);

### Decode the string into another string, forgetting the charsets:
$decoded = decode_mimewords(
      'To: =?ISO-8859-1?Q?Keld_J=F8rn_Simonsen?= <keld AT dkuug DOT dk>',
      );

### Split string into array of decoded [DATA,CHARSET] pairs:
@decoded = decode_mimewords(
      'To: =?ISO-8859-1?Q?Keld_J=F8rn_Simonsen?= <keld AT dkuug DOT dk>',
      );

### Encode a single unsafe word:
$encoded = encode_mimeword("\xABFran\xE7ois\xBB");

### Encode a string, trying to find the unsafe words inside it:
$encoded = encode_mimewords("Me and \xABFran\xE7ois\xBB in town");
```

DESCRIPTION

Fellow Americans, you probably won't know what the hell this module is for. Europeans, Russians, et al, you probably do. `:-)`.

For example, here's a valid MIME header you might get:

```
From: =?US-ASCII?Q?Keith_Moore?= <moore AT cs DOT utk DOT edu>
To: =?ISO-8859-1?Q?Keld_J=F8rn_Simonsen?= <keld AT dkuug DOT dk>
CC: =?ISO-8859-1?Q?Andr=E9_?= Pirard <PIRARD AT vm1 DOT ulg DOT ac DOT be>
Subject: =?ISO-8859-1?B?SWYgeW91IGNhbiByZWFkIHRoaXMgeW8=?=
 =?ISO-8859-2?B?dSB1bmRlcnN0YW5kIHRoZSBleGFtcGxlLg==?=
 =?US-ASCII?Q?.._cool!?=
```

The fields basically decode to (sorry, I can only approximate the Latin characters with 7 bit sequences /o and 'e):

```
From: Keith Moore <moore AT cs DOT utk DOT edu>
To: Keld J/orn Simonsen <keld AT dkuug DOT dk>
CC: Andr'e  Pirard <PIRARD AT vm1 DOT ulg DOT ac DOT be>
Subject: If you can read this you understand the example... cool!
```

Supplement: Fellow Americans, Europeans, you probably won't know what the hell this module is for. East Asians, et al, you probably do. `(^_^)`.

For example, here's a valid MIME header you might get:

```
Subject: =?EUC-KR?B?sNTAuLinKGxhemluZXNzKSwgwvzB9ri7seIoaW1w?=
 =?EUC-KR?B?YXRpZW5jZSksILGzuLgoaHVicmlzKQ==?=
```

The fields basically decode to (sorry, I cannot approximate the non-Latin multibyte characters with any 7 bit sequences):

```
Subject: ???(laziness), ????(impatience), ??(hubris)
```

PUBLIC INTERFACE

decode_mimewords ENCODED, [OPTS...]

Function. Go through the string looking for RFC 2047−style ‘‘Q’’ (quoted-printable, sort of) or ‘‘B’’ (base64) encoding, and decode them.

In an array context, splits the ENCODED string into a list of decoded `[DATA, CHARSET]` pairs, and returns that list. Unencoded data are returned in a 1−element array `[DATA]`, giving an effective CHARSET of `undef`.

```
    $enc = '=?ISO−8859−1?Q?Keld_J=F8rn_Simonsen?= <keld AT dkuug DOT dk>';
    foreach (decode_mimewords($enc)) {
        print "", ($_[1] || 'US−ASCII'), ": ", $_[0], "\n";
    }
```

****** However, adjacent encoded-words with same charset will be concatenated to handle multibyte sequences safely.

****** Language information defined by RFC2231, section 5 will be additional third element, if any.

***** Whitespaces surrounding unencoded data will not be stripped so that compatibility with MIME::Words will be ensured.

In a scalar context, joins the ‘‘data’’ elements of the above list together, and returns that. *Warning: this is information-lossy,* and probably *not* what you want, but if you know that all charsets in the ENCODED string are identical, it might be useful to you. (Before you use this, please see ‘‘unmime’’ in MIME::WordDecoder, which is probably what you want.) ****** See also ‘‘Charset’’ option below.

In the event of a syntax error, $@ will be set to a description of the error, but parsing will continue as best as possible (so as to get *something* back when decoding headers). $@ will be false if no error was detected.

***** Malformed encoded-words will be kept encoded. In this case $@ will be set.

Any arguments past the ENCODED string are taken to define a hash of options. ****** When Unicode/multibyte support is disabled (see ‘‘USE_ENCODE’’ in MIME::Charset), these options will not have any effects.

Charset ******

Name of character set by which data elements in scalar context will be converted. The default is no conversion. If this option is specified as special value `"_UNICODE_"`, returned value will be Unicode string.

Note: This feature is still information-lossy, *except* when `"_UNICODE_"` is specified.

Detect7bit ******

Try to detect 7−bit charset on unencoded portions. Default is `"YES"`.

Mapping ******

In scalar context, specify mappings actually used for charset names. `"EXTENDED"` uses extended mappings. `"STANDARD"` uses standardized strict mappings. Default is `"EXTENDED"`.

encode_mimeword RAW, [ENCODING], [CHARSET]

Function. Encode a single RAW ‘‘word’’ that has unsafe characters. The ‘‘word’’ will be encoded in its entirety.

```
    ### Encode "<<Franc,ois>>":
    $encoded = encode_mimeword("\xABFran\xE7ois\xBB");
```

You may specify the ENCODING (`"Q"` or `"B"`), which defaults to `"Q"`. ****** You may also specify it as ‘‘special’’ value: `"S"` to choose shorter one of either `"Q"` or `"B"`.

You may specify the CHARSET, which defaults to `iso−8859−1`.

***** Spaces will be escaped with ‘‘_’’ by `"Q"` encoding.

encode_mimewords RAW, [OPTS]

Function. Given a RAW string, try to find and encode all ‘‘unsafe’’ sequences of characters:

```
### Encode a string with some unsafe "words":
$encoded = encode_mimewords("Me and \xABFran\xE7ois\xBB");
```

Returns the encoded string.

****** RAW may be a Unicode string when Unicode/multibyte support is enabled (see ‘‘USE_ENCODE’’ in MIME::Charset). Furthermore, RAW may be a reference to that returned by ‘‘decode_mimewords’’ on array context. In latter case ‘‘Charset’’ option (see below) will be overridden (see also a note below).

Note: ***** When RAW is an arrayref, adjacent encoded-words (i.e. elements having non-ASCII charset element) are concatenated. Then they are split taking care of character boundaries of multibyte sequences when Unicode/multibyte support is enabled. Portions for unencoded data should include surrounding whitespace(s), or they will be merged into adjoining encoded−word(s).

Any arguments past the RAW string are taken to define a hash of options:

Charset

Encode all unsafe stuff with this charset. Default is ’ISO−8859−1’, a.k.a. ‘‘Latin−1’’.

Detect7bit ******

When ‘‘Encoding’’ option (see below) is specified as `"a"` and ‘‘Charset’’ option is unknown, try to detect 7−bit charset on given RAW string. Default is `"YES"`. When Unicode/multibyte support is disabled, this option will not have any effects (see ‘‘USE_ENCODE’’ in MIME::Charset).

Encoding

The encoding to use, `"q"` or `"b"`. ****** You may also specify ‘‘special’’ values: `"a"` will automatically choose recommended encoding to use (with charset conversion if alternative charset is recommended: see MIME::Charset); `"s"` will choose shorter one of either `"q"` or `"b"`. **Note**: ***** As of release 1.005, The default was changed from `"q"` (the default on MIME::Words) to `"a"`.

Field

Name of the mail field this string will be used in. ****** Length of mail field name will be considered in the first line of encoded header.

Folding ******

A Sequence to fold encoded lines. The default is `"\n"`. If empty string `""` is specified, encoded-words exceeding line length (see ‘‘MaxLineLen’’ below) will be split by SPACE.

Note: ***** Though RFC 5322 (formerly RFC 2822) states that the lines in Internet messages are delimited by CRLF (`"\r\n"`), this module chose LF (`"\n"`) as a default to keep backward compatibility. When you use the default, you might need converting newlines before encoded headers are thrown into session.

Mapping ******

Specify mappings actually used for charset names. `"EXTENDED"` uses extended mappings. `"STANDARD"` uses standardized strict mappings. The default is `"EXTENDED"`. When Unicode/multibyte support is disabled, this option will not have any effects (see ‘‘USE_ENCODE’’ in MIME::Charset).

MaxLineLen ******

Maximum line length excluding newline. The default is 76. Negative value means unlimited line length (as of release 1.012.3).

Minimal ******

Takes care of natural word separators (i.e. whitespaces) in the text to be encoded. If `"NO"` is specified, this module will encode whole text (if encoding needed) not regarding whitespaces; encoded-words exceeding line length will be split based only on their lengths. Default is `"YES"` by which minimal portions of text are encoded. If `"DISPNAME"` is specified, portions including special characters described in RFC5322 (formerly RFC2822, RFC822) address specification (section 3.4) are also encoded. This is useful for encoding display-name of address fields.

Note: As of release 0.040, default has been changed to `"YES"` to ensure compatibility with MIME::Words. On earlier releases, this option was fixed to be `"NO"`.

Note: `"DISPNAME"` option was introduced at release 1.012.

Replacement ******
See ‘‘Error Handling’’ in MIME::Charset.

Configuration Files **

Built-in defaults of option parameters for ‘‘decode_mimewords’’ (except ’Charset’ option) and ‘‘encode_mimewords’’ can be overridden by configuration files: *MIME/Charset/Defaults.pm* and *MIME/EncWords/Defaults.pm*. For more details read *MIME/EncWords/Defaults.pm.sample*.

VERSION

Consult `$VERSION` variable.

Development versions of this module may be found at <http://hatuka.nezumi.nu/repos/MIME−EncWords/>.

SEE ALSO

MIME::Charset, MIME::Tools

AUTHORS

The original version of function *decode_mimewords()* is derived from MIME::Words module that was written by:
Eryq (*eryq AT zeegee DOT com*), ZeeGee Software Inc (*http://www.zeegee.com*).
David F. Skoll (dfs AT roaringpenguin DOT com) http://www.roaringpenguin.com

Other stuff are rewritten or added by:
Hatuka*nezumi − IKEDA Soji <hatuka(at)nezumi.nu>.