NAME

MIME::Words − deal with RFC 2047 encoded words

SYNOPSIS

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::Words 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!
```

PUBLIC INTERFACE

decode_mimewords ENCODED

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";
}
```

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.)

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.

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

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

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. 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”.

Encoding

The encoding to use, `"q"` or `"b"`. The default is `"q"`.

Warning: this is a quick-and-dirty solution, intended for character sets which overlap ASCII. **It does not comply with the RFC 2047 rules regarding the use of encoded words in message headers**. You may want to roll your own variant, using `encode_mimeword()`, for your application. *Thanks to Jan Kasprzak for reminding me about this problem.*

SEE ALSO

MIME::Base64, MIME::QuotedPrint, MIME::Tools

For other implementations of this or similar functionality (particularly, ones with proper UTF8 support), see:

Encode::MIME::Header, MIME::EncWords, MIME::AltWords

At some future point, one of these implementations will likely replace MIME::Words and MIME::Words will become deprecated.

NOTES

Exports its principle functions by default, in keeping with MIME::Base64 and MIME::QuotedPrint.

AUTHOR

Eryq (*eryq AT zeegee DOT com*), ZeeGee Software Inc (*http://www.zeegee.com*). Dianne Skoll (dfs AT roaringpenguin DOT com) http://www.roaringpenguin.com

Thanks also to...

```
Kent Boortz        For providing the idea, and the baseline
                   RFC−1522−decoding code!
KJJ at PrimeNet    For requesting that this be split into
                   its own module.
Stephane Barizien  For reporting a nasty bug.
```