NAME

Mail::Message::Construct::Forward − forwarding a Mail::Message

SYNOPSIS

```
my Mail::Message $forward = $message->forward(To => 'you');
$forward->send;
```

DESCRIPTION

Complex functionality on Mail::Message objects is implemented in different files which are autoloaded. This file implements the functionality related to creating forwarded messages.

METHODS

Constructing a message

$obj->**forward**(%options)

Forward the content of this message. The body of the message to be forwarded is encapsulated in some accompanying text (if you have no wish for that, than `bounce` is your choice). A Mail::Message object is returned on success.

You may forward a whole message, but also message parts. You may wish to overrule some of the default header settings for the reply immediately, or you may do that later with `set` on the header.

When a multi-part body is encountered, and the message is included to ATTACH, the parts which look like signatures will be removed. If only one message remains, it will be the added as single attachment, otherwise a nested multipart will be the result. The value of this option does not matter, as long as it is present. See `Mail::Message::Body::Multipart`.

```
-Option    --Default
 Bcc         undef
 Cc          undef
 Date        <now>
 From        <'to' in current>
 Message-ID  <uniquely generated>
 Subject     forwardSubject()
 To          <required>
 body        undef
 include     <if body then 'NO' else C<'INLINE'>>
 preamble    constructed from prelude and postlude
 signature   undef
```

Bcc => ADDRESSES

Receivers of blind carbon copies: their names will not be published to other message receivers.

Cc => ADDRESSES

The carbon-copy receivers, by default none.

Date => DATE

The date to be used in the message sent.

From => ADDRESSES

Your identification, by default taken from the `To` field of the source message.

Message-ID => STRING

Supply a STRING as specific message-id for the forwarded message. By default, one is generated for you. If there are no angles around your id, they will be added.

Subject => STRING|CODE

Force the subject line to the specific STRING, or the result of the subroutine specified by CODE. The subroutine will be called passing the subject of the original message as only argument. By default, the **forwardSubject()** method is used.

To => ADDRESSES
The destination of your message. Obligatory. The ADDRESSES may be specified as string, a Mail::Address object, or as array of Mail::Address objects.

body => OBJECT
If you specify a fully prepared body OBJECT, it will be used as forwarded message contents. In this case, only the headers are constructed for you.

include => 'NO'|'INLINE'|'ATTACH'|'ENCAPSULATE'
Must the message where this is a reply to be included in the message? When `INLINE` is given, you may pass the options of **forwardInline()** as well.

In many applications, the forward option `as attachment` results in a structure which is produced when this option is set to `ENCAPSULATE`. Their default behavior is usually `INLINE`.

It is only possible to inline textual messages, therefore binary or multi-part messages will always be enclosed as attachment. Read the details in section ''Creating a forward''..

preamble => STRING|BODY
Part which is attached before the forwarded message. If no preamble is given, then it is constructed from the prelude and postlude. When these are also not present, you will still get a one liner: the result of **forwardPrelude()**

signature => BODY|MESSAGE
The signature to be added in case of a multi-part forward. The mime-type of the signature body should indicate this is a used as such. However, in INLINE mode, the body will be taken, a line containing `'-- '` added before it, and added behind the epilogue.

`$obj`->**forwardAttach**(%options)
Forward the message as *flat* attachment to the specified `preamble`. You can specify all options available to `forward()`, although a `preamble` which is provided as body object is required, and any specified `body` is ignored.

```
-Option  --Default
 preamble  <required>
```

preamble => BODY|PART

`$obj`->**forwardEncapsulate**(%options)
Like **forwardAttach()**, but in this case the original message is first encapsulated as nested message in a Mail::Message::Body::Nested, and then joint into a multipart.

You can specify all options available to `forward()`, although a `preamble` which is provided as body object is required, and any specified `body` is ignored. Signatures are not stripped. Signatures are not stripped.

```
-Option  --Default
 preamble  <required>
```

preamble => BODY|PART

`$obj`->**forwardInline**(%options)
This method is equivalent in behavior to **forward()** with the option `include` set to `'INLINE'`. You can specify most of the fields which are available to **forward()** except `include` and `body`.

```
-Option         --Default
 is_attached      "[The forwarded message is attached]\n"
 max_signature    10
 postlude         undef
 prelude          undef
 quote            undef
 strip_signature  qr/^--\s/
```

is_attached => STRING
A forward on binary messages can not be inlined. Therefore, they are automatically translated into an attachment, as made by **forwardAttach()**. The obligatory preamble option to that method may be specified as option to this method, to be used in case of such a forward of a binary, but is otherwise constructed from the prelude, the value of this option, and the postlude.

max_signature => INTEGER
Passed to Mail::Message::Body::stripSignature(max_lines). Only effective for single-part messages.

postlude => BODY
The line(s) which to be added after the quoted reply lines. Create a body for it first. This should not include the signature, which has its own option. The signature will be added after the postlude when the forwarded message is `INLINE`d.

prelude => BODY
The line(s) which will be added before the quoted forwarded lines. If nothing is specified, the result of the **forwardPrelude()** method is used. When `undef` is specified, no prelude will be added.

quote => CODE|STRING
Mangle the lines of an `INLINE`d reply with CODE, or by prepending a STRING to each line. The routine specified by CODE is called when the line is in `$_`.

By default, nothing is added before each line. This option is processed after the body has been decoded.

strip_signature => REGEXP|STRING|CODE
Remove the signature of the sender. The value of this parameter is passed to Mail::Message::Body::stripSignature(pattern), unless the source text is not included. The signature is stripped from the message before quoting.

`$obj`->**forwardNo**(%options)
Construct a forward, where the whole body of the message is already constructed. That complex body is usually produced in **forwardInline()**, **forwardAttach()**, or **forwardEncapsulate()**.

The `%options` are the same as for `forward()` except that `body` is required. Some other options, like `preamble`, are ignored.

```
-Option--Default
 body    <required>
```

body => BODY

`$obj`->**forwardPostlude**()
Added after the forwarded message.

example:

```
---- END forwarded message
```

`$obj`->**forwardPrelude**()
Create a few lines to be included before the forwarded message content. The return is an array of lines.

example:

```
---- BEGIN forwarded message
From: him@somewhere.else.nl (Original Sender)
To: me@example.com (Me the receiver)
Cc: the.rest@world.net
Date: Wed, 9 Feb 2000 15:44:05 -0500
<blank line>
```

`$obj->`**forwardSubject**(STRING)

Create a subject for a message which is a forward from this one. This routine tries to count the level of reply in subject field, and transform it into a standard form. Please contribute improvements.

example:

```
subject                 --> Forw: subject
Re: subject             --> Forw: Re: subject
Re[X]: subject          --> Forw: Re[X]: subject
<blank>                 --> Forwarded
```

DETAILS

Creating a forward

The main difference between **bounce()** and **forward()** is the reason for message processing. The *bounce* has no intention to modify the content of message: the same information is passed-on to someplace else. This may mean some conversions, but for instance, the Message-ID does not need to be changed.

The purpose of ***forward()*** is to pass on information which is modified: annotated or reduced. The information is not sent back to the author of the original message (which is implemented by **reply()**), but to someone else.

So: some information comes in, is modified, and than forwarded to someone else. Currently, there are four ways to get the original information included, which are explained in the next sections.

After the creation of the forward, you may want to **rebuild()** the message to remove unnecessary complexities. Of course, that is not required.

forward, specify a body

When you specify forward(body), you have created your own body object to be used as content of the forwarded message. This implies that forward(include) is `'NO'`: no automatic generation of the forwarded body.

forward, inline the original

The forward(include) is set to `'INLINE'` (the default) This is the most complicated situation, but most often used by MUAs: the original message is inserted textually in the new body. You can set-up automatic stripping of signatures, the way of encapsulation, and texts which should be added before and after the encapsulated part.

However, the result may not always be what you expect. For instance, some people use very long signatures which will not be automatically stripped because the pass the threshold. So, you probably need some manual intervention after the message is created and before it is sent.

When a binary message is encountered, inlining is impossible. In that case, the message is treated as if `'ENCAPSULATE'` was requested.

forward, attach the original

When forward(include) is explicitly set to `'ATTACH'` the result will be a multipart which contains two parts. The first part will be your message, and the second the body of the original message.

This means that the headers of the forwarded message are used for the new message, and detached from the part which now contains the original body information. Content related headers will (of course) still be part of that part, but lines line `To` and `Subject` will not be stored with that part.

As example of the structural transformation:

```
# code: $original->printStructure;
multipart/alternative: The source message
  text/plain: content in raw text
  text/html: content as html

# code: $fwd = $original->forward(include => 'ATTACH');
```

```
# code: $fwd->printStructure
multipart/mixed: The source message
  text/plain: prelude/postlude/signature
  multipart/alternative
    text/plain: content in raw text
    text/html: content as html
```

forward, encapsulate the original

When forward(include) is explicitly set to `'ENCAPSULATE'`, then the original message is left in-tact as good as possible. The lines of the original message are used in the main message header but also enclosed in the part header.

The encapsulation is implemented using a nested message, content type `message/rfc822`. As example of the structural transformation:

```
# code: $original->printStructure;
multipart/alternative: The source message
  text/plain: content in raw text
  text/html: content as html

# code: $fwd = $original->forward(include => 'ENCAPSULATE');
# code: $fwd->printStructure
multipart/mixed: The source message
  text/plain: prelude/postlude/signature
  message/rfc822
     multipart/alternative: The source message
        text/plain: content in raw text
        text/html: content as html
```

The message structure is much more complex, but no information is lost. This is probably the reason why many MUAs use this when the forward an original message as attachment.

DIAGNOSTICS

Error: Cannot include forward source as `$include`.

Unknown alternative for the forward(include). Valid choices are `NO`, `INLINE`, `ATTACH`, and `ENCAPSULATE`.

Error: Method forwardAttach requires a preamble

Error: Method forwardEncapsulate requires a preamble

Error: No address to create forwarded to.

If a forward message is created, a destination address must be specified.

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