NAME

Mail::Message::Head::FieldGroup − a sub set of fields in a header

INHERITANCE

```
Mail::Message::Head::FieldGroup
  is a Mail::Reporter

Mail::Message::Head::FieldGroup is extended by
  Mail::Message::Head::ListGroup
  Mail::Message::Head::ResentGroup
  Mail::Message::Head::SpamGroup
```

SYNOPSIS

Never instantiated directly.

DESCRIPTION

Some fields have a combined meaning: a set of fields which represent one intermediate step during the transport of the message (a *resent group*, implemented in Mail::Message::Head::ResentGroup), fields added by mailing list software (implemented in Mail::Message::Head::ListGroup), or fields added by Spam detection related software (implemented by Mail::Message::Head::SpamGroup). Each set of fields can be extracted or added as group with objects which are based on the implementation in this class.

Extends ‘‘DESCRIPTION’’ in Mail::Reporter.

METHODS

Extends ‘‘METHODS’’ in Mail::Reporter.

Constructors

Extends ‘‘Constructors’’ in Mail::Reporter.

`$obj`−>**clone**()

Make a copy of this object. The collected fieldnames are copied and the list type information. No deep copy is made for the header: this is only copied as reference.

`$obj`−>**from**($head|$message)

Create a group of fields based on the specified `$message` or message `$head`. This may return one or more of the objects, which depends on the type of group. Mailing list fields are all stored in one object, where resent and spam groups can appear more than once.

`$obj`−>**implementedTypes**()

Mail::Message::Head::FieldGroup−>**implementedTypes**()

Returns a list of strings containing all possible return values for **type()**.

Mail::Message::Head::FieldGroup−>**new**($fields, `%options`)

Construct an object which maintains one set of header `$fields`. The`$fields` may be specified as `Mail::Message::Field` objects or as key-value pairs. The `%options` and `$fields` (as key-value pair) can be mixed: they are distinguished by their name, where the fields always start with a capital. The field objects must aways lead the `%options`.

```
 -Option  --Defined in     --Default
  head                       undef
  log       Mail::Reporter   'WARNINGS'
  software                   undef
  trace     Mail::Reporter   'WARNINGS'
  type                       undef
  version                    undef
```

head => HEAD

The header HEAD object is used to store the grouped fields in. If no header is specified, a Mail::Message::Head::Partial is created for you. If you wish to scan the existing fields in a header, then use the **from()** method.

log => LEVEL
software => STRING
Name of the software which produced the fields.

trace => LEVEL
type => STRING
Group name for the fields. Often the same, or close to the same STRING, as the `software` option contains.

version => STRING
Version number for the software which produced the fields.

The header

`$obj`->**add**(<$field, `$value`> | `$object`)
Add a field to the header, using the field group. When the field group is already attached to a real message header, it will appear in that one as well as being registered in this set. If no header is defined, the field only appears internally.

example: adding a field to a detached list group

```
my $this = Mail::Message::Head::ListGroup->new(...);
$this->add('List-Id' => 'mailbox');
$msg->addListGroup($this);
$msg->send;
```

example: adding a field to an attached list group

```
my $lg = Mail::Message::Head::ListGroup->from($msg);
$lg->add('List-Id' => 'mailbox');
```

`$obj`->**addFields**([$fieldnames])
Add some `$fieldnames` to the set.

`$obj`->**attach**($head)
Add a group of fields to a message `$head`. The fields will be cloned(!) into the header, so that the field group object can be used again.

example: attaching a list group to a message

```
my $lg = Mail::Message::Head::ListGroup->new(...);
$lg->attach($msg->head);
$msg->head->addListGroup($lg);   # same

$msg->head->addSpamGroup($sg);   # also implemented with attach
```

`$obj`->**delete**()
Remove all the header lines which are combined in this fields group, from the header.

`$obj`->**fieldNames**()
Return the names of the fields which are used in this group.

`$obj`->**fields**()
Return the fields which are defined for this group.

`$obj`->**head**()
Returns the header object, which includes these fields.

Access to the header

`$obj`->**software**()
Returns the name of the software as is defined in the headers. The may be slightly different from the return value of **type()**, but usually not too different.

`$obj->`**type**()
Returns an abstract name for the field group; which software is controlling it. `undef` is returned in case the type is not known. Valid names are group type dependent: see the applicable manual pages. A list of all types can be retrieved with **implementedTypes()**.

`$obj->`**version**()
Returns the version number of the software used to produce the fields. Some kinds of software do leave such a trace, other cases will return `undef`

Internals

`$obj->`**collectFields**([$name])
Scan the header for fields which are usually contained in field group with the specified `$name`. For mailinglist groups, you can not specify a `$name:` only one set of headers will be found (all headers are considered to be produced by exactly one package of mailinglist software).

This method is automatically called when a field group is constructed via **from()** on an existing header or message.

Returned are the names of the list header fields found, in scalar context the amount of fields. An empty list/zero indicates that there was no group to be found.

Please warn the author of MailBox if you see that to few or too many fields are included.

`$obj->`**detected**($type, `$software`, `$version`)
Sets the values for the field group type, software, and version, prossibly to `undef`.

Error handling

Extends ''Error handling'' in Mail::Reporter.

`$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::Head::FieldGroup->**defaultTrace**([$level]|[$loglevel, `$tracelevel`]|[$level, `$callback`])
Inherited, see ''Error handling'' in Mail::Reporter

`$obj->`**details**()
Produce information about the detected/created field group, which may be helpful during debugging. A nicely formatted string is returned.

`$obj->`**errors**()
Inherited, see ''Error handling'' in Mail::Reporter

`$obj->`**log**([$level, [$strings]])
Mail::Message::Head::FieldGroup->**log**([$level, [$strings]])
Inherited, see ''Error handling'' in Mail::Reporter

`$obj->`**logPriority**($level)
Mail::Message::Head::FieldGroup->**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

`$obj->`**print**([$fh])
Print the group to the specified `$fh` or GLOB. This is probably only useful for debugging purposed. The output defaults to the selected file handle.

`$obj`->**report**([$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::Reporter.

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

DIAGNOSTICS

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