NAME

Mail::Cap − understand mailcap files

SYNOPSIS

```
my $mc   = Mail::Cap->new;

my $desc = $mc->description('image/gif');
print "GIF desc: $desc\n";

my $cmd  = $mc->viewCmd('text/plain; charset=iso-8859-1', 'file.txt');
```

DESCRIPTION

Parse mailcap files as specified in "RFC 1524 −−A User Agent Configuration Mechanism For Multimedia Mail Format Information>. In the description below `$type` refers to the MIME type as specified in the `Content-Type` header of mail or HTTP messages. Examples of types are:

```
image/gif
text/html
text/plain; charset=iso-8859-1
```

You could also take a look at the File::MimeInfo distribution, which are accessing tables which are used by many applications on a system, and therefore have succeeded the mail-cap specifications on modern (UNIX) systems.

METHODS

Constructors

Mail::Cap−>**new**(%options)

Create and initialize a new Mail::Cap object. If you give it an argument it will try to parse the specified file. Without any arguments it will search for the mailcap file using the standard mailcap path, or the MAILCAPS environment variable if it is defined.

```
−Option  −−Default
 filename  undef
 take      'FIRST'
```

filename => FILENAME

Add the specified file to the list to standard locations. This file is tried first.

take => 'ALL'|'FIRST'

Include all mailcap files you can find. By default, only the first file is parsed, however the RFC tells us to include ALL. To maintain backwards compatibility, the default only takes the FIRST.

example:

```
$mcap = new Mail::Cap;
$mcap = new Mail::Cap "/mydir/mailcap";
$mcap = new Mail::Cap filename => "/mydir/mailcap";
$mcap = new Mail::Cap take => 'ALL';
$mcap = Mail::Cap->new(take => 'ALL');
```

Run commands

These methods invoke a suitable program presenting or manipulating the media object in the specified file. They all return `1` if a command was found, and `0` otherwise. You might test `$?` for the outcome of the command.

`$obj`−>**compose**($type, `$file`)
`$obj`−>**edit**($type, `$file`)
`$obj`−>**print**($type, `$file`)
`$obj`−>**view**($type, `$file`)

Command creator

These methods return a string that is suitable for feeding to **system()** in order to invoke a suitable program presenting or manipulating the media object in the specified file. It will return `undef` if no suitable specification exists.

`$obj->`**composeCmd**($type, `$file`)
`$obj->`**editCmd**($type, `$file`)
`$obj->`**printCmd**($type, `$file`)
`$obj->`**viewCmd**($type, `$file`)

Look-up definitions

Methods return the corresponding mailcap field for the type.

`$obj->`**description**($type)
`$obj->`**field**($type, `$field`)
Returns the specified field for the type. Returns undef if no specification exists.

`$obj->`**nametemplate**($type)
`$obj->`**textualnewlines**($type)
`$obj->`**x11_bitmap**($type)

SEE ALSO

This module is part of the MailTools distribution, *http://perl.overmeer.net/mailtools/*.

AUTHORS

The MailTools bundle was developed by Graham Barr. Later, Mark Overmeer took over maintenance without commitment to further development.

Mail::Cap by Gisle Aas <aas@oslonett.no>. Mail::Field::AddrList by Peter Orbaek <poe@cit.dk>. Mail::Mailer and Mail::Send by Tim Bunce <Tim.Bunce@ig.co.uk>. For other contributors see ChangeLog.

LICENSE