

Package ‘emayili’

December 13, 2021

Type Package

Title Send Email Messages

Version 0.7.0

Description A light, simple tool for sending emails with minimal dependencies.

URL <https://datawookie.github.io/emayili/>

BugReports <https://github.com/datawookie/emayili/issues>

License GPL-3

Language en-GB

Imports base64enc, commonmark, curl (>= 4.0), digest, dplyr, glue, htmltools, httr, logger, magrittr, mime, purrr, rmarkdown, stringi, stringr, tidyr, urltools, vctrs, xfun, xml2

Suggests cld2, cld3, gpg, here, memoise, testthat (>= 2.1.0), roxygen2, showtext, Microsoft365R

SystemRequirements The function render() requires Pandoc (<http://pandoc.org>). To use PGP/GnuPG encryption requires gpg.

Encoding UTF-8

RoxygenNote 7.1.2

KeepSource true

NeedsCompilation no

Author Andrew B. Collier [aut, cre, cph],
Matt Dennis [ctb],
Antoine Bichat [ctb] (<<https://orcid.org/0000-0001-6599-7081>>),
Daniel Fahey [ctb],
Johann R. Kleinbub [ctb],
Panagiotis Moulos [ctb],
Swechhya Bista [ctb],
Colin Fay [ctb] (<<https://orcid.org/0000-0001-7343-1846>>)

Maintainer Andrew B. Collier <andrew@fathomdata.dev>

Repository CRAN

Date/Publication 2021-12-13 09:30:09 UTC

R topics documented:

address	3
addresses	4
append.envelope	5
as.address	6
as.character.envelope	6
as.character.header	7
as.character.MIME	7
as.character.vctrs_address	8
attachment	8
comments	9
compare	10
compliant	11
cutoff	11
display	12
domain	13
encrypt	13
envelope	14
format.vctrs_address	16
html	16
keywords	18
local	18
mime-parameters	19
new_address	20
parties	20
precedence	21
print.envelope	22
print.vctrs_address	23
qp	23
raw	24
render	24
request_receipt_read	26
response	27
sensitivity	28
server	29
subject	32
text	33

address	<i>Email Address</i>
---------	----------------------

Description

Create an address object which represents an email address.

Usage

```
address(email = NA, display = NA, local = NA, domain = NA, normalise = TRUE)
```

Arguments

email	Email address.
display	Display name.
local	Local part of email address.
domain	Domain part of email address.
normalise	Whether to try to normalise address to RFC-5321 requirements.

Details

Implemented as an [S3 vector class](#).

Value

An address object, representing an email address.

Examples

```
address("gerry@gmail.com")
address("gerry@gmail.com", "Gerald")
address(
  c("gerry@gmail.com", "alice@yahoo.com", "jim@aol.com"),
  c("Gerald", "Alice", NA)
)
```

addresses	<i>Add address fields to message</i>
-----------	--------------------------------------

Description

Add address fields to message

Usage

```
to(msg, ..., append = TRUE)
```

```
cc(msg, ..., append = TRUE)
```

```
bcc(msg, ..., append = TRUE)
```

```
from(msg, addr = NULL)
```

```
reply(msg, addr = NULL)
```

```
return_path(msg, addr = NULL)
```

```
sender(msg, addr = NULL)
```

Arguments

msg	A message object.
...	Addresses.
append	Whether to append or replace addresses.
addr	Single address.

Value

A message object.

Examples

```
# Populating the To field.
msg <- envelope()
msg %>% to("bob@gmail.com", "alice@yahoo.com")
msg %>% to("bob@gmail.com", "alice@yahoo.com")
msg %>% to(c("bob@gmail.com", "alice@yahoo.com"))

# Populating the Cc field.
msg <- envelope()
msg %>% cc("bob@gmail.com", "alice@yahoo.com")
msg %>% cc("bob@gmail.com", "alice@yahoo.com")
msg %>% cc(c("bob@gmail.com", "alice@yahoo.com"))
```

```
# Populating the Bcc field.
msg <- envelope()
msg %>% bcc("bob@gmail.com, alice@yahoo.com")
msg %>% bcc("bob@gmail.com", "alice@yahoo.com")
msg %>% bcc(c("bob@gmail.com", "alice@yahoo.com"))

msg <- envelope()

# Populating the From field.
msg %>% from("craig@gmail.com")

# Populating the Reply-To field.
msg <- envelope()
msg %>% reply("gerry@gmail.com")

# Populating the Return-Path field.
msg <- envelope()
msg %>% return_path("bounced-mail@devnull.org")

# Populating the Sender field.
msg <- envelope()
msg %>% sender("on_behalf_of@gmail.com")
```

append.envelope	<i>Append children to message</i>
-----------------	-----------------------------------

Description

Append children to message

Usage

```
## S3 method for class 'envelope'
append(x, child)
```

Arguments

x	Message object
child	A child to be appended

as.address *Create an address object*

Description

Create an address object

Usage

```
as.address(addr)
```

Arguments

addr An email address.

Value

An address object.

Examples

```
as.address("gerry@gmail.com")
as.address("Gerald <gerry@gmail.com>")
as.address(c("Gerald <gerry@gmail.com>", "alice@yahoo.com", "jim@aol.com"))
as.address("Gerald <gerry@gmail.com>, alice@yahoo.com, jim@aol.com")
as.address(c("Gerald <gerry@gmail.com>", "alice@yahoo.com, jim@aol.com"))
```

as.character.envelope *Create formatted message.*

Description

Accepts a message object and formats it as a MIME document.

Usage

```
## S3 method for class 'envelope'
as.character(x, ..., details = TRUE)
```

Arguments

x A message object.
 ... Further arguments passed to or from other methods.
 details Whether or not to display full message content.

Value

A formatted message object.

as.character.header *Create formatted header.*

Description

Accepts a header object and formats it as a header field.

Usage

```
## S3 method for class 'header'  
as.character(x, width = 28, ...)
```

Arguments

x	A header object.
width	The width of the head name field.
...	Further arguments passed to or from other methods.

Value

A formatted header field.

as.character.MIME *Convert MIME object to character vector*

Description

Convert MIME object to character vector

Usage

```
## S3 method for class 'MIME'  
as.character(x, ...)
```

Arguments

x	MIME object
...	Further arguments passed to or from other methods.

`as.character.vctrs_address`

Convert address object to character

Description

Convert address object to character

Usage

```
## S3 method for class 'vctrs_address'  
as.character(x, ...)
```

Arguments

`x` A vector of address objects.
`...` Further arguments passed to or from other methods.

Value

A character vector.

`attachment`

Add attachments to a message object

Description

Add attachments to a message object

Usage

```
attachment(  
  msg,  
  path,  
  name = NA,  
  type = NA,  
  cid = NA,  
  disposition = "attachment"  
)
```


Arguments

msg	A message object.
path	Path to file.
name	Name to be used for attachment (defaults to base name of path).
type	MIME type or NA, which will result in a guess based on file extension.
cid	Content-ID or NA.
disposition	How is attachment to be presented ("inline" or "attachment")?

Value

A message object.

Examples

```
path_mtcars <- tempfile(fileext = ".csv")
path_scatter <- tempfile(fileext = ".png")
path_cats <- system.file("cats.jpg", package = "emayili")

write.csv(mtcars, path_mtcars)

png(path_scatter)
plot(1:10)
dev.off()

msg <- envelope() %>%
  attachment(path_mtcars) %>%
  # This attachment will have file name "cats.jpg".
  attachment(path_cats, name = "cats.jpg", type = "image/jpeg") %>%
  attachment(path_scatter, cid = "scatter")

file.remove(path_scatter, path_mtcars)
```

comments

Add or query comments of message.

Description

Add or query comments of message.

Usage

```
comments(msg, comments = NULL)
```

Arguments

msg	A message object.
comments	Comments for the message.

Value

A message object or the comments of the message object (if comments is NULL).

See Also

[subject](#)

Examples

```
# Create a message and set the comments.
msg <- envelope() %>% comments("This is a comment")

# Retrieve the comments for a message.
comments(msg)
```

compare

Compare vectors

Description

Returns TRUE wherever elements are the same (including NA), and FALSE everywhere else.

Usage

```
compare(lhs, rhs)
```

Arguments

lhs	LHS of operation.
rhs	RHS of operation.

Value

A Boolean value.

compliant	<i>Tests whether an email address is syntactically correct</i>
-----------	--

Description

Checks whether an email address conforms to the **syntax rules**.

Usage

```
compliant(addr, error = FALSE)
```

Arguments

addr	An email address.
error	Whether to create an error if not compliant.

Details

An email address may take either of the following forms:

- local@domain or
- Display Name <local@domain>.

Value

A Boolean.

Examples

```
compliant("alice@example.com")  
compliant("alice?example.com")
```

cutoff	<i>Set or query message expiry or reply-by time</i>
--------	---

Description

Functions to specify the time at which a message expires or by which a reply is requested.

Usage

```
expires(msg, datetime = NULL, tz = "")  
replyby(msg, datetime = NULL, tz = "")
```

Arguments

msg	A message object.
datetime	Date and time.
tz	A character string specifying the time zone.

Details

Manipulate the Expires and Reply-By fields as specified in [RFC 2156](#).

Value

A message object.

Examples

```
envelope() %>%  
  expires("2030-01-01 13:25:00", "UTC")  
envelope() %>%  
  replyby("2021-12-25 06:00:00", "GMT")
```

display	<i>Extract display name</i>
---------	-----------------------------

Description

Extracts the display name from an email address.

Usage

```
display(addr)
```

Arguments

addr	An address object.
------	--------------------

Value

The display name or NA.

Examples

```
gerry <- as.address("Gerald <gerry@gmail.com>")  
display(gerry)
```

domain	<i>Extract domain of email address</i>
--------	--

Description

Extract domain of email address

Usage

```
domain(addr)
```

Arguments

addr	An address object.
------	--------------------

Value

A character vector.

Examples

```
domain("alice@example.com")
```

encrypt	<i>Encrypt or sign a message</i>
---------	----------------------------------

Description

Specify whether the message should be encrypted, signed or have a public key attached.

Usage

```
encrypt(msg, encrypt = TRUE, sign = TRUE, public_key = TRUE)
```

Arguments

msg	A message object.
encrypt	Whether to encrypt the message. If TRUE then the entire message will be encrypted using the private key of the sender.
sign	Whether to sign the message. If TRUE then the entire message will be signed using the private key of the sender.
public_key	Whether to attach a public key. If TRUE then the public key of the sender will be attached.

Details

If a recipient's email client is unable to decrypt an encrypted message then they will not be able to access the message contents.

Value

A message object.

Examples

```
## Not run:
msg <- envelope(
  to = "schunk@u-boat.com",
  subject = "Top Secret Message",
  text = "Immediate readiness. There are indications that the invasion has begun."
)
msg %>% encrypt()

## End(Not run)
```

envelope

Create a message.

Description

Create a message.

Usage

```
envelope(
  to = NULL,
  from = NULL,
  cc = NULL,
  bcc = NULL,
  reply = NULL,
  subject = NULL,
  importance = NULL,
  priority = NULL,
  text = NULL,
  html = NULL,
  encrypt = FALSE,
  sign = FALSE,
  public_key = FALSE
)
```

Arguments

to	See to() .
from	See from() .
cc	See cc() .
bcc	See bcc() .
reply	See reply() .
subject	See subject() .
importance	See importance() .
priority	See priority() .
text	See text() .
html	See html() .
encrypt	Whether to encrypt the message. If TRUE then the entire message will be encrypted using the private key of the sender.
sign	Whether to sign the message. If TRUE then the entire message will be signed using the private key of the sender.
public_key	Whether to attach a public key. If TRUE then the public key of the sender will be attached.

Value

A message object.

See Also

[subject\(\)](#), [from\(\)](#), [to\(\)](#), [cc\(\)](#), [bcc\(\)](#), [reply\(\)](#) and [encrypt\(\)](#).

Examples

```
# Create an (empty) message object.
#
msg <- envelope()

# Create a complete message object, specifying all available fields.
#
envelope(
  to = "bob@gmail.com",
  from = "craig@gmail.com",
  cc = "alex@gmail.com",
  bcc = "shannon@gmail.com",
  reply = "craig@yahoo.com",
  importance = "high",
  priority = "urgent",
  subject = "Hiya!",
  text = "Hi Bob, how are you?"
)
```

`format.vctrs_address` *Encode email addresses in a common format*

Description

Encode email addresses in a common format

Usage

```
## S3 method for class 'vctrs_address'  
format(x, ...)
```

Arguments

`x` A vector of address objects.
`...` Further arguments passed to or from other methods.

Value

A character vector.

`html` *Add an HTML body to a message object.*

Description

Add an HTML body to a message object.

Usage

```
html(  
  msg,  
  content,  
  disposition = "inline",  
  charset = "utf-8",  
  encoding = "quoted-printable",  
  css_files = c(),  
  language = FALSE,  
  interpolate = TRUE,  
  .open = "{{",  
  .close = "}}",  
  .envir = NULL  
)
```


Arguments

msg	A message object.
content	A string of message content.
disposition	Should the content be displayed inline or as an attachment? Valid options are "inline" and "attachment". If set to NA then will guess appropriate value.
charset	What character set is used. Most often either "UTF-8" or "ISO-8859-1".
encoding	How content is transformed to ASCII. Options are "7bit", "quoted-printable" and "base64". Use NA or NULL for no (or "identity") encoding.
css_files	Extra CSS files.
language	Language of content. If FALSE then will not include language field. If TRUE then will attempt to auto-detect language. Otherwise will use the specified language.
interpolate	Whether or not to interpolate into input using glue .
.open	The opening delimiter.
.close	The closing delimiter.
.envir	Environment used for glue interpolation. Defaults to parent.frame().

Value

A message object.

See Also

[text](#), [render](#)

Examples

```
# Inline HTML message.
envelope() %>% html("<b>Hello!</b>")

# Read HTML message from a file.
htmlfile <- tempfile(fileext = ".html")
cat("<p>Hello!</p>\n", file = htmlfile)
envelope() %>% html(htmlfile)

# You can pass a vector of character. Components will be separated by a
# "\n".
envelope() %>% html(c("<b>Hello</b>", "<p>World!</p>"))

# You can also pass a tagList from {htmltools}.
if (requireNamespace("htmltools", quietly = TRUE)) {
  library(htmltools)
  envelope() %>% html(tagList(h2("Hello"), p("World!")))
}
```

keywords	<i>Add or query keywords of message.</i>
----------	--

Description

Add or query keywords of message.

Usage

```
keywords(msg, ..., append = FALSE)
```

Arguments

msg	A message object.
...	Keywords.
append	Whether to append or replace keywords.

Value

A message object or the comments of the message object (if comments is NULL).

See Also

[to](#), [from](#), [cc](#), [bcc](#) and [reply](#)

Examples

```
# Create a message and set the keywords.
envelope() %>% keywords("newsletter, marketing")
envelope() %>% keywords("newsletter", "marketing")
envelope() %>% keywords(c("newsletter", "marketing"))

# Retrieve the keywords for a message.
msg <- envelope() %>% keywords("newsletter, marketing")
keywords(msg)
```

local	<i>Extract local part of email address</i>
-------	--

Description

Extract local part of email address

Usage

```
local(addr)
```

Arguments

addr An address object.

Value

A character vector.

Examples

```
local("alice@example.com")
```

 mime-parameters

Parameters for MIME functions

Description

These are parameters which occur commonly across functions for components of a MIME document.

Arguments

content	A string of message content.
disposition	Should the content be displayed inline or as an attachment? Valid options are "inline" and "attachment". If set to NA then will guess appropriate value.
charset	What character set is used. Most often either "UTF-8" or "ISO-8859-1".
encoding	How content is transformed to ASCII. Options are "7bit", "quoted-printable" and "base64". Use NA or NULL for no (or "identity") encoding.
language	Language of content. If FALSE then will not include language field. If TRUE then will attempt to auto-detect language. Otherwise will use the specified language.
description	Description of content.
name	Name used when downloading file.
filename	Path to a file.
boundary	Boundary string.
type	The MIME type of the content.
children	List of child MIME objects.
interpolate	Whether or not to interpolate into input using glue .
.open	The opening delimiter.
.close	The closing delimiter.
.envir	Environment used for glue interpolation. Defaults to parent.frame().

new_address	<i>Helper function for creating address objects</i>
-------------	---

Description

Helper function for creating address objects

Usage

```
new_address(  
    email = character(),  
    display = character(),  
    local = character(),  
    domain = character(),  
    normalise = TRUE  
)
```

Arguments

email	Email address.
display	Display name.
local	Local part of email address.
domain	Domain part of email address.
normalise	Whether to try to normalise address to RFC-5321 requirements.

Value

An address object, representing an email address.

parties	<i>Extract sender and recipient(s)</i>
---------	--

Description

Extract sender and recipient(s)

Usage

```
parties(msg)
```

Arguments

msg	A message object.
-----	-------------------

Value

A tibble.

Examples

```
email <- envelope() %>%
  from("Gerald <gerald@gmail.com>") %>%
  to(c("bob@gmail.com", "alice@yahoo.com")) %>%
  cc("Craig <craig@gmail.com>") %>%
  bcc(" Erin <erin@yahoo.co.uk >")

parties(email)
```

precedence

Add fields for message importance and priority

Description

A hint to influence transmission speed and delivery.

A hint to the message recipient about how important the message is.

Usage

```
priority(msg, priority = NULL)
```

```
importance(msg, importance = NULL)
```

Arguments

`msg` A message object.

`priority` Priority level. One of "non-urgent", "normal", or "urgent".

`importance` Importance level. One of "low", "normal", or "high".

Details

Does not influence transmission speed or delivery.

Value

A message object.

Examples

```
# How rapidly does the message need to be delivered?
#
envelope() %>%
  subject("Deliver this immediately!") %>%
  priority("urgent")

envelope(priority = "non-urgent") %>%
  subject("No rush with this.")

# How much attention should be paid by recipient?
#
envelope() %>%
  subject("Read this immediately!") %>%
  importance("high")

envelope(importance = "low") %>%
  subject("Not important at all. Just delete.")
```

```
print.envelope      Print a message object
```

Description

The message body will be printed if `details` is `TRUE` or if the `envelope_details` option is `TRUE`.

Usage

```
## S3 method for class 'envelope'
print(x, details = NA, ...)
```

Arguments

<code>x</code>	A message object.
<code>details</code>	Whether or not to display full message content.
<code>...</code>	Further arguments passed to or from other methods.

Examples

```
msg <- envelope() %>% text("Hello, World!")

print(msg)
print(msg, details = TRUE)

options(envelope_details = TRUE)
print(msg)
```

print.vctrs_address *Print an address object*

Description

Print an address object

Usage

```
## S3 method for class 'vctrs_address'  
print(x, ...)
```

Arguments

x	An address object.
...	Further arguments passed to or from other methods.

Examples

```
gerry <- as.address("gerry@gmail.com")  
print(gerry)
```

qp *Quoted-Printable encoding*

Description

Encode to and decode from Quoted-Printable encoding.

Usage

```
qp_encode(x, crlf = CRLF)  
  
qp_decode(x)
```

Arguments

x	A string for encoding or decoding.
crlf	End-of-line characters.

Value

An encoded string for qp_encode() or a decoded string for qp_decode().

Examples

```
qp_encode("Mieux vaut être seul que mal accompagné.")  
qp_decode("Mieux vaut =C3=AAtre seul que mal accompagn=C3=A9.")
```

raw *Extract raw email address*

Description

Strips the display name off an email address (if present).

Usage

```
raw(addr)
```

Arguments

addr An address object.

Value

A raw email address.

Examples

```
gerry <- as.address("Gerald <gerry@gmail.com>")
raw(gerry)
```

render *Render Markdown into email*

Description

Render either Plain Markdown or R Markdown directly into the body of an email.

If input is a file then it will be interpreted as R Markdown if its extension is either "Rmd" or "Rmarkdown". Otherwise it will be processed as Plain Markdown.

Usage

```
render(
  msg,
  input,
  params = NULL,
  squish = TRUE,
  css_files = c(),
  include_css = c("rmd", "bootstrap", "highlight"),
  language = FALSE,
  interpolate = TRUE,
  .open = "{{",
  .close = "}}",
  .envir = NULL
)
```


Arguments

<code>msg</code>	A message object.
<code>input</code>	The input Markdown file to be rendered or a character vector of Markdown text.
<code>params</code>	A list of named parameters that override custom parameters specified in the YAML front-matter.
<code>squish</code>	Whether to clean up whitespace in rendered document.
<code>css_files</code>	Extra CSS files.
<code>include_css</code>	Whether to include rendered CSS from various sources (" <code>rmd</code> " — native R Markdown CSS; " <code>bootstrap</code> " — Bootstrap CSS; " <code>highlight</code> " — highlight.js CSS).
<code>language</code>	Language of content. If FALSE then will not include language field. If TRUE then will attempt to auto-detect language. Otherwise will use the specified language.
<code>interpolate</code>	Whether or not to interpolate into input using glue .
<code>.open</code>	The opening delimiter.
<code>.close</code>	The closing delimiter.
<code>.envir</code>	Environment used for glue interpolation. Defaults to <code>parent.frame()</code> .

Value

A message object.

Plain Markdown

Plain Markdown is processed with `commonmark::markdown_html()`.

R Markdown

R Markdown is processed with `rmarkdown::render()`.

Regardless of what output type is specified in the input file, `render()` will always use the "`html_document`" output format.

See Also

[text](#), [html](#)

Examples

```
# Plain Markdown

markdown <- "[This](https://www.google.com) is a link."
filename <- "message.md"

# Render from Markdown in character vector.
msg <- envelope() %>% render(markdown)
```

```

# Create a file containing Markdown
cat(markdown, file = filename)

# Render from Markdown in file.
msg <- envelope() %>% render(filename)

# Cleanup.
file.remove(filename)

# R Markdown

filename <- "gh-doc.Rmd"

# Create an Rmd document from template.
rmarkdown::draft(
  filename,
  template = "github_document",
  package = "rmarkdown",
  edit = FALSE
)

# Check for suitable version of Pandoc (https://pandoc.org/).
#
# Need to have version 2.0 or greater to support required --quiet option.
#
pandoc <- rmarkdown::find_pandoc()
suitable_pandoc <- !is.null(pandoc$dir) && grepl("^2", pandoc$version)

# Render from Rmd file.
if (suitable_pandoc) {
  msg <- envelope() %>%
    render(filename, include_css = c("rmd", "highlight"))
}

# Cleanup.
file.remove(filename)

```

request_receipt_read *Request read receipt*

Description

Request the recipient to acknowledge that they have read the message. Inserts MDN (Message Disposition Notification) header entries.

Usage

```
request_receipt_read(msg, addr = NULL)
```

Arguments

msg A message object.
 addr Single address.

Value

A message object.

response	<i>Add In-Reply-To and References header fields</i>
----------	---

Description

Add In-Reply-To and References header fields

Usage

```
inreplyto(msg, msgid, subject_prefix = "Re: ")
references(msg, msgid, subject_prefix = "Re: ")
```

Arguments

msg A message object.
 msgid A message ID. This would be the contents of the Message-ID field from another message.
 subject_prefix Prefix to add to subject. If specified will be prepended onto the Subject field. Set to NULL if not required.

Value

A message object.

Examples

```
envelope() %>% inreplyto("<6163c08e.1c69fb81.65b78.183c@mx.google.com>")
# Now for German.
envelope() %>%
  inreplyto("6163c08e.1c69fb81.65b78.183c@mx.google.com", "AW: ")
# And also for Danish, Norwegian and Swedish (but not Finnish!).
envelope() %>%
  references("6163c08e.1c69fb81.65b78.183c@mx.google.com", "SV: ")
```

sensitivity	<i>Set or query message sensitivity</i>
-------------	---

Description

Manipulate the Sensitivity field as specified in [RFC 2156](#).

Usage

```
sensitivity(msg, sensitivity = NULL)
```

Arguments

msg	A message object.
sensitivity	Sensitivity level. One of "personal", "private", or "company-confidential".

Value

A message object.

Examples

```
# Not sensitive.
envelope() %>%
  subject("Your daily dose of spam")

# Sensitive personal message.
envelope() %>%
  subject("The results from your test") %>%
  sensitivity("personal")

# Sensitive private message.
envelope() %>%
  subject("Your OTP (don't show this to anybody!)") %>%
  sensitivity("private")

# Sensitive business message.
envelope() %>%
  subject("Top Secret Strategy Document") %>%
  sensitivity("company-confidential")
```

server *Create a SMTP server object.*

Description

Create a SMTP server object.

Usage

```
server(
  host,
  port = 25,
  username = NULL,
  password = NULL,
  insecure = FALSE,
  reuse = TRUE,
  helo = NA,
  protocol = NA,
  pause_base = 1,
  max_times = 5,
  ...
)

gmail(username, password, ...)

sendgrid(password, ...)

mailgun(username, password, ...)

sendinblue(username, password, ...)

mailersend(username, password, ...)
```

Arguments

host	DNS name or IP address of the SMTP server.
port	Port that the SMTP server is listening on.
username	Username for SMTP server.
password	Password for SMTP server or API key.
insecure	Whether to ignore SSL issues.
reuse	Whether the connection to the SMTP server should be left open for reuse.
helo	The HELO domain name of the sending host. If left as NA then will use local host name.
protocol	Which protocol (SMTP or SMTPS) to use for communicating with the server. Default will choose appropriate protocol based on port.

pause_base	Base delay (in seconds) for exponential backoff. See rate_backoff .
max_times	Maximum number of times to retry.
...	Additional curl options. See <code>curl::curl_options()</code> for a list of supported options.

Value

A function which is used to send messages to the server.

Sendgrid

To use SendGrid you'll need to first [create an API key](#). Then use the API key as the password.

SendGrid will accept messages on ports 25, 587 and 2525 (using SMTP) as well as 465 (using SMTPS).

Mailgun

To use Mailgun you'll need to first register a sender domain. This will then be assigned a username and password.

Mailgun will accept messages on ports 25 and 587 (using SMTP) as well as 465 (using SMTPS).

Sendinblue

To use Sendinblue you'll need to first create an account. You'll find your SMTP username and password in the SMTP & API section of your account settings.

MailerSend

To use MailerSend you'll need to first create an account. You'll find your SMTP username and password under Domains. See [How to send emails via SMTP with MailerSend](#).

Although this is not likely to be a problem in practice, MailerSend insists that all messages have at minimum a valid subject and either text or HTML content.

Examples

```
# Set parameters for SMTP server (with username and password).
smtp <- server(
  host = "smtp.gmail.com",
  port = 587,
  username = "bob@gmail.com",
  password = "bd40ef6d4a9413de9c1318a65cbae5d7"
)

# Set parameters for a (fake) testing SMTP server.
#
# More information about this service can be found at https://www.smtpbucket.com/.
#
smtp <- server(
  host = "mail.smtpbucket.com",
```

```
    port = 8025
  )

  # Create a message
  msg <- envelope() %>%
    from("bob@gmail.com") %>%
    to("alice@yahoo.com")

  # Send message (verbose output from interactions with server)
  ## Not run:
  smtp(msg, verbose = TRUE)

  ## End(Not run)

  # To confirm that the message was sent, go to https://www.smtpbucket.com/ then:
  #
  # - fill in "bob@gmail.com" for the Sender field and
  # - fill in "alice@yahoo.com" for the Recipient field then
  # - press the Search button.

  # With explicit HELO domain.
  #
  smtp <- server(host = "mail.example.com",
                 helo = "client.example.com")

  # Set parameters for Gmail SMTP server. The host and port are implicit.
  smtp <- gmail(
    username = "bob@gmail.com",
    password = "bd40ef6d4a9413de9c1318a65cbae5d7"
  )

  # Set API key for SendGrid SMTP server.
  smtp <- sendgrid(
    password = "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"
  )

  # Set username and password for Mailgun SMTP server.
  smtp <- mailgun(
    username = "postmaster@sandbox9ptce35fdf0b31338dec4284eb7aaa59.mailgun.org",
    password = "44d072e7g2b5f3bf23b2b642da0fe3a7-2ac825a1-a5be680a"
  )

  # Set username and password for Sendinblue SMTP server.
  smtp <- sendinblue(
    username = "bob@gmail.com",
    password = "xsmtpsib-c75cf91323adc53a1747c005447cbc9a893c35888635bb7bef1a624bf773da33"
  )

  # Set username and password for MailerSend SMTP server.
  smtp <- mailersend(
    username = "NS_Pf3ALM@gmail.com",
    password = "e5ATWL1TnWWDaKeE"
  )
)
```

subject	<i>Add or query subject of message.</i>
---------	---

Description

Add or query subject of message.

Usage

```
subject(  
  msg,  
  subject = NULL,  
  interpolate = TRUE,  
  .open = "{{",  
  .close = "}}",  
  .envir = NULL  
)
```

Arguments

msg	A message object.
subject	A subject for the message.
interpolate	Whether or not to interpolate into input using glue .
.open	The opening delimiter.
.close	The closing delimiter.
.envir	Environment used for glue interpolation. Defaults to <code>parent.frame()</code> .

Value

A message object or the subject of the message object (if subject is NULL).

See Also

[to](#), [from](#), [cc](#), [bcc](#) and [reply](#)

Examples

```
# Create a message and set the subject  
msg <- envelope() %>% subject("Updated report")  
  
# Retrieve the subject for a message  
subject(msg)
```

text	<i>Add a text body to a message.</i>
------	--------------------------------------

Description

Add text/plain content to a message.

Usage

```
text(
  msg,
  content,
  disposition = "inline",
  charset = "utf-8",
  encoding = "7bit",
  language = FALSE,
  interpolate = TRUE,
  .open = "{",
  .close = "}",
  .envir = NULL
)
```

Arguments

msg	A message object.
content	A string of message content.
disposition	Should the content be displayed inline or as an attachment? Valid options are "inline" and "attachment". If set to NA then will guess appropriate value.
charset	What character set is used. Most often either "UTF-8" or "ISO-8859-1".
encoding	How content is transformed to ASCII. Options are "7bit", "quoted-printable" and "base64". Use NA or NULL for no (or "identity") encoding.
language	Language of content. If FALSE then will not include language field. If TRUE then will attempt to auto-detect language. Otherwise will use the specified language.
interpolate	Whether or not to interpolate into input using glue .
.open	The opening delimiter.
.close	The closing delimiter.
.envir	Environment used for glue interpolation. Defaults to parent.frame().

Details

The text/plain format is described in [RFC 2646](#).

Uses `glue::glue()` to evaluate expressions enclosed in brackets as R code.

Value

A message object.

See Also

[html](#), [render](#)

Examples

```
msg <- envelope() %>% text("Hello!")

# Using {glue} interpolation.
#
name <- "Alice"
msg <- envelope() %>% text("Hello {name}.")

print(msg, details = TRUE)

# Disable {glue} interpolation.
#
msg <- envelope() %>% text("This is a set: {1, 2, 3}.", interpolate = FALSE)
```

Index

address, 3
addresses, 4
append.envelope, 5
as.address, 6
as.character.envelope, 6
as.character.header, 7
as.character.MIME, 7
as.character.vctr_address, 8
attachment, 8

bcc, 18, 32
bcc (addresses), 4
bcc(), 15

cc, 18, 32
cc (addresses), 4
cc(), 15
comments, 9
commonmark::markdown_html(), 25
compare, 10
compliant, 11
cutoff, 11

display, 12
domain, 13

encrypt, 13
encrypt(), 15
envelope, 14
expires (cutoff), 11

format.vctr_address, 16
from, 18, 32
from (addresses), 4
from(), 15

glue, 17, 19, 25, 32, 33
gmail (server), 29

html, 16, 25, 34
html(), 15

importance (precedence), 21
importance(), 15
inreplyto (response), 27

keywords, 18

local, 18

mailersend (server), 29
mailgun (server), 29
mime-parameters, 19

new_address, 20

parties, 20
precedence, 21
print.envelope, 22
print.vctr_address, 23
priority (precedence), 21
priority(), 15

qp, 23
qp_decode (qp), 23
qp_encode (qp), 23

rate_backoff, 30
raw, 24
references (response), 27
render, 17, 24, 34
reply, 18, 32
reply (addresses), 4
reply(), 15
replyby (cutoff), 11
request_receipt_read, 26
response, 27
return_path (addresses), 4
rmarkdown::render(), 25

sender (addresses), 4
sendgrid (server), 29
sendinblue (server), 29

sensitivity, [28](#)

server, [29](#)

subject, [10](#), [32](#)

subject(), [15](#)

text, [17](#), [25](#), [33](#)

text(), [15](#)

to, [18](#), [32](#)

to (addresses), [4](#)

to(), [15](#)