NAME

S-nail [v14.9.24]— send and recei ve Internet mail

SYNOPSIS

s-nail [**-DdEFinv~**#] [**-:** *spec*] [**-A** *account*] [:**-a** *attachment*:] [:**-b** *bcc-addr*:]
[:**-C** *"field: body"*:] [:**-c** *cc-addr*:] [**-M** *type* | **-m** *file* | **-q** *file* | **-t**]
[**-r** *from-addr*] [:**-S** *var*[=*value*]:] [**-s** *subject*] [:**-T** *"field: addr"*:]
[:**-X** *cmd*:] [:**-Y** *cmd*:] [**-.**] :*to-addr*: [**--** :*mta-option*:]

s-nail [**-DdEeHiNnRv~**#] [**-:** *spec*] [**-A** *account*] [:**-C** *"field: body"*:] [**-L** *spec*]
[**-r** *from-addr*] [:**-S** *var*[=*value*]:] [**-u** *user*] [:**-X** *cmd*:] [:**-Y** *cmd*:]
[**--** :*mta-option*:]
s-nail [**-DdEeHiNnRv~**#] [**-:** *spec*] [**-A** *account*] [:**-C** *"field: body"*:] **-f**
[**-L** *spec*] [**-r** *from-addr*] [:**-S** *var*[=*value*]:] [:**-X** *cmd*:] [:**-Y** *cmd*:] [*file*]
[**--** :*mta-option*:]

s-nail -h | **--help**
s-nail -V | **--version**

TABLE OF CONTENTS

NAME[1]
SYNOPSIS[2]
TABLE OF CONTENTS[3]
DESCRIPTION[4]
- **Options**[5]
- **A starter**[6]
- **On sending mail, and non-interactive mode**[7]
- **Compose mode**[8]
- **On reading mail, and more on interactive mode**[9]
- **HTML mail and MIME attachments**[10]
- **Mailing lists**[11]
- **Character sets**[12]
- **Message states**[13]
- **Specifying messages**[14]
- **On terminal control and line editor**[15]
- **Coloured display**[16]
- **Signed and encrypted messages with S/MIME**[17]
- **On URL syntax and credential lookup**[18]
- **Encrypted network communication**[19]
- **Handling spam**[20]

COMMANDS[21]
- **Command modifiers**[22]
- **Old-style argument quoting**[23]
- **Shell-style argument quoting**[24]
- **Shell-style expansions**[25]
- **Message list arguments**[26]
- **Raw data arguments for codec commands**[27]
- **Filename transformations**[28]
- **Commands**[29]

COMMAND ESCAPES[30]

INTERNAL VARIABLES[31]
Initial settings[32]
Built-in variables[33]
ENVIRONMENT[34]
FILES[35]
Resource files[36]
The mime.types files[37]
The Mailcap files[38]
The .netrc file[39]
EXAMPLES[40]
S/MIME step by step[41]
Using CRLs with S/MIME or TLS[42]
FAQ[43]
S-nail shortly hangs on startup[44]
I cannot login to Google mail (via OAuth)[45]
But, how about XOAUTH2 / OAUTHBEARER?[46]
Not "defunctional", but the editor key does not work[47]
Can S-nail git-send-email?[48]
Howto handle stale dotlock files[49]
IMAP CLIENT[50]
SEE ALSO[51]
HISTORY[52]
AUTHORS[53]
CAVEATS[54]
BUGS[55]

DESCRIPTION

Warning! *v15-compat[615]* (with value) is default since v14.10.0, and the manual expects this context; Most old (other context) documentation has been removed. S-nail (S-nail) will see major changes until v15.0 (circa 2023). Some backward incompatibilities cannot be avoided, for example **COMMANDS[21]** will change to **Shell-style argument quoting[24]**.

S-nail provides a simple and friendly environment for sending and receiving mail. It is intended to provide the functionality of the POSIX `mailx`(1)[683] command, is MIME capable, and optionally offers extensions for line editing, S/MIME, SMTP and POP3, among others. Through many **COMMANDS[21]** and **INTERNAL VARIABLES[31]** users are given tools for email appraisal and management, as well as increasingly powerful, reliable scripting capabilities.

Options

`-:` *spec*, **`--resource-files`**=..

Controls loading (as via **`source`[284]**) of **Resource files[36]**. *spec* consists of case-insensitive letters: '`s`' for the system-wide`s-nail.rc` [659], '`u`' for the personal file`MAILRC` [636] (`~/.mailrc`[660]), and '`x`' for a compiled-in copy of the (upstream) system-wide file. The letters '`-`' and '`/`' disable usage of resource files. Order matters, default is '`su`'. This option overrides **`-n`[76]**.

`-A` *name*, **`--account`**=..

Activate user **`account`[144]** *name* after program startup is complete (resource files loaded, only **`-X`**[87] commands to follow), and switch to its **primary system mailbox[136]** (*inbox[453]*). Upon failure the program e**`xit`[311]**s if used non-interactively, or if any of *errexit[419]* or *posix[523]* are set.

-a *file*[=[!]input-charset[#[!]output-charset]], **--attach**=..
(Send mode) Attach *file*, subject to tilde expansion (see **Filename transformations**[28] and **folder**[203]). In**Compose mode** [8] the **COMMAND ESCAPES**[30] **~@**[320] and especially the scriptable **~^**[322] provide alternatives for attaching files.

If *file* is not accessible but contains an equal-sign '**=**' a character set specification is split off. If only an input one is given it is fixated and no conversion is applied; an empty, or the special string hyphen-minus '**-**' means*ttyc harset*[611]. If an output one is given the conversion is performed on-the-fly, not considering file type nor content; however, empty string or hyphen-minus '**-**' enforce the default **Character sets**[12] conversion (`-a file`, `-a file=#`, and `-a file=-#-` are identical), later applied after MIME-classifying *file* (**HTML mail and MIME attachments**[10], **The mime.types files**[37]). Without `,+iconv,` in *features*[425] only this mode is available. The character set names may be prefixed with exclamation mark '**!**' to enforce base64 *mime-encoding*[479] of the attachment.

-b *addr*, **--bcc**=..
(Send mode) Send a blind carbon copy of the message to *addr*, invisible for other recipients. May be used multiple times. See also **On sending mail, and non-interactive mode**[7].

-C *"name: content"*, **--custom-header**=..
Create a custom header *name* that lasts for the entire session. Content follows after a colon '**:**', for example `-C "Blah: Neminem laede; imo omnes, quantum potes, juva"`. May be used multiple times; *name* may not be a standard header. Adjustable custom headers can be created via *customhdr*[409]; in **Compose mode**[8] **~^**[322] (**COMMAND ESCAPES**[30]) and **digmsg**[178] are the most powerful options.

-c *addr*, **--cc**=..
(Send mode) Like **-b**[60], but adds carbon copies (visible recipients).

-D, **--disconnected**
[Option] Startup with *disconnected*[676] **set**[271].

-d, **--debug**
Enter a debug-only sandbox mode by setting *debug*[412], as via **-S**[80] *debug* or **set**[271] *debug*. Also see**-v**[86].

-E, **--discard-empty-messages**
(Send mode) **set**[271] *skipemptybody*[558] and discard messages with an empty message part body, successfully.

-e, **--check-and-exit**
If messages are present (in the system *inbox*[453] or the one given by **-f**[68]) exit with status zero 0, with non-zero otherwise. Control message selection by **Specifying messages**[14] via **-L**[72]. Quickrun: does not open an interactive session.

-F
(Send mode) Save the message in a file named after the local part of the first recipient's address, overwriting *record*[537], but honouring *outfolder*[493].

-f, **--file**
Open the user's **secondary mailbox**[137] `MBOX`[638], or a given argument *file*, instead of the **primary system mailbox**[136] (note *hold*[446], *keepsave*[457]). *file* is not an option argument, but taken from the command line after command line option processing. It is inspected for protocol specifications and undergoes **Filename transformations**[28], as if given to **folder**[203]. As a special case that requires read-only mode (**-R**[78]) hyphen-minus '**-**' denotes standard input (in MBOX or EML format) that can also be a pipe instead of a regular file.

-H, **--header-summary**
: Display a summary of **headers**[214] of the given **folder**[203] (dependent on **-u**[84], *inbox*[453] or MAIL[634], or as specified via **-f**[68]), then exit. *showlast*[553] is ignored. Control message selection by **Specifying messages**[14] via **-L**[72]. Quickrun: does not open an interactive session. Tip: COLUMNS[624] can be honoured in batch mode (**-#**[90]).

-h, **--help**
: Show a brief usage summary; use **--long-help** to list long options.

-i
: **set**[271] *ignore*[451] to ignore tty interrupt signals.

-L *spec*, **--search**=..
: Display a summary of **headers**[214] of messages that match *spec* (**Specifying messages**[14]) in the **folder**[203] (dependent on **-u**[84], *inbox*[453] or MAIL[634], or as specified via **-f**[68]), then exit. *showlast*[553] is ignored. Nothing is displayed with **-e**[66], only the exit status denotes matches. Quickrun: does not open an interactive session. Tip: COLUMNS[624] can be honoured in batch mode (**-#**[90]).

-M *type*
: (Send mode) Will flag standard input with MIME Content-Type: *type* (**HTML mail and MIME attachments**[10], **The mime.types files**[37]), and use it as the main message body. [v15 behaviour may differ] Using this option will bypass processing of *message-inject-head*[473] and *message-inject-tail*[474]. Also see**-m**[74], **-q**[77], **-t**[83].

-m *file*
: (Send mode) Initialize the message body from MIME classified *file*. [v15 behaviour may differ] Using this option will bypass processing of *message-inject-head*[473] and *message-inject-tail*[474]. Also see**-q**[77], **-M**[73], **-t**[83].

-N, **--no-header-summary**
: **unset**[272] *header*[438] to inhibit displaying the summary of **headers**[214] when opening a **folder**[203].

-n
: Inhibit reading the system-wide s-nail.rc[659].

-q *file*, **--quote-file**=..
: (Send mode) Initialize the message body from *file*; only non-interactively this may denote standard input (hyphen-minus '-'). Also see**-M**[73], **-m**[74], **-t**[83].

-R, **--read-only**
: Any mailbox will be opened read-only as via **Folder**[202].

-r *from-addr*, **--from-address**=..
: The RFC 5321 reverse-path used for relaying and delegating messages, for example to report back delivery errors, is derived from *from*[436] (or *sender*[551]). However, a file-based (local) *mta*[482] (Message-Transfer-Agent) will instead use LOGNAME[633]. With this option *from-addr* is assigned to *from*[436], and in addition file-based *mta*[482]s are invoked with **-f** *from-addr*. If*from-addr* includes non-address components (*fullnames* [437]), these are instead passed via **-F** *name*. If*from-addr* is an empty string *fr om*[436] (or *sender*[551]) is evaluated whenever *mta*[482] is invoked; also see *r-option-implicit*[535]. Even though not a recipient the shquote *expandaddr*[422] flag is supported.

  Remarks: many MTA installations and sites disallow setting an explicit reverse-path, but for members of dedicated user groups, or after MTA reconfiguration.

-S *var*[=value], **--set**=..
: **set**[271] (or, with a 'no' prefix as documented in**INTERN AL VARIABLES**[31], **unset**[272]) *var*iable and optionally assign *value*, if supported, evaluated as if specified

within dollar-single-quotes (**Shell-style argument quoting**[24]). Upon failure the program will exit if any of *errexit*[419] or *posix*[523] is set. Settings established via **-S** cannot be changed from within **Resource files**[36], or an **-A** [57]ccount switch; they become mutable again for **-X**[87] commands.

-s *subject*, **--subject**=..
(Send mode) Specify a message subject. Newline (NL) and carriage-return (CR) are normalized to space (SP).

-T *"field: addr"*, **--target**=..
(Send mode) Add *addr*, parsed like a message header address line (see **-t** [83]), and supporting the same modifiers, to the list of recipients targeted by *field*: supported are `bcc`, '`cc`', `fcc`, and '`to`'. Field and body (address) are separated by a colon '`:`' and optionally blank (space, tabulator) characters. The`shquote` *e xpandaddr*[422] flag is supported. *addr* This option may be used multiple times.

-t, **--template**
(Send mode) Standard input is expected to contain one or multiple plain text message headers, an empty line, and the message body. [v15 behaviour may differ] Readily prepared MIME mail messages cannot be passed. Header lines are parsed as follows. A line starting with number sign '#' in the first column is ignored. A header can span multiple consecutive lines if follow lines start with (ignored) whitespace.

Recipients will be *expandaddr*[422] checked, and added onto the command line ones: `To:`, `Cc:`, `Bcc:`; the line is parsed as a single recipient with the modifier `?single`, for example `To?single: exa, <m@ple>`. `Fcc:` is supported (see **Compose mode**[8]). A subject specified via `Subject:` is used in favour of the command line option **-s** [81].

More optional headers are `Reply-To:` (possibly overriding *reply-to*[542]), `Sender:` (*sender*[551]), `From:` (*from*[436] and / or option **-r** [79]). Normally created automatically, but used if specified are `Message-ID:`, `In-Reply-To:`, `References:` and `Mail-Followup-To:`. All other custom header fields (see**-C** [61], *customhdr*[409], **~^**[322]) are passed through as-is, and in conjunction with the options **-~** [89] or **-#** [90] **COMMAND ESCAPES**[30] are evaluated. Also see**-M** [73], **-m**[74], **-q**[77].

-u *user*, **--inbox-of**=..
Open the **primary system mailbox**[136] of *user*, appropriate privileges presumed; identical to **-f** %user.

-V, **--version**
Print *version*[617] and exit. The command**version** [305] will also show *features*[425]: `$ s-nail -:/ -#v -Xversion -Xx`.

-v, **--verbose**
set[271] *verbose*[616]. (Multiple levels.) Also see**-d** [64].

-X *cmd*, **--startup-cmd**=..
Add (the multiline) *cmd* (block) to a list evaluated before normal operation starts, as via **source**[284]. Correlates with**-#** [90] and *errexit*[419].

-Y *cmd*, **--cmd**=..
Add (the multiline) *cmd* (block) to a list evaluated after normal operation has started. It is evaluated successively in the given order, and as if given on the program's standard input — before interactive prompting begins in interactive mode, after standard input has been consumed otherwise.

-~, **--enable-cmd-escapes**
Enable **COMMAND ESCAPES[30]** in **Compose mode[8]**, even in non-interactive use cases. This can for example be used to format the composed message text:

```
$ ( echo 'line    one. Word.     Word2.';
    echo '~| /usr/bin/fmt -tuw11' ) |
  LC_ALL=C s-nail -d~:/ -Sttycharset=utf-8 bob@exam.ple
```

-#, **--batch-mode**
Enable batch mode: standard input is made line buffered, all (interactive) commands are made available, processing of **COMMAND ESCAPES[30]** is enabled in **Compose mode[8]**, and diverse **INTERNAL VARIABLES[31]** are adjusted for batch necessities, exactly as via **-S[80]**: *emptystart[418]*, no*errexit[419]*, no*header[438]*, no*posix[523]*, *quiet[527]*, *sendwait[552]*, *typescript-mode[612]* as well as to `/dev/null`[657]: `MAIL`[634], `MBOX`[638] and *inbox[453]*. The values of `COLUMNS`[624] and `LINES`[631] are acted upon. For example :

```
$ for name in bob alice@exam.ple lisa@exam.ple; do
    printf 'mail %s\n~s ubject\nText\n~.\n' "${name}"
  done |
  LC_ALL=C s-nail -#:x -Smta=test \
    -X'alias bob bob@exam.ple'
```

-., **--end-options**
Force termination of option processing (prevent "option injection" attacks), and forcefully enter send mode.

In difference to *mta-arguments[484]* the setting of *expandargv[424]* is checked before *`mta-option`* arguments given after a '`--`' command line separator will be passed to file-based*mta [482]*s (Message-Transfer-Agent) during the session. The `shquote` constraint of *expandaddr[422]* applies to recipient addresses on the command line. For more see **On sending mail, and non-interactive mode[7]**.

```
$ s-nail -#:/ -X 'addrcodec enc <silver@go> Hey, ho' -Xx
```

A starter

S-nail is a direct descendant of BSD Mail, itself a successor to the Research UNIX mail which "was there from the start" according to **HISTORY[52]**. As a message user agent (MUA) it represents the user side of the UNIX mail system, the traditional server Message-Transfer-Agent (MTA) was `sendmail`(8)[684], and for compatibility a binary of this name usually exists to this day. If*featur es[425]* announces the [Option]al SMTP *mta[482]* message delivery does not require the server side.

This program strives for POSIX `mailx`(1)[685] compliance, however *posix[523]* (**INTERNAL VARIABLES[31]**) or its **`environ`[193]**mental equivalent `POSIXLY_CORRECT`[642] (**ENVIRONMENT[34]**) needs to be set to tweak behaviour accordingly. There is an important deviation: POSIX **Shell-style argument quoting[24]** is ([v15 behaviour may differ] increasingly) used instead of POSIX mailx's **Old-style argument quoting[23]**, which is believed to be a feature. The built-in as well as the (default) global `s-nail.rc`[659] **Resource files[36]** also bend standard imposed settings.

For example, *hold[446]* and *keepsave[457]* are **`set`[271]** in order to disable the default (**Message states[13]**) automatic moving of messages to the **secondary mailbox[137]** `MBOX`[638], and *keep[455]* to not remove empty system MBOX mailbox files (or all empty mailboxes in *posix[523]* mode) to avoid mangling of file permissions when files eventually get recreated.

Even if the opened **`folder`[203]** is empty interactive mode is entered due to *emptystart[418]*, in **Compose mode[8]** *editheaders[417]* enables header editing and *fullnames[437]* avoids address skinning, when **`reply`[261]**ing responded messages are *quote[528]*d, prefixed with an *indentprefix[454]* that also deviates from standard imposed settings, and *followup-to-honour[431]* and *reply-to-honour[543]* are set to

comply to sender address desires. Fully enabled is *mime-counter-evidence[478]*.

User credentials and settings are easily addressable by grouping them in **account[144]**s. The file mode creation mask can be managed with *umask[613]*. Files and shell pipe output can be**source [284]**d for **eval[127]**uation, also during startup from within the **Resource files[36]**. Informational context can be available by **set[271]**ting *verbose[616]* or *debug[412]* (as via **-v[86]**, **-d[64]**). Many un∗ commands, like **unaccount[145]**, **unalias[148]**, **unalternates[150]**, **uncommandalias[167]** etc. support an asterisk '∗' wildcard argument that matches all covered settings.

On sending mail, and non-interactive mode

To send a message to one or more people give their email addresses (and *fullnames[437]*) on the command line, the options **-b** [60] and **-c** [62] add (blind) carbon copy recipients. When delivered through a local *mta[482]* (Message-Transfer-Agent) plain system-local user names can be addressed. The message text will be read from standard input:

```
# Via test MTA
$ echo Hello, world | s-nail -:/ -Smta=test -s test $LOGNAME

# Via sendmail(1) MTA
$ </dev/null s-nail -:x -s test $LOGNAME

# Debug dry-run mode:
$ </dev/null LC_ALL=C s-nail -d -:/ \
   -Sttycharset=utf8 -Sfullnames \
   -b bcc@exam.ple -c cc@exam.ple -. \
   '(Lovely) Bob <bob@exam.ple>' dave@exam.ple

# With SMTP (no real sending due to -d debug dry-run)
$ LC_ALL=C s-nail -d -:/ -Sttycharset=utf8 \
    -S mta=smtps://me@exam.ple:465 -Ssmtp-config=-auth \
    -S from=scriptreply@exam.ple \
    --attach /etc/passwd --end-options \
    dave@exam.ple < /tmp/letter.txt
```

Plain user names are expanded through **alias[147]** and *mta-aliases[483]*, all recipients are subject to **alternates[149]** filtering. A valid local user <name> in angle brackets (an invalid address) expands to a qualified address if *hostname[447]* is not set or non-empty; if empty conversion is left up to the *mta[482]*. *expandaddr[422]* offers fine-grained control over allowed recipients and more.

Recipients are classified as follows: any name that starts with a vertical bar '|' specifies a pipe: the sh(1)[686]ell command following the '|' is executed with the message available on its standard input. Other than that hyphen-minus '-' or any name that starts with solidus '/' or dot solidus './' is treated as a file. Any other name which contains a commercial at '@' is an (email) address. Any other name which starts with a plus sign '+', or which contains a solidus '/' but no exclamation mark '!' or percent sign '%' before that is a mailbox name. What remains is treated as an (email) address. Classification can be avoided by using a Fcc: header, see **Compose mode[8]**.

```
$ echo bla | s-nail -S expandaddr -s test ./mbox.mbox
$ echo bla | s-nail -Sexpandaddr -stest '|cat >> ./mbox.mbox'
$ echo safe | LC_ALL=C s-nail -:/ -Smta=test \
    --set expandaddr=fail,-all,+addr,failinvaddr \
    -Sttycharset=utf8 -S mime-force-sendout -S fullnames \
    -s test -. 'Imagine John <cold@turk.ey>'
```

A lot of configuration can be **set[271]** generally. The envelope sender address for example via *from[436]*, especially with the built-in SMTP *mta[482]* a *hostname[447]* must be set. **Character sets [12]** for message text and MIME part content are configurable via *sendcharsets[549]*, input data is expected to be in *ttycharset[611]*. Emails need a*mime-encoding [479]*, MIME parts aka attachments need a **mimetype[227]**, usually taken out of **The mime.types files[37]**. Saving copies of sent messages in a *record[537]* mailbox may be desirable — as for most mailbox **folder[203]**s **Filename transformations[28]** are performed.

account[144]s aid in arranging complete configurations. Alternatively so-called variable chains that automatically pick `USER@HOST` or `HOST` context-dependent variants could be sufficient: for example **set[271]** *mta[482]*=smtp://yaa@exam.ple would find *smtp-config-yaa@exam.ple*, *smtp-config-exam.ple* and *smtp-config[573]*, in order. For more see **On URL syntax and credential lookup[18]** and **INTERNAL VARIABLES[31]**.

To avoid environmental noise scripts should be isolated by excluding configuration files via **-: [56]**, and use repetitions of **-S** [80] to specify variables:

```
$ env LC_ALL=C s-nail -:/ \
    -Sttycharset=utf-8 -Smime-force-sendout \
    -Sexpandaddr=fail,-all,failinvaddr \
    -S mta=smtps://me@exam.ple:465 -Ssmtp-config=-allmechs,plain \
    -S from=scriptreply@exam.ple \
    -s 'Subject to go' -a attachment_file \
    -Sfullnames --end-options \
    'Recipient 1 <rec1@exam.ple>' rec2@exam.ple \
    < content_file
```

As shown scripts can fake a locale **ENVIRONMENT[34]**, the above specifies the all-compatible 7-bit clean `LC_ALL[628]` "C", but nonetheless takes and sends UTF-8 message text via *ttycharset[611]*. If character set conversion is compiled in (*features[425]* includes `,+iconv,`) invalid (according to *ttycharset[611]*) input data would cause errors: setting *mime-force-sendout[480]* will classify input as binary data as a last resort, and therefore allow message creation. (Such content can be inspected either by installing a *pipe-TYPE/SUBTYPE[511]* handler for `application/octet-stream`, or possibly automatically through *mime-counter-evidence[478]*).

In interactive mode messages can be send with the command **mail[224]** and a list of recipient addresses, in the entered **Compose mode[8] COMMAND ESCAPES[30]**, like **~?[319]**, can then be used:

```
$ s-nail -:/ -Squiet -Semptystart -Sfullnames -Smta=test
"/var/spool/mail/user": 0 messages
? mail "Recipient 1 <rec1@exam.ple>", rec2@exam.ple
...
? # Will do the right thing (tm)
? m rec1@exam.ple rec2@exam.ple
```

Compose mode

In compose mode of interactive sessions, or when requested via **-~** [89] or **-# [90]**, lines beginning with '~' (the value of *escape[421]* in fact) are **COMMAND ESCAPES[30]**. For example **~v[346]** will start `VISUAL[650]` and **~e[328]** `EDITOR[626]` to revise the message, the potent **~^[322]** can manage attachments and headers. [Option]ally **~?[319]** shows a summary of available command escapes.

`Fcc:` header values are not classified like message recipients, therefore names with vertical bars or commercial ats can be used. They are subject to the checks of *expandaddr[422]*, undergo **Filename transformations[28]** and are then interpreted like a **folder[203]** target. Any local file and pipe command recipient honours *mbox-fcc-and-pcc[469]*.

on-compose-enter[496], *on-compose-splice[498]*, *on-compose-leave[497]* and *on-compose-cleanup[495]* are hooks that may be set to **define[174]**d macros. The splice hook can operate on messages like a user, and use **COMMAND ESCAPES[30]**, **digmsg[178]** may be helpful to query and adjust status of message(s) otherwise. ([v15 behaviour may differ] The compose mode hooks work for **forward[210]**, **mail[224]**, **reply[261]** and variants; **resend[264]** and **Resend[263]** only provide the hooks *on-resend-enter[506]* and *on-resend-cleanup[505]*, which are pretty restricted due to the nature of the operation.)

Once ready **~.[315]** will leave compose mode, call according hooks, perform auto-insertions (*message-inject-tail[474]*, *autocc[379]*, *autobcc[378]*), and pass the final message to the *mta[482]*. Unless *ignoreeof[452]* is set typing end-of-transmission (EOT) control-D (‘^D’) at the beginning of an empty line has the same effect. **~x[348]** or **~q[339]** abort message composition, the latter saves the message draft in DEAD[625] unless no*save[546]* is set. Typing end-of-text (ETX) twice via control-C (‘^C’) equals **~q[339]**.

COMMANDS[21] which enter compose mode support change **local[130]**ization (**Command modifiers[22]**): covered changes will be reverted once compose mode is left.

On reading mail, and more on interactive mode

When invoked without recipients “receive mode” is entered, more oriented towards interactive use (**On terminal control and line editor[15]**, **Coloured display[16]**). Through many **COMMANDS[21]** mail can be read, sent and managed in this mode. It starts into a mailbox as via **folder[203]**: either the **primary system mailbox[136]** (of **-u[84]** *user*), or, with the option **-f[68]**, a given *file* or the secondary MBOX[638]. If this initial mailbox is empty the program**quit[252]**s unless *emptystart[418]* is **set[271]**.

With *header[438]* a *screen[547]*ful of *headline[439]*s of **headers[214]** is then shown, **sort[282]**ed according to *autosort[382]*. **z[312]** will move through the summary. Messages are uniquely identified by numbers counting from 1; the current – named “dot” – will either be the first new, the first unread, or the first (*showlast[553]*: the last) message of the mailbox, in view. By**Specifying messages [14]** selective **search[269]** results can be created.

At the *prompt[525]* **list[220]** shows all built-in commands in a lookup order that does not always correlate to the alphabetical one: names can be abbreviated, and POSIX standardized some abbreviations. (But) **commandalias[166]**es can be defined and listed. A overall summary **help[215]** is available, and [Option]ally also for a given command (expansion). **help[215]** and **list[220]** may react upon *verbose[616]*.

```
? help reply
? set verbose=2; help reply; unset verbose
```

type[297] (alias **print[251]**) displays headers and text content of the dot or the specified messages. Whether and when PAGER[640] is used for display instead of dumping to the *screen[547]* is controlled by *crt[408]*. **more[237]** always uses PAGER[640]. **top[294]** shows only the first *toplines[609]* of messages, even *topsqueeze[610]*d. Setting*mime-counter-e vidence[478]* can improve real-life display experience, and also see **HTML mail and MIME attachments[10]**.

```
? type :nu # new and unread
? type @'time drift' # that substring in Subject:
```

By default all message headers are **type[297]**d, but they may be **retain[265]**ed or **ignore[219]**d for a variety of applications via **headerpick[212]**, for example, to restrict what is **forward[210]**ed: **headerpick[212] forward retain** *from to cc subject*. In order to display all header fields of a message regardless of currently active ignore or retain lists use **Type[296]** and **Top[293]**; **Show** will instead show raw message content. Historically the global s-nail.rc[659] not only adjusts the list of displayed headers, but also sets *crt[408]*.

When reading the *inbox[453]*, or when **-f** [68] or **folder[203]** was given a mailbox name prefixed with the modifier '%:', turning it into a **primary system mailbox[136]**, read messages (**Message states[13]**) will be moved to a **secondary mailbox[137]**, the users MBOX[638] file, when the mailbox is left; *hold[446]* disables this automatic moving from a system- or primary- to the secondary mailbox. Messages may of course be **move[235]**d, whereas **copy[169]** keeps the original message. **write [309]** saves selected message parts.

One may **reply[261]** 'r' to the sender and all recipients (also placed in To: unless *recipients-in-cc[536]* is set) of a message, or **Reply[259]** 'R' exclusively to the sender. To comply with the sender's desired reply address the **quadoption[349]**s *followup-to-honour[431]* and *reply-to-honour[543]* should usually be set. A special recipient massage for **Mailing lists[11]** is applied by **Lreply[222]** and **Lfollowup[221]**. The message being replied to can be*quote [528]*d (value defines style). **forward[210]**ing a message includes the original message in the message body instead. It is possible to **resend[264]** or **Resend[263]** messages, the former only will add a series of Resent- headers; different to newly created messages editing is not possible and no copy will be saved in *record[537]* unless *record-resent[539]* is **set[271]**.

Messages can be **delete[176]**'d' and**undelete [177]**d. The S-nail session is ended quickly via **exit[195]** or **xit[311]**, a full program exit that includes mailbox state and line editor *history-file[442]* updates among others is performed by **quit[252]**.

HTML mail and MIME attachments

Messages with only a HTML part, or MIME (Multipurpose Internet Mail Extensions) alternative messages with only a useful HTML type part become more and more common. And there are MIME attachments containing numerous data types. No knowledge of those is necessary for saving message parts via **write[309]**, for other purposes a notion of MIME types is required. A small set of types is built-in, onto which **The mime.types files[37]** are added under *mimetypes-load-control[481]*. **mimetype[227]** can create further types, and dynamically manage the list. *mime-counter-e vidence[478]* tries to address the faulty MIME part declarations of real life by possibly providing better fitting MIME types.

A simple HTML-to-text filter is [Option]ally (*features[425]* contains ,+filter-html-tagsoup,) built-in, but other non-text MIME types cannot be handled directly: handlers need to be registered that either convert data to (re-)integratable plain text (called **copiousoutput[661]** mode), or display it externally, for example in a graphical window: this latter kind is not considered when messages are **type[297]**d, but only by **mimeview[229]**.

To install a handler for a MIME type an according *pipe-TYPE/SUBTYPE[511]* variable must be set, alternatively the higher-ranked per file extension *pipe-EXTENSION[510]* can be used. [Option] Standard mail user agent configuration (RFC 1524) and **The Mailcap files[38]** preferably share MIME handler knowledge in between many programs. **mimetype [227]** type-markers are inspected last. Except with *mime-alternative-favour-rich[477]* plain text alternatives are used.

The following example reintegrates HTML formatted by the text browsers lynx(1)[687] or elinks(1)[688], registers a JSON MIME type handled as plain text, and establishes a setting to open PDF parts in an external viewer, asynchronously and with some other magic attached, like **Command modifiers[22]**:

```
set noprompt # unset prompt for this early example

if "$features" !% ,+filter-html-tagsoup,
  #set pipe-text/html='?* elinks -force-html -dump 1'
  set pipe-text/html='?* lynx -stdin -dump -force_html'
endif

mimetype ?t application/json json
```

```
set pipe-application/pdf='?&=? \
  trap "rm -f \"${MAILX_FILENAME_TEMPORARY}\"" EXIT;\
  trap "trap \"\" INT QUIT TERM; exit 1" INT QUIT TERM;\
  mupdf "${MAILX_FILENAME_TEMPORARY}"'

define showhtml {
  \local set mime-alternative-favour-rich pipe-text/html=?h?
  \type "$@"
}
\commandalias html \\call showhtml
```

Mailing lists

Mailing lists can be flagged in the summary of **headers**[214] (in *headline*[439] via '%L'), and gain special treatment when sending mails: *followup-to-honour*[431] ensures Mail-Followup-To: headers are honoured when replying (**reply**[261], **followup**[207], **Lreply**[222], **Lfollowup**[221]), and *followup-to*[429] creates this header when **mail**[224]ing messages with a proper user setup (*from*[436], *sender*[551]); it may be created automatically, for example when list-replying via **Lreply**[222] or **Lfollowup**[221], when **followup**[207] or **reply**[261] is used and Mail-Followup-To: is honoured etc.

mlist[230] and **mlsubscribe**[232] manage the correlation of email addresses and mailing lists. With the [Option]al regular expression support addresses which contain any of the magic regular expression characters (^[*+?|$; see re_format(7)[689] or regex(7)[690], dependent on host system) can then match many addresses. It is not possible to escape the "magic": in order to match special characters as-is, bracket expressions must be used, for example **search**[269] @subject@'[]open bracket'.

```
? set followup-to followup-to-honour=ask-yes \
    reply-to-honour=ask-yes
? mlist a1@b1.c1 a2@b2.c2 '.*@lists\.c3$'
? mlsubscribe a4@b4.c4 exact@lists.c3
```

Known and subscribed lists differ in that for the latter user addresses are not included in generated Mail-Followup-To:. There are exceptions, for example if multiple lists are addressed and not all have the subscription attribute. When replying to a message its list address (List-Post: header) is temporarily treated like a known **mlist**[230]; dependent on *reply-to-honour*[543] an existing Reply-To: is used instead (if it is a single address on the same domain as List-Post:) in order to accept a list administrator's wish that is supposed to have been manifested like that.

For convenience and compatibility with mail programs that do not honour the non-standard M-F-T an automatic user entry in Cc: can be created via *followup-to-add-cc*[430] whenever the user is placed in Mail-Followup-To:, but is not a regular recipient of the message. *r eply-to-swap-in*[544] tries to deal with the address rewriting that many mailing-lists nowadays perform to work around DMARC etc. standard imposed problems.

Character sets

[Option] The user's locale is detected by looking at the LC_ALL[628] (see also LC_CTYPE[629], LANG[630]) **ENVIRONMENT**[34] variable, deriving and storing the according character set in *ttycharset*[611]: this character set is targeted when displaying data, and any user input data is expected to be in it, too.

When creating messages their character input data is classified. 7-bit clean text data and attachments will be classified as *charset-7bit*[396]. [Option]ally 8-bit data will be converted into members of *sendcharsets*[549] until a character set conversion succeeds. *c harset-8bit*[397] is the implied default last member of this list. If no 8-bit character set is capable to represent input data, no message will be sent, and its text will optionally

be *save[546]*d in DEAD[625]. If that is not acceptable, for example in scripts, *mime-force-sendout[480]* forces sending of non-convertible data as `application/octet-stream` classified binary content instead: like this recipients still have the option to inspect message content (for example via *mime-counter-evidence[478]*).

If the [Option]al character set conversion is not available (*features[425]* misses `,+iconv,`), *ttycharset[611]* is the only supported character set for non 7-bit clean data, and it is simply assumed it can be used to exchange 8-bit messages.

ttycharset[611] may also be given an explicit value to send mail in a completely "faked" locale, for example generate and send 8-bit UTF-8 input data in a pure 7-bit US-ASCII `LC_ALL=C` **ENVIRONMENT[34]** (as shown in **On sending mail, and non-interactive mode[7]**). Unfortunate: due to lack of programming interfaces reading mail will not truly work here: whereas *ttycharset[611]* might be addressable, any output will be made safely printable, as via **`vexpr[306] makeprint`**, according to the actual locale, which is not affected by *ttycharset*.

Classifying 7-bit clean data as *charset-7bit[396]* is a problem if the input character set (*ttycharset[611]*) is a multibyte character set that is itself 7-bit clean. For example, the Japanese character set ISO-2022-JP is, but is capable to encode the rich set of Japanese Kanji, Hiragana and Katakana characters: in order to notify recipients of this character set the mail message must be MIME encoded so that the character set ISO-2022-JP can be advertised, otherwise an invalid email message would result! To achieve this, the variable *charset-7bit[396]* can be set to ISO-2022-JP. (Today a better approach regarding email is the usage of UTF-8, which uses 8-bit bytes for non-US-ASCII data.)

When replying to a message and *reply-in-same-charset[540]* is set the character set of the message being replied to is tried first as a target character set (still being a subject of **`charsetalias[159]`** filtering, however). Another opportunity is*sendc harsets-else-ttycharset[550]* to reflect the user's locale automatically, it will treat *ttycharset[611]* as an implied member of (an unset) *sendcharsets[549]*.

[Option] When reading messages, text data is converted into *ttycharset[611]* as necessary. Unprintable characters and invalid byte sequences are detected and replaced by substitution characters. Character set mappings for source character sets can be established with **`charsetalias[159]`**, which may be handy to work around faulty or incomplete character set catalogues (one could for example add a missing LATIN1 to ISO-8859-1 mapping), or to enforce treatment of one character set as another one ("interpret LATIN1 as CP1252"). Also see*c harset-unknown-8bit[398]* for another hairy aspect of message interpretation.

In general, if a message saying "cannot convert from a to b" appears, either some characters are not appropriate for the currently selected (terminal) character set, or the needed conversion is not supported by the system. In the first case, it is necessary to set an appropriate`LC_ALL [628]` locale and/or the variable *ttycharset[611]*. The best results are usually achieved when running in a UTF-8 locale on a UTF-8 capable terminal, in which case the full Unicode spectrum of characters is available. In this setup characters from various countries can be displayed, while it is still possible to use more simple character sets for sending to retain maximum compatibility with older mail clients.

On the other hand the POSIX standard defines a locale-independent 7-bit "portable character set" that should be used when overall portability is an issue, the even more restricted subset named "portable filename character set" consists of A-Z, a-z, 0-9, period '`.`', underscore '_' and hyphen-minus '`-`'.

Message states

Several message states are distinguished. The state of a message is shown by its *headline[439]* in the summary of **`headers[214]`**, and **Specifying messages[14]** by their state is possible.

`new` Neither read nor moved to another state. Retained even in a **primary system mailbox[136]**.

`unread` Neither read nor moved to another state, but already present when the **`folder`**[203] was opened. Retained even in a **primary system mailbox**[136].

`read` Processed by one of **`~f`**[330], **`~m`**[336], **`~F`**[329], **`~M`**[335], **`copy`**[169], **`mbox`**[226], **`next`**[241], **`pipe`**[248], **`Print`**[250], **`print`**[251], **`top`**[294], **`Type`**[296], **`type`**[297], **`undelete`**[177]. **`dp`**[180] and **`dt`**[181] try to automatically "step" and **`type`**[297] the "next" logical message, and may thus mark multiple messages as read, **`delete`**[176] will do so if the internal variable *autoprint*[381] is set.

Except when left via **`exit`**[195], `read` messages of a **primary system mailbox**[136] are saved to the **secondary mailbox**[137] `MBOX`[638] unless *hold*[446] it set.

`deleted` Processed by one of **`delete`**[176], **`dp`**[180] and **`dt`**[181]. It may be**`undelete`**[177]d, but other than that disappeared.

`preserved` Marked via **`preserve`**[249], and it will be retained.

`saved` Processed by **`save`**[268] or **`write`**[309].

Except when left via **`exit`**[195], `saved` messages of a **primary system mailbox**[136] are deleted; they are instead saved to the **secondary mailbox**[137] `MBOX`[638] if *keepsave*[457] is set.

In addition to these states, otherwise meaningless flags that may be set exist; they are addressable when **Specifying messages**[14]. They are persistently saved with messages, and are portable between a set of widely used MUAs:

`answered`[151] Mark as having been answered.

`draft`[183] Mark as being a draft.

`flag`[200] Mark for special attention.

Specifying messages

[Only new quoting rules]Some **COMMANDS**[21] take **Message list arguments**[26], for example **`copy`**[169], **`delete`**[176], **`search`**[269], and **`type`**[297], and can perform actions on a number of messages at once.

For example, `delete 1 2` deletes the messages 1 and 2 (shall they are valid), whereas `delete 1-5` will delete messages 1 through 5. In sorted mode (**`sort`**[282], *autosort*[382]) `delete 1-5` will delete all messages that are located in between (and including) messages 1 through 5 in the sorted order, as shown by **`headers`**[214].

Errors that occur are tracked by the **INTERNAL VARIABLES**[31] *!*[351], *^ERR*[356] and companions, as well as the command exit status *?*[350]. For example *^ERR*[356]-BADMSG when requesting an invalid message, *^ERR*[356]-NOMSG if no applicable message can be found, *^ERR*[356]-CANCELED for missing informational data (mostly thread-related). *^ERR* [356]-INVAL for invalid syntax as well as *^ERR*[356]-IO for input/output errors can happen.

`.` The current message, the so-called "dot". Many commands use it if no specification was given.

`;` The previously current message; needs to be quoted.

`,` The dot's parent message. This is the message with the Message-ID given in the `In-Reply-To:` field, or the last entry of the `References:` field of the current message.

`-` The previous undeleted message, or the previous deleted one for **`undelete`**[177]. In **`sort`**[282]ed mode, the previous such message in the according order.

`+` The next undeleted message, or the next deleted one for **undelete[177]**. In**sort [282]**ed mode, the next such message in the according order.

`^` The first undeleted message, or the first deleted one for **undelete[177]**. In**sort [282]**ed mode, the first such message in the according order.

`$` The last message. In **sort[282]**ed mode, the last such one in the according order. Needs to be quoted.

`&``[`*`x`*`]` Selects the message addressed with *`x`* in threaded **sort[282]** mode, where *`x`* is any other message specification, and all messages from the thread that begins at it. Otherwise identical to *`x`*. Omitting *`x`* equals using dot.

`*` All messages.

`'` All messages matched by the **Message list arguments[26]** of the previous command; needs to be quoted. (Tip: to read all new messages `search :n` them, then successively type ‘’’ to invoke the default command **next[241]**; *showlast[553]* must be set for this to work.)

`x-y` An inclusive range of message numbers. Selectors that may also be used as endpoints include any of `.;-+^$`.

`address`
Case-insensitive “any substring matches” search for `From:` fields. It matches addresses even if *showname[554]* is set (and POSIX says “any address shown in a header summary shall be matchable in this form”); However, if the *allnet[368]* variable is set, only the local part of the address is evaluated, not ignoring case, and *showname[554]* is completely ignored. For finer control and match boundaries use ‘`@`’ instead.

`/string`
All messages that contain *`string`* in the subject field (case ignored according to locale). Also see *searchheaders[548]*. An empty*`string`* equals the last one used by this specification.

`[`*`@name-list`*`]`*`@expr`*
A case-insensitive search *`expr`*ession. If*`expr`* contains a commercial at ‘`@`’ *`name-list`* is effectively non-optional, but can be empty. *`name-list`* specifies a comma-separated list of header fields, without it only the `Subject:` is searched. [Option] *`expr`* and *`name-list`* are interpreted as extended regular expressions if they contain **magic regular expression characters[92]**.

An empty search *`expr`*ession tests for existence of the given header fields (compare **digmsg[178]**). Some header fields may be abbreviated: ‘`a`’, ‘`f`’, ‘`t`’, ‘`c`’, ‘`b`’ and ‘`s`’ will match `Author:`, `From:`, `To:`, `Cc:`, `Bcc:` and `Subject:`, respectively, and case-insensitively. But for the existence test `Author` indeed means all of `Author:`, `From:`, `Sender:`.

The special header fields `header` or ‘`<`’ can be used to search in (all of) the header(s), and`body` or ‘`>`’ and`text` or ‘`=`’ will perform full te xt searches – whereas the former searches only the body, the latter also searches the message header ([v15 behaviour may differ] this mode yet brute force searches over the entire decoded content of messages, including administrativa strings).

Even with regular expression support it is almost impossible to safely match only a specific address domain. To request that header content is treated as a list of addresses, and to strip those down to the plain email address which the *`expr`* is to be matched against, prefix the effective *`name-list`* with a tilde ‘`~`’:

```
'@~f,c@@a\.safe\.domain\.match$'
```

`:c` All messages of state or with matching condition ‘`c`’, where ‘`c`’ is one or multiple of the following colon modifiers:

a	**answered[151]** messages (*markanswered[468]*).
d	deleted messages (only for **undelete[177]** and **from[211]**).
f	**flag[200]**ged messages.
L	Messages with recipients that match **mlsubscribe[232]**d addresses.
l	Messages with recipients that match **mlist[230]**ed addresses.
n	new messages.
o	Old messages (any not in state read or new).
r	read messages.
S	[Option] Messages with unsure spam classification (see **Handling spam[20]**).
s	[Option] Messages classified as spam.
t	Messages marked as **draft[183]**.
u	unread messages.

[Option] IMAP-style SEARCH expressions may be used. These consist of keywords and criterions, and because **Message list arguments[26]** are split into tokens according to **Shell-style argument quoting[24]** it is necessary to quote the entire IMAP search expression in order to ensure that it remains a single token.

This addressing mode is available with all types of mailbox **folder[203]**s, a local search is performed as necessary. Strings must be enclosed by double quotation marks ‘"’ in their entirety if they contain whitespace or parentheses; within the quotes, only reverse solidus ‘\’ is recognized as an escape character. All string searches are case-insensitive. When the description indicates that the “envelope” representation of an address field is used, it means the search string is checked against both a list constructed as

```
'("name" "source" "local-part" "domain-part")'
```

for each address, and the addresses without real names from the respective header field. These search expressions can be nested using parentheses, see below for examples.

(*criterion*)
: All messages that satisfy the given *criterion*.

(*criterion1 criterion2 ... criterionN*)
: All messages that satisfy all of the given criteria.

(*or criterion1 criterion2*)
: All messages that satisfy either *criterion1* or *criterion2*, or both. To connect more than two criteria using ‘or’ specifications have to be nested using additional parentheses, as with (or a (or b c)), since (or a b c) really means ((a or b) and c). For a simple ‘or’ operation of independent criteria on the lowest nesting level, it is possible to achieve similar effects by using three separate criteria, as with (a) (b) (c).

(*not criterion*)
: All messages that do not satisfy *criterion*.

(*bcc "string"*)
: All messages that contain *string* in the envelope representation of the Bcc: field.

(*cc "string"*)
: All messages that contain *string* in the envelope representation of the Cc: field.

(*from "string"*)
: All messages that contain *string* in the envelope representation of the From: field.

(*subject "string"*)
: All messages that contain *string* in the Subject: field.

(*to "string"*)
: All messages that contain *string* in the envelope representation of the To: field.

(*header name "string"*)
: All messages that contain *string* in the specified Name: field.

(*body "string"*)
: All messages that contain *string* in their body.

(*text "string"*)
: All messages that contain *string* in their header or body.

(*larger size*)
: All messages that are larger than *size* (in bytes).

(*smaller size*)
: All messages that are smaller than *size* (in bytes).

(*before date*)
: All messages that were received before *date*, which must be in the form `d[d]-mon-yyyy`, where ‘`d`’ denotes the day of the month as one or two digits, `mon` is the name of the month – one of `Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec`, and `yyyy` is the year as four digits, for example `28-Dec-2012`.

(*on date*)
: All messages that were received on the specified date.

(*since date*)
: All messages that were received since the specified date.

(*sentbefore date*)
: All messages that were sent on the specified date.

(*senton date*)
: All messages that were sent on the specified date.

(*sentsince date*)
: All messages that were sent since the specified date.

()
: The same criterion as for the previous search. This specification cannot be used as part of another criterion. If the previous command line contained more than one independent criterion then the last of those criteria is used.

On terminal control and line editor

[Option] Terminal control depends on the standard UNIX libraries Termcap Access Library (libtermcap, −ltermcap) or Terminal Information Library (libterminfo, −lterminfo), and enhances or enables interactive usage aspects, like **Coloured display[16]**, and understanding of cursor and function keys for the Mailx-Line-Editor MLE for the `TERM[646]`inal found in the **ENVIRONMENT[34]**. Library interaction can be disabled via *termcap-disable[594]*, whereas *termcap[592]* is always inspected to learn about terminal capabilities.

[Option] The built-in Mailx-Line-Editor (MLE) should work in all environments which comply to the ISO C standard ISO/IEC 9899/AMD1:1995 (“ISO C90, Amendment 1”), and supports wide glyphs if possible (necessary functionality was removed from ISO C, but is included in X/Open Portability Guide Issue 4 (“XPG4”)). It offers some **`colour[164]`** support, and is tunable to an extend via *line-editor-config[458]*. Especially without terminal control some keys may be problematic, corrections via *termcap[592]* help – an example is shown in the **FAQ[43]** entry **Not "defunctional", but the editor key does not work[47]**. *line-editor-disable[460]* controls usage of the line editor.

[Option] Input from line editor prompts can be saved in a list of searchable and expandable **`history[216]`** entries. Any amount of leading whitespace prevents this saving, and the macro hook *on-history-addition[500]* allows further user control. Aspects of history like size and persistancy can be configured via *history-file[442]*, *history-gabby[443]*, *history-gabby-persist[444]* and *history-size[445]*.

The MLE supports a set of editing and control commands. By default (as) many (as possible) of these will be assigned to a set of single-letter control codes. These should work on any terminal, and can be generated by holding the “control” key while simultaneously pressing the key of desire, for example `control-D`.

[Option] Custom key **bind[153]**ings that map MLE or other commands to key sequences can be established. If so, the MLE will itself use**bind [153]** to install the built-in set from above, plus, unless prevented via *line-editor-no-defaults[461]*.

In the following list keys are shown via **Shell-style argument quoting[24]** and MLE command names in parenthesis.

- `\cA` Go to the start of the line (**mle-go-home**).
- `\cB` Move the cursor backward one character (**mle-go-bwd**).
- `\cC` `raise`(3)[691] `SIGINT` (**mle-raise-int**).
- `\cD` Forward delete the character under the cursor; **quit[252]**s if used on an empty line unless *ignoreeof[452]* is set (**mle-del-fwd**).
- `\cE` Go to the end of the line (**mle-go-end**).
- `\cF` Move the cursor forward one character (**mle-go-fwd**).
- `\cG` Cancel current operation, full reset. An active history search or tabulator expansion is reset first, restoring former line content: a second invocation is needed for a full reset, then. (**mle-reset**).
- `\cH` Backspace: backward delete one character (**mle-del-bwd**).
- `\cI` [Only new quoting rules]Horizontal tabulator: try to expand the word before the cursor, supporting the usual **Filename transformations[28]** (**mle-complete**; affected by **mle-quote-rndtrip[108]** and *line-editor-cpl-word-breaks[459]*).
- `\cJ` Newline: complete input line (**mle-commit**).
- `\cK` Cut all characters from the cursor to the end of the line (**mle-snarf-end**).
- `\cL` Repaint the line (**mle-repaint**).
- `\cN` [Option] Go to the next history entry (**mle-hist-fwd**).
- `\cO` ([Option]ally context-dependent) Invokes the command **dt[181]**.
- `\cP` [Option] Go to the previous history entry (**mle-hist-bwd**).
- `\cQ` Toggle roundtrip mode shell quotes, where produced, on and off (**mle-quote-rndtrip**). The default is configurable via *line-editor-config[458]*; also see **shcodec[274]**.
- `\cR` [Option] Complete line content from (the remaining) older history entries (**mle-hist-srch-bwd**). Search behaviour is configurable via *line-editor-config[458]*.
- `\cS` [Option] Complete line content from (the remaining) newer history entries (**mle-hist-srch-fwd**). Search behaviour is configurable via *line-editor-config[458]*.
- `\cT` Paste the snarf buffer (**mle-paste**).
- `\cU` The same as `\cA` followed by `\cK` (**mle-snarf-line**).
- `\cV` Prompts for a Unicode character (hexadecimal number without prefix, see **number syntax rules[135]**) to be inserted (**mle-prompt-char**). Note this needs to be assigned to a single-letter control code in order to become recognized and executed during input of a key-sequence (only three single-letter control codes can be used for that shortcut purpose); this control code is special-treated, then, and cannot be part of any other sequence (because it will trigger the **mle-prompt-char** function immediately).
- `\cW` Cut the characters from the one preceding the cursor to the preceding word boundary (**mle-snarf-word-bwd**).
- `\cX` Move the cursor forward one word boundary (**mle-go-word-fwd**).
- `\cY` Move the cursor backward one word boundary (**mle-go-word-bwd**).
- `\cZ` `raise`(3)[692] `SIGTSTP` (**mle-raise-tstp**).
- `\c[` Escape: reset a possibly used multibyte character input state machine and [Option]ally a lingering, incomplete key binding (**mle-cancel**). Note this needs to be assigned to a single-letter control code in order to become recognized and executed during input of a key-sequence (only three single-letter control codes can be used for that shortcut purpose). The control code may also be part of a multi-byte sequence, but if a sequence is active and the very control code is currently also an expected input, then the active sequence takes precedence and will consume the control code.

`\c\` ([Option]ally context-dependent) Invokes the command **`z`**`[312]`+.
`\c]` ([Option]ally context-dependent) Invokes the command **`z`**`[312]`$.
`\c^` ([Option]ally context-dependent) Invokes the command **`z`**`[312]`0.
`\c_` Cut the characters from the one after the cursor to the succeeding word boundary (**`mle-snarf-word-fwd`**).
`\c?` Backspace: **`mle-del-bwd`**`[101]`.
– **`mle-bell`**: ring the audible bell.
– [Option] **`mle-clear-screen`**: move the cursor home and clear the screen.
– **`mle-fullreset`**: different to **`mle-reset`**`[100]` this will immediately reset a possibly active search etc.
– **`mle-go-screen-bwd`**: move the cursor backward one screen width.
– **`mle-go-screen-fwd`**: move the cursor forward one screen width.
– **`mle-raise-quit:`** `raise`(3)[693] `SIGQUIT`.

Coloured display

[Option] Colours and font attributes through ANSI aka ISO 6429 SGR (select graphic rendition) escape sequences solely depend upon capabilities of the `TERM`[646]inal (see **On terminal control and line editor**[15]), and are configurable via **`colour`**`[164]` and **`uncolour`**`[165]`. It may be necessary to pass special command line options to make the `PAGER`[640] interpret the escape sequences correctly. *colour-disable*[400] controls usage of established colour mappings.

Colour setup could be conditionalized on interactive mode via **`if`**`[218]` (`terminal` indeed means “interactive”):

```
if terminal && "$features" =% ,+colour,
  colour iso view-msginfo ft=bold,fg=green
  colour iso view-header ft=bold,fg=red (from|subject) # regex
  colour iso view-header fg=red

  uncolour iso view-header from,subject
  colour iso view-header ft=bold,fg=magenta,bg=cyan
  colour 256 view-header ft=bold,fg=208,bg=230 "subject,from"
  colour mono view-header ft=bold
  colour mono view-header ft=bold,ft=reverse subject,from
endif
```

Signed and encrypted messages with S/MIME

[Option] S/MIME provides two central mechanisms: message signing and message encryption. Signing allows recipients to verify the message sender, and message encryption provides end-to-end security that reveals the clear text of a message only to the sender and the recipient.

A message is signed with a private key. This adds some data to the regular content which can be used to verify it was sent using a valid certificate, that the sender address is covered by the certificate, and that the content has not been altered. A signed message is received as clear text and can be handled without restrictions.

Encryption, in contrast, uses a public encryption key to make the message text invisible for all people except those who have access to a secret decryption key. The public encryption key must have been retrieved by other means, for example from previous communication, from securely visited web sites, from public key directories, etc. Because of the publicity messages should be signed before they become encrypted.

A central concept of S/MIME are certification authorities (CAs). These are trusted institutions which issue certificates that correlate a user’s private decryption and signing key (and its public variant) with the CA’s own by means of a certificate request. This makes it cryptographically possible to verify the key correlation of a CA and a user: verification will succeed if a CA certificate is found that is a(n in)direct signer of (a)

presented user certificate(s).

A set of CA certificates is usually available from the distributor of the operating system, or comes shipped with the used cryptographical library: reasonable security for S/MIME on the Internet is provided if the source that provides the set is trusted. A certificate cannot be more secure than the method its CA certificate has been retrieved with. A personal set of trusted certificates can be specified via *smime-ca-file[560]* and/or (with special preparation) *smime-ca-dir[559]*. Setting*smime-ca-no-defaults [562]* disables (additional) usage of the default certificate set.

The trusted set of certificates is used by the command **verify[304]** to ensure the given S/MIME message(s) can be trusted. If so, verified sender certificates that were embedded in signed messages can be saved locally with **certsave[158]**, and henceforth be used to encrypt further communication:

```
? certsave FILENAME
? set smime-encrypt-USER@HOST=FILENAME \
    smime-cipher-USER@HOST=AES256
```

To sign outgoing messages a personal S/MIME certificate is required. **S/MIME step by step [41]** shows examplarily how a personal S/MIME certificate can be obtained. In general, if the private key plus certificate "pair" is available, all that needs to be done is to set some variables, in particular *smime-sign-cert[569]*:

```
? set smime-sign-cert=myname@exam.ple.paired \
    smime-sign-digest=SHA512 \
    smime-sign from=myname@exam.ple
```

Variables of interest for S/MIME in general are *smime-ca-dir[559]*, *smime-ca-file[560]*, *smime-ca-flags[561]*, *smime-ca-no-defaults[562]*, *smime-crl-dir[564]*, *smime-crl-file[565]*. For S/MIME signing of interest are *smime-sign[568]*, *smime-sign-cert[569]*, *smime-sign-include-certs[571]* and *smime-sign-digest[570]*. Additional variables of interest for S/MIME en- and **decrypt[173]**ion: *smime-cipher[563]* and *smime-encrypt-USER@HOST[566]*. Variables of secondary interest may be *content-description-smime-message[406]* and *content-description-smime-signature[407]*. S/MIME is available if `,+smime,` is included in *features[425]*.

[v15 behaviour may differ] Note that neither S/MIME signing nor encryption applies to message subjects or other header fields yet. Thus they may not contain sensitive information for encrypted messages, and cannot be trusted even if the message content has been verified. When sending signed messages, it is recommended to repeat any important header information in the message text.

On URL syntax and credential lookup

FIXME new proto config stuff, query string?? FIXME more refs in between 'folder', ∗folder∗, and ∗inbox∗!!: FIXME #?0!0/NONE#1|sn_gm:/var/spool/mail/steffen? acc hulhu FIXME s-nail: Obsoletion warning: no more expansion of ∗folder∗ in "%": please set ∗inbox∗ For accessing protocol-specific resources Uniform Resource Locators (URL, RFC 3986) have become omnipresent. Here they are expected in a "normalized" variant that is not used in data exchange, but only meant as a compact, easy-to-use way of defining and representing information in a well-known notation; as such they do not conform to any real standard. Optional parts in brackets '`[]`' are optional either because other ways exist to define the information, or because the part is protocol specific. `/path` for example is used by [Option]al Maildir **folder[203]**s and IMAP, but not by POP3. Note `USER` and `PASSWORD` within an URL must be URL percent encoded (RFC 3986), see **urlcodec[302]**.

```
PROTOCOL://[USER[:PASSWORD]@]server[:port][/path]
```

Often **INTERNAL VARIABLES[31]** exist in "variable chains": the plain `variable` as well as `variable-HOST` and `variable-USER@HOST`. If a port was specified `HOST` really means `server:port`, not `server`. And this`USER` is ne ver in URL percent encoded form. For example, whether the hypothetical `smtp://hey%2Bhey:wings%3Aof@a.dove` including user and password

was used, or whether it was `smtp://a.dove` and they came from a different sources, to lookup the chain *tls-config-pairs[601]* first `tls-config-pairs-hey+hey@a.dove` is looked up, then `tls-config-pairs-a.dove`, then the plain variable at last.

The logic to collect credentials (of an **`account`[144]**) is as follows:

- A user is always required. If no `USER` has been given in the URL *user-HOST* and *user[614]* are looked up.

  [Option] Thereafter, when allowed by *netrc-lookup-HOST* or *netrc-lookup[489]*, **The .netrc file[39]** (of `LOGNAME`[633]) will be searched for an unambiguous (one possible match) `HOST` entry with a `login` name.

  If there is still no `USER` the verified, valid `LOGNAME`[633] is used.

- Authentication: unless otherwise noted the chain *PROTOCOL-auth-USER@HOST*, *PROTOCOL-auth-HOST*, *PROTOCOL-auth* is checked, falling back to a protocol-specific default as necessary.

- If no `PASSWORD` has been given in the URL, then if the `USER` has been found through the [Option]al *netrc-lookup[489]*, that may have also provided the password. Otherwise the chain *password-USER@HOST*, *password-HOST*, *password[508]* is looked up.

  [Option] Thereafter the (now complete) chain *netrc-lookup-USER@HOST*, *netrc-lookup-HOST*, *netrc-lookup[489]* is checked, if set the **`netrc`[239]** cache is searched for a password only (multiple user accounts for a single machine may exist as well as a fallback entry without user but with a password).

  If at that point a password is required but not available, then in interactive mode the user will be prompted.

Note: S/MIME verification works relative to the values found in the `From:` (or `Sender:`) header field(s), therefore *smime-sign[568]*, *smime-sign-cert[569]*, *smime-sign-include-certs[571]* and *smime-sign-digest[570]* will not be looked up using above's `USER` and `HOST` chains, but instead use values from the message that is being worked on. If no address matches *from[436]* is used. In unusual cases multiple and different `USER` and `HOST` combinations may therefore be involved – on the other hand those unusual cases become possible. The usual case is as short as:

```
set mta=smtp://USER:PASS@HOST smtp-config=-starttls \
    smime-sign smime-sign-cert=+smime.pair \
    from=myname@my.host
```

Encrypted network communication

[Option] SSL (Secure Sockets Layer) aka its successor TLS (Transport Layer Security) are protocols which aid in securing communication by providing a safely initiated and encrypted network connection. As part of each connection setup a set of certificates will be exchanged through which the identity of the network peer can be cryptographically verified against a local set of trusted certificates. If possible the TLS/SNI (Server-NameIndication) extension will be enabled to allow servers fine-grained control over the presented certificates.

A central concept of TLS are certification authorities (CAs). These are trusted institutions which issue certificates that correlate a partie's private key (and its public variant) with the CA's own by means of a certificate request. This makes it cryptographically possible to verify the key correlation of a CA and a party: verification will succeed if a CA certificate is found that is a(n in)direct signer of (a) presented party certificate(s).

A set of CA certificates is usually available from the distributor of the operating system, or comes shipped with the used cryptographical library: reasonable security for TLS on the Internet is provided if the source that provides the set is trusted. A certificate cannot be more secure than the method its CA certificate has been retrieved with. A personal set of trusted certificates can be specified via *tls-ca-file[596]* and/or (with special preparation) *tls-ca-dir[595]*. Setting*tls-ca-no-defaults [598]* disables (additional) usage of the default certificate set.

For inspection or other purposes, the certificate of a server (as seen when connecting to it) can be fetched with the command **`tls[292]`** (port can usually be the protocol name, too, and *tls-verify[608]* is taken into account here):

```
$ s-nail -vX 'tls certchain SERVER-URL[:PORT]; x'
```

Server certificates can also be verified through their fingerprint, eliminating the need for a local set of CA certificates: a message digest will be calculated and compared against *tls-fingerprint[605]*. A digest (algorithm) can be configured via *tls-fingerprint-digest[606]*; **`tls[292]`** can again be used:

```
$ s-nail -X 'set verbose; tls fingerprint SERVER-URL[:PORT]; x'
```

It depends on the protocol whether encrypted communication is available, and which configuration steps have to be taken to enable it. Some protocols, like POP3S, are implicitly encrypted, others, like POP3, can upgrade a plain text connection if so requested. For example, to use the `STLS` that POP3 offers *pop3-use-starttls[522]* needs to be set; here with a convenient **`shortcut[276]`**:

```
shortcut encpop1 pop3s://pop1.exam.ple

shortcut encpop2 pop3://pop2.exam.ple
set pop3-use-starttls-pop2.exam.ple

set mta=smtps://smtp.exam.ple:465
# Automatically upgrades unless smtp-config=-starttls
set mta=smtp://smtp.exam.ple
```

TLS libraries try to provide safe defaults, plenty of knobs however exist to adjust settings. For example certificate verification can be fine-tuned via *tls-ca-flags[597]*, and TLS configuration basics are accessible via *tls-config-pairs[601]*, for example to control protocol versions or cipher lists. In the past hints on how to restrict the set of protocols to highly secure ones were indicated, but as of the time of this writing the list of protocols or ciphers may need to become relaxed in order to be able to connect to some servers; the following example allows connecting to a "Lion" that uses OpenSSL 0.9.8za from June 2014:

```
set tls-config-pairs-lion@exam.ple='MinProtocol=TLSv1.1,\
    CipherString=TLSv1.2:!aNULL:!eNULL:\
      ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:\
      DHE-RSA-AES256-SHA:@STRENGTH'
```

The OpenSSL program `ciphers`(1)[694] should be referred to when creating a custom cipher list. Variables of interest for TLS in general are *tls-ca-dir[595]*, *tls-ca-file[596]*, *tls-ca-flags[597]*, *tls-ca-no-defaults[598]*, *tls-config-file[599]*, *tls-config-module[600]*, *tls-config-pairs[601]*, *tls-crl-dir[602]*, *tls-crl-file[603]*, *tls-rand-file[607]* as well as *tls-verify[608]*. Also see*tls-featur es[604]*. TLS is available if `+tls` is included in *features[425]*.

Handling spam

[Option] Several *spam-interface[579]*s for dealing with spam messages are configurable. The volatile `is-spam` state of classified messages can appear in the *attrlist[377]* of their *headline[439]* in the summary of **`headers[214]`**, and *`:s`* and *`:S`* select them when **Specifying messages[14]**.

- **spamrate**[289] rates the given messages and sets their `is-spam` flag accordingly. If the spam interface offers spam scores these can be shown in *headline*[439] by using the format ‘%$’.
- **spamham**[288], **spamspam**[291] and **spamforget**[287] interact with the Bayesian filter of the chosen interface.
- **spamclear**[286] and **spamset**[290] simply set and clear, respectively, the mentioned volatile `is-spam` message flag, they do not interact with *spam-interface*[579].

The `spamassassin`(1)[695] based *spam-interface*[579] `spamc` requires a running instance of the `spamd`(1)[696] server in order to function, started with the option **--allow-tell** shall Bayesian filter learning be possible.

```
$ spamd -i localhost:2142 -i /tmp/.spamsock -d [-L] [-l]
$ spamd --listen=localhost:2142 --listen=/tmp/.spamsock \
    --daemonize [--local] [--allow-tell]
```

Thereafter these interfaces can be used:

```
$ s-nail -Sspam-interface=spamc -Sspam-maxsize=500000 \
    -Sspamc-command=/usr/local/bin/spamc \
    -Sspamc-arguments="-U /tmp/.spamsock" -Sspamc-user=
or
$ s-nail -Sspam-interface=spamc -Sspam-maxsize=500000 \
    -Sspamc-command=/usr/local/bin/spamc \
    -Sspamc-arguments="-d localhost -p 2142" -Sspamc-user=
```

Using the generic filter approach allows usage of programs like `bogofilter`(1)[697]. Here is an example that assumes the program in `PATH`[641]:

```
$ s-nail -Sspam-interface=filter -Sspam-maxsize=500000 \
    -Sspamfilter-ham="bogofilter -n" \
    -Sspamfilter-noham="bogofilter -N" \
    -Sspamfilter-nospam="bogofilter -S" \
    -Sspamfilter-rate="bogofilter -TTu 2>/dev/null" \
    -Sspamfilter-spam="bogofilter -s" \
    -Sspamfilter-rate-scanscore="1;^(.+)$"
```

Messages must be locally present for scoring or Bayesian filter training purposes. Spam can be checked automatically when opening specific folders by setting a specialized form of *on-mailbox-open*[501].

```
define spamdelhook {
  # Server side DCC
  spamset (header x-dcc-brand-metrics "bulk")
  # Server-side spamassassin(1)
  spamset (header x-spam-flag "YES")
  del :s # TODO we HAVE to be able to do ‘spamrate :u ! :sS’
  move :S +maybe-spam
  spamrate :u
  del :s
  move :S +maybe-spam
}
set on-mailbox-open-SOMEFOLDER=spamdelhook
```

See also *spam-interface*[579], *spam-maxsize*[580], *spamc-command*[581], *spamc-arguments*[582], *spamc-user*[583], *spamfilter-ham*[584], *spamfilter-noham*[585], *spamfilter-nospam*[586], *spamfilter-rate*[587] and *spamfilter-rate-scanscore*[589].

COMMANDS

Input is read in lines. An unquoted reverse solidus ‘\’ at the end of a line “escapes” the newline character: it is discarded and the next line of input is used as a follow-up line, with all leading whitespace removed. Once a line is completed, the whitespace characters **space**, **tabulator**, **newline** as well as those defined by *ifs[449]* are removed from both ends. [Option] Prepending any whitespace prevents addition to the **history[216]**.

The line is then scanned for a command name, possibly prepended by **Command modifiers[22]**. Names may be abbreviated: the first matching the given prefix is used. A name may also be a **commandalias[166]**, it is then recursively expanded first. Once the command that shall be evaluated is known, the remains of the line will be interpreted according to command-specific rules, most often **Shell-style argument quoting[24]**.

This behaviour is different to the `SHELL[643]`, which is a programming language with syntactic elements of clearly defined semantics, and therefore capable to sequentially expand and evaluate individual elements of a line. `? set one=spoon two=$one` for e xample will never assign `spoon` to two, because it is the command **set[271]** that performs the assignment, long after the expansion has happened.

A list of all commands in lookup order that not always fits the alphabetical one due to POSIX standardized abbreviations is dumped by **list[220]**. [Option]ally**help [215]** (alias: **?[142]**) can show a documentation for the argument, as in ‘`?t`’, which should be a shorthand of `? type`; with these documentation strings both commands support a more *verbose[616]* mode that prints argument types and some more information which applies; a handy suggestion may thus be:

```
? define __xv {
  local set verbose; ignerr eval "${@}"; return ${?}
}
? commandalias xv '\call __xv'
? xv help set
```

Command modifiers

Multiple (case-insensitive) modifiers may be prepended to commands, the order of which is insignificant. Some apply to only a restricted command set, as shown by the [Option]al *verbose[616]* mode of **list[220]**.

**** Reverse solidus prevents **commandalias[166]** expansion (that may contain additional modifiers), for example `\echo` will always evaluate the command **echo[185]**, even if an (command)alias of that name exists. Need not be whitespace-separated from other modifiers or command names.

eval Construct a command by concatenating arguments, expanded according to **Shell-style argument quoting[24]**, thereafter separated by single space characters.

```
define xxx {
  echo "xxx arg <$1>"
  shift
  if $# -gt 0
    \xcall xxx "$@"
  endif
}
define yyy {
  eval "$@ ' ball"
}
call yyy '\call xxx' "b\$'\t'u ' "
call xxx arg <b      u>
```

```
call xxx arg <  >
call xxx arg <ball>
```

global When supported, apply changes in global scope despite an established **local[130]** scope. **Remarks:** for commands which support the **vput[132]** modifier this can lead to strange constructs like `\global local vput vpospar myvar quote` where the command is advised to work in global scope, but the variable assignment happens in local scope.

ignerr Ignore command errors, and do neither **quit[252]** the program with *errexit[419]* nor for the standardized exit cases of *posix[523]* mode. *? [350]* will contain the real exit status of the command regardless.

local When supported, apply changes only in local scope. The commands **set[271]**, **unset[272]**, **call[155]** and **xcall[310]** overload locality for built-in **ENVIRONMENT[34]** (**environ[193]**) and **INTERNAL VARIABLES[31]**: changes will be passed down the call chain, but their former state is restored once the "covered scope" is left: once execution is passed up to the callee of a **define[174]**d macro, once a different account is activated for **account[144]**, and some macros, notably *on-mailbox-open[501]*s, use their own specific notion of covered scope, here it will be extended until the **folder[203]** is left again.

This extended local scoping stacks up: for example, if `macro1` has it enabled and calls `macro2` normally, then all changes done by `macro2` or deeper in the call chain will still be reverted once the scope of `macro1` is left. Caveats: if in this example `macro2` changes to a different **account[144]** which sets some variables that are already covered by local scoping, their scope will be extended, and in fact leaving the **account[144]** will (thus) restore settings in (likely) global scope which actually were defined in a local, macro private context!

u [v15 behaviour may differ] Does not yet implement any functionality. When supported command arguments are interpreted as UTF-8 character data, regardless of *ttycharset[611]* (aka `LC_ALL[628]`).

vput When supported a (shell-expanded) variable name is expected as the first command argument: the computation result will be stored in that variable instead of the default location (usually written to standard output), which may be **local[130]**ized: ? `[local] vput = msgno; echo $msgno`.

The name must comply to `SHELL[643]` variable name rules and consist only of upper- and lowercase characters, digits, and the underscore; hyphen-minus may be used as a non-portable extension, leading digits and hyphen-minus, as well as a trailing hyphen-minus are not allowed. Variables which are linked to the **ENVIRONMENT[34]** may not use extensions. **INTERNAL VARIABLES[31]** other than writable (non-boolean) value variables may not be used; also storage may fail in that case nonetheless if content constraints are not satisfied. Any test or storage error causes the command as such to fail, with *![351]* set to *^ERR[356]*-NOTSUP and *?[350]* set to '`-1`', but some commands deviate, then documented.

wysh Can be used for some old and established commands to choose the new **Shell-style argument quoting[24]** rules over the traditional **Old-style argument quoting[23]**. This modifier is implied if *v15-compat[615]* is set to a non-empty value, which now is the default. This modifier will first become a no-op, and later be removed.

Old-style argument quoting

[v15 behaviour may differ] This section documents the traditional and POSIX standardized style of quoting non-message list arguments to commands which expect this type of arguments: whereas still used by the majority of such commands, the new **Shell-style argument quoting[24]** may be available even for those via **wysh[134]**. Nonetheless care must be taken, because only new commands have been designed with all the capabilities of the new quoting rules in mind, which can, for example, generate control characters.

- An argument can be enclosed between paired double-quotes `"argument"` or single-quotes `'argument'`; any whitespace, shell word expansion, or reverse solidus characters (except as described next) within the quotes are treated literally as part of the argument. A double-quote will be treated literally within single-quotes and vice versa. Inside such a quoted string the actually used quote character can be used nonetheless by escaping it with a reverse solidus '\', as in `"y\"ou"`.
- An argument that is not enclosed in quotes, as above, can usually still contain space characters if those spaces are reverse solidus escaped, as in `you\ are`.
- A reverse solidus outside of the enclosing quotes is discarded and the following character is treated literally as part of the argument.

Shell-style argument quoting

SHELL[643]-style, and therefore POSIX standardized, argument parsing, expansion, and quoting rules are used by most commands. [v15 behaviour may differ] Most new commands only support these new rules and are flagged [Only new quoting rules], some elder ones can use them via **wysh**[134]; in the future only this type of argument quoting will remain.

The command line is parsed from left to right and an input token is completed whenever an unquoted, otherwise ignored, metacharacter is seen. Metacharacters are vertical bar **|**, semicolon **;**, as well as all characters from the variable *ifs*[449], and/or **space**, **tabulator**, **newline**. The additionalSHELL [643] metacharacters ampersand **&**, left and right parenthesis **(**, **)** and less-than and greater-than signs **<**, **>** are treated as ordinary characters: they are either vivid parts of email addresses or **Filename transformations**[28]. Any unquoted number sign '#' that begins a new token starts a comment that extends to the end of the line, and therefore ends argument processing. An unquoted dollar sign '$' starts**Shell-style expansions**[25].

> **Compatibility note:** [v15 behaviour may differ] Note that even many new-style commands do not yet honour *ifs*[449] to parse their arguments: whereas the SHELL[643] is a language with syntactic elements of clearly defined semantics, S-nail parses entire input lines and decides on a per-command base what to do with the rest of the line. This also means that whenever an unknown command is seen all that can be done is cancellation of the processing of the remains of the line.
>
> It also often depends on an actual subcommand of a multiplexer command how the rest of the line should be treated, and until v15 we are not capable to perform this deep inspection of arguments. Nonetheless, at least the following commands which work with positional parameters fully support *ifs*[449] for an almost shell-compatible field splitting: **call**[155], **call_if**[156], **read**[253], **vpospar**[307], **xcall**[310].

Whereas the metacharacters **space**, **tabulator**, **newline** only complete an input token, vertical bar **|** and semicolon **;** also act as control operators and perform control functions. For now supported is semicolon **;**, it terminates a single command, therefore sequencing the line, and making the remainder of it subject to reevaluation. With sequencing, Multiple command argument types and quoting rules may therefore apply to a single line, which can be problematic before v15 and without *v15-compat*[615]; for example, the first of the following will cause surprising results:

```
? echo one; set verbose; echo verbose=$verbose.
? echo one; wysh set verbose; echo verbose=$verbose.
```

Quoting is a mechanism that removes the special meaning of metacharacters and reserved words, and prevents expansion. There are four quoting mechanisms: the escape character, single-quotes, double-quotes and dollar-single-quotes:

- The literal value of any character can be preserved by preceding it with the escape character reverse solidus ‘\’.
- Arguments which are enclosed in `'single-quotes'` retain their literal value. A single-quote cannot occur within single-quotes.
- The literal value of all characters enclosed in `"double-quotes"` is retained, with the exception of dollar sign ‘$’ that causes**Shell-style expansions [25]**, backquote (grave accent) ‘‘’ (which not yet means anything special), reverse solidus ‘\’ that escapes any of the characters dollar sign ‘$’, backquote (grave accent) ‘‘’, double-quote ‘"’ (to prevent ending the quote), and reverse solidus ‘\’ (to prevent escaping, that is, to embed a reverse solidus character as-is), but has no special meaning otherwise.
- Arguments enclosed in `$'dollar-single-quotes'` extend normal single quotes in that reverse solidus escape sequences are expanded as follows:

  ‘\a’ bell control character (ASCII and ISO-10646 BEL).
  ‘\b’ backspace control character (ASCII and ISO-10646 BS).
  ‘\E’ escape control character (ASCII and ISO-10646 ESC).
  ‘\e’ the same.
  ‘\f’ form feed control character (ASCII and ISO-10646 FF).
  ‘\n’ line feed control character (ASCII and ISO-10646 LF).
  ‘\r’ carriage return control character (ASCII and ISO-10646 CR).
  ‘\t’ horizontal tabulator control character (ASCII and ISO-10646 HT).
  ‘\v’ vertical tabulator control character (ASCII and ISO-10646 VT).
  ‘\\’ emits a reverse solidus character.
  ‘\’’ single quote.
  ‘\"’ double quote (need not be escaped, but can).
  `\NNN` eight-bit byte with the octal value `NNN` (one to three octal digits), optionally prefixed by an additional ‘`0`’. A 0 byte will suppress further output for the quoted argument.
  `\xHH` eight-bit byte with the hexadecimal value ‘`HH`’ (one or two hexadecimal characters, no prefix, see **number syntax rules[135]**). A 0 byte will suppress further output for the quoted argument.
  `\UHHHHHHHH`
  the Unicode / ISO-10646 character with the hexadecimal codepoint value `HHHHHHHH` (one to eight hexadecimal characters) — note that Unicode defines the maximum codepoint ever to be supported as `0x10FFFF` (in planes of `0xFFFF` characters each). This escape is only supported in locales that support Unicode (see **Character sets[12]**), in other cases the sequence will remain unexpanded unless the given code point is ASCII compatible or (if the [Option]al character set conversion is available) can be represented in the current locale. The character NUL will suppress further output for the quoted argument.
  `\uHHHH`
  Identical to `\UHHHHHHHH` except it takes only one to four hexadecimal characters.
  `\cX` Emits the non-printable (ASCII and compatible) C0 control codes 0 (NUL) to 31 (US), and 127 (DEL). Printable representations of ASCII control codes can be created by mapping them to a different, visible part of the ASCII character set. Adding the number 64 achieves this for the codes 0 to 31, here 7 (BEL): `7 + 64 = 71 = G`. The real operation is a bitwise logical XOR with 64 (bit 7 set, see **`vexpr`[306]**), thus also covering code 127 (DEL), which is mapped to 63 (question mark): `? vexpr ^ 127 64`.

  Whereas historically circumflex notation has often been used for visualization purposes of control codes, as in ‘`^G`’, the reverse solidus notation has been standardized: `\cG`. Some control codes also have standardized (ISO-10646, ISO C) aliases, as shown above (‘`\a`’, ‘`\n`’, ‘`\t`’ etc) : whenever such an alias exists it will be used for display purposes. The

control code NUL (`\c@`, a non-standard extension) will suppress further output for the remains of the token (which may extend beyond the current quote), or, depending on the context, the remains of all arguments for the current command.

`\$expression`
: Non-standard extension to embed **Shell-style expansions[25]**.

`\'{command}`
: Not yet supported, just to raise awareness: Non-standard extension.

Caveats:

```
? echo 'Quotes '${HOME}' and 'tokens" differ!"# no comment
? echo Quotes ${HOME} and tokens differ! # comment
? echo Don"'"t you worry$'\x21' The sun shines on us. $'\u263A'

# Note: different to sh(1)ell _language_:
? define xy {
?   echo $#, $1, $2, $*
? }
? set x='one two three'
? call xy $x four
1, one two three, four, one two three four
```

Shell-style expansions

Within **Shell-style argument quoting[24]** an unquoted dollar sign '`$`' triggers expression expansions. The simplemost expression is an optionally brace enclosed variable name, for example `$NAME` or `${NAME}`, which must honour **variable name rules[133]**: **INTERNAL VARIABLES[31]** as well as **ENVIRONMENT[34]** (shell) variables can be accessed and expanded through this mechanism.

Within pairs of double parenthesis `$(( ))` 64-bit signed integer arithmetic expansions are performed. Different to **`vexpr`[306]** no saturated mode is available (overflow constants will result), and errors are not trackable: a syntactically invalid expression causes the context (**COMMANDS[21]**) to fail due to invalid arguments. Variable NAMEs can be referenced: unset or empty values expand to 0, otherwise the value is interpreted as a self-contained expression to expand arithmetically first. Numbers prefixed with '`0x`' or '`0X`' denote hexadecimal (base 16) numbers, '`0`' indicates octal (base 8), and '`0b`' as well as '`0B`' denote binary (base 2) numbers. A permissive `NUMBER` parse mode (see example) is used for the `BASE#NUMBER` notation; `BASE` is an unsigned decimal in the range 2 and 64, inclusive. Power-of-two bases are parsed as unsigned integers (making for overflow constant differences). The following list of operators is sorted in decreasing precedence.

```
? set i=0; echo $((i + i)), $((i+=1)), $((i + i)); varshow i
0, 1, 2
set i=1
? set i=-2; eval echo \$(( 10# $i))
-2
```

Compatibility note: [v15 behaviour may differ] Of the following, only "The result is treated as the arithmetic expression to be evaluated" is true: The expression is treated as if it were within double quotes, but a double quote inside the parentheses is not treated specially. All tokens in the expression undergo parameter and variable expansion, command substitution, and quote removal. The result is treated as the arithmetic expression to be evaluated. Arithmetic expansions may be nested.

(, *)*
: Parenthesis can be used to form sub-expressions which are evaluated with the highest precedence.

VAR++, *VAR*--
: Postfix increment and decrement, for variables.

+, -
: Unary plus and minus.

++*VAR*, --*VAR*
: Prefix increment and decrement, for variables.

!, ~
: Logical and bitwise negation.

**
: Exponentiation.

*, /, %
: Multiplication, division, remainder (modulo).

+, -
: Addition and subtraction.

<<, >>, >>>
: Bitwise shifts, the last performs an unsigned right shift.

<=, >=, <, >
: Comparison.

==, !=
: Equality and inequality.

&
: Bitwise AND.

^
: Bitwise XOR.

|
: Bitwise OR.

&&
: Logical AND.

||
: Logical OR.

EXPR0 ? EXPR1 : EXPR2
: Conditional operator: if EXPR0 is true evalute EXPR1, otherwise evaluate EXPR2; this can be nested. Precedence note: for example `1?crt=10:crt=5` is an error, as it is `(1?(crt=10):crt)=5`; `1?crt=10:(crt=5)` is ok.

**=, *=, /=, %=, +=, -=, <<=, >>=, >>>=, &=, ^=, |=, =
: Assignment. It is subject to scoping rules;**local [130]**ly **set[271]**ting a variable before arithmetically expanding it is an option.

EXPR1 , EXPR2
: Comma (sequential evaluation).

Message list arguments

Many commands operate on message lists as documented in **Specifying messages[14]**. Input is first split into individual tokens via **Shell-style argument quoting[24]**, and then interpreted as said. If no explicit message list has been specified, many commands will search for and use the next message forward that satisfies the commands' requirements, and if there are no messages forward of the current message, the search proceeds backwards; if there are no good messages at all to be found, and the default ("dot") is not applicable, an error message is shown instead. The *verbose[616]* output of **list[220]** will indicate when no default is used.

Raw data arguments for codec commands

A special set of commands, all of which with "codec" in their name, like **addrcodec[146]**, **shcodec[274]**, **urlcodec[302]**, take raw string data as input, which means that the input line is taken literally: like this the effect of the actual codec is visible without any noise of possible shell quoting rules etc., that is, the user can input one-to-one the desired or questionable data. To gain a level of expansion, the entire command line can be **eval[127]**uated first, for example

```
? vput shcodec res encode /usr/Schönes Wetter/heute.txt
? echo $res
$'/usr/Sch\u00F6nes Wetter/heute.txt'
? shcodec d $res
$'/usr/Sch\u00F6nes Wetter/heute.txt'
? eval shcodec d $res
/usr/Schönes Wetter/heute.txt
```

Filename transformations

Filenames, where expected, and unless documented otherwise, are subject to the following transformations, in sequence:

- If the name is a registered **shortcut**[276] it will be expanded. This step is mostly taken by **folder**[203]s only.
- The name is matched against the following patterns or strings. But for plus *+file* *folder*[427] expansion this step is mostly taken by **folder**[203]s only.

  # (Number sign) is expanded to the previous file.

  % (Percent sign) is replaced by primary system mailbox, which either is the (itself expandable) *inbox*[453] if set, the standardized absolute pathname indicated by MAIL[634] if set, or a built-in compile-time default otherwise. When opening a **folder**[203] the name is checked for being a primary mailbox, first *inbox*, then MAIL[634].

  %user Expands to the primary system mailbox of *user* (and never the value of *inbox*[453], regardless of its actual setting).

  & (Ampersand) is replaced with the secondary mailbox, the MBOX[638].

  +file Refers to a *file* in the *folder*[427] directory (if set).

  %:filespec Expands to the same value as *filespec*, but has special meaning when used with, for example, **folder**[203]: the file will be treated as a primary system mailbox by, among others, the **mbox**[226] and **save**[268] commands, meaning that messages that have been read in the current session will be moved to the MBOX[638] mailbox instead of simply being flagged as read.
- Meta expansions may be applied to the resulting filename, as allowed by the operation and applicable to the resulting access protocol (also see **On URL syntax and credential lookup**[18]). For the file-protocol, a leading tilde ‘~’ character will be replaced by the expansion of HOME[627], except when followed by a valid user name, in which case the home directory of the given user is used instead.

  A shell expansion as if specified in double-quotes (see **Shell-style argument quoting**[24]) may be applied, so that any occurrence of $VARIABLE (or ${VARIABLE}) will be replaced by the expansion of the variable, if possible; **INTERNAL VARIABLES**[31] as well as **ENVIRONMENT**[34] (shell) variables can be accessed through this mechanism.

  Shell pathname wildcard pattern expansions (glob(7)[698]) may be applied as documented. If the fully expanded filename results in multiple pathnames and the command is expecting only one file, an error results.

  In interactive context, in order to allow simple value acceptance (via “ENTER”), arguments will usually be displayed in a properly quoted form, so a file `diet\ is \curd.txt` may be displayed as `'diet\ is \curd.txt'`.

Commands

! *command*

Executes the given SHELL[643] *command*, replacing unescaped exclamation marks with the previously executed command if *bang*[383] is set. **vput**[132] (**Command modifiers**[22]) and error number *!*[351] are supported. A 0 or positive exit status *?*[350] reflects the exit status of the command, negative ones that an error happened before the command was executed, or that the program did not exit cleanly, but maybe due to a signal: the error number is *^ERR*[356]-CHILD, then.

Special cases exist in conjunction with **vput**[132]: a negative exit status occurs if the collected data could not be stored in the given variable, which is a *^ERR*[356]-NOTSUP error that should otherwise not occur. *^ERR*[356]-CANCELED indicates that no temporary file could be created to collect the command output at first glance. In case of catchable out-of-memory situations

ˆERR[356]-NOMEM will occur and it will be tried to store the empty string, just like with all other detected error conditions.

The comment-command causes the entire line to be ignored. This is a normal command which' purpose is to discard its arguments. **Shell-style argument quoting[24]** is required for shell-style comments.

= "dotquery": query the unique number of the current message (the "dot"). With arguments show the resulting list of numbers, separated by the first byte of *ifs[449]*, and followed by the first byte of *if-ws* if non-empty and not identical. If this results in no separation at all a **space** is used. Note neither is "dot" moved (see **dotmove[182]**), nor are **Message states[13]** updated. **vput[132]** and the error number *![351]* are supported.

: No-effect command: does nothing but expanding arguments. (For an otherwise empty input line that invokes the default command **next[241]** this conflicts the *:* message specification documented in **Specifying messages[14]**, but compatibility to the well-known sh(1)[699] **:** command deemed more important than this special case.)

? [Option] Show a brief summary of commands. [Option] Given an optionally abbreviated name the according synopsis is shown instead, try for example '?h', ?hel and ?help, and see how the output changes. To avoid that **commandalias[166]**es are resolved the modifier **\[126]** can be prepended to the argument, but note it must be quoted. A more *verbose[616]* output is supported.

| A synonym for the **pipe[248]** command.

account, **unaccount**

Create, select or list, and delete, respectively, macros that bundle settings for email account creation, covered by a specific **local[130]** scope that extends until another account is activated. When left, also for a regular program **quit[252]** or when the account macro cannot be switched to successully, the hook *on-account-cleanup[494]* is called. There is a special read-only null account for the purpose of changing to global scope.

Without arguments a list of all accounts is shown. With one argument the given one is activated: after the macro body has been evaluated its system *inbox[453]* is opened as via **folder[203]**, a possibly installed *on-mailbox-open[501]* will be run, and *account[366]* will be updated. The two argument form creates a new account macro as via **define[174]**.

Important settings for accounts include *folder[427]*, *from[436]*, *hostname[447]*, *inbox[453]*, *mta[482]*, *password[508]*, *record[537]*, and *user[614]* (**On URL syntax and credential lookup[18]**), as well as things like *smime-sign-cert[569]* (**Signed and encrypted messages with S/MIME[17]**), *tls-config-pairs[601]* (**Encrypted network communication[19]**), and protocol specifics like *imap-auth[677]*, *pop3-auth[518]*, *smtp-config[573]*.

```
account myisp {
  set folder=~/mail inbox=+syste.mbox record=+sent.mbox
  set from='(My Name) myname@myisp.example'
  set mta=smtp://me@smtp.myisp.example
}
```

addrcodec

Perform email address codec transformations on raw-data argument, rather according to email standards (RFC 5322; [v15 behaviour may differ] will furtherly improve). Supports**vput [132]** and the error number *![351]*. The first argument must be one of *[+[+[+]]]e[ncode]*, *d[ecode]*, *s[kin]* and *skinl[ist]*.

Decoding will show how a standard-compliant MUA will display the data. MUAs have varying support for address standards regarding comments, parenthesis, double-quoted strings, or so-called quoted-pairs. [v15 behaviour may differ] S-nail currently does not perform decoding when displaying addresses.

Skinning is identical to decoding but only outputs the plain address, without any string, comment etc. components. Another difference is that it may fail with *![351]* set to *^ERR[356]*-INVAL if decoding fails to find a(n) (valid) email address, in which case the unmodified input will be output again.

`skinlist` first performs a skin operation, and thereafter checks a valid result for whether it is registered via **`mlist`[230]** or **`mlsubscribe`[232]**, eventually reporting the result in *![351]* as *^ERR[356]*-EXIST. (That could become overwritten by an I/O error, though.)

Encoding supports four different modes: the more plus signs, the lesser standard imposed special bytes are converted to so-called quoted-pairs. One plus sign to leave alone parenthesis, two for double quotation marks, three for also leaving reverse solidus alone. The result will be valid if a successful exit status is reported ([v15 behaviour may differ] the current parser fails this assertion for some constructs). [v15 behaviour may differ] Addresses need to be specified in between angle brackets ‘<’, ‘>’ if the construct becomes more difficult, otherwise the current parser will fail; it is not smart enough to guess right.

```
? addrc enc "Hey, you",<diet@exam.ple>\ out\ there
"\"Hey, you\", \\ out\\ there" <diet@exam.ple>
? addrc d "\"Hey, you\", \\ out\\ there" <diet@exam.ple>
"Hey, you", \ out\ there <diet@exam.ple>
? addrc s "\"Hey, you\", \\ out\\ there" <diet@exam.ple>
diet@exam.ple
```

`alias`, **`unalias`**

[Only new quoting rules](a, una) Define or list, and remove, respectively, address aliases, a method of creating personal distribution lists that map single names to none to multiple recipients, to be expanded after **Compose mode[8]** is left (see *metoo[475]*). Without arguments the former shows a list of all known aliases, one per line, with one argument only the expansion of the given one. With two arguments, hyphen-minus ‘-’ being the first, the recursive expansion of the second is shown.

In all other cases the given alias is created or appended to: arguments must be valid alias names or other address types (see **On sending mail, and non-interactive mode[7]**). The later recursive expansion can be prevented by prefixing the desired name with reverse solidus ****. A valid alias name conforms to *mta-aliases[483]* syntax, but follow-up characters can also be the number sign ‘#’, colon ‘:’, commercial at ‘@’, exclamation mark ‘!’, period ‘.’ as well as “any character that has the high bit set”. The dollar sign ‘$’ may be the last character. The number sign ‘#’ may need **Shell-style argument quoting[24]**.

[v15 behaviour may differ] Unfortunately the colon is currently not supported, as it interferes with normal address parsing rules. [v15 behaviour may differ] High bit characters will likely cause warnings at the moment for the same reasons why colon is unsupported; also, in the future locale dependent character set validity checks will be performed.

```
? alias cohorts  bill jkf \mark kridle@ucbcory ~/cohorts.mbox
? alias mark  mark@exam.ple
? set mta-aliases=/etc/aliases
```

alternates, **unalternates**

[Only new quoting rules](alt) Manage a list of alternate addresses or names for the user, members of which will be removed from recipient lists (but see *metoo[475]*). A set of implicit alternates is formed out of `LOGNAME`[633], *from[436]*, *sender[551]* and *reply-to[542]*. *from[436]* will not be used if *sender[551]* is set.

The former command manages the error number *![351]*. Without arguments it shows the current alternates; only then it supports **vput[132]**. Otherwise the given arguments are appended to the alternate list; in *posix[523]* mode they instead replace that list.

answered, **unanswered**

Take a message list and mark each member as (not) having been answered. Dependent on *markanswered[468]* this mark may be set automatically by **reply[261]**. See**Message states[13]**.

bind, **unbind**

[Option][Only new quoting rules] Freely configure key bindings for the M(ailx)-L(ine)-E(ditor) (**On terminal control and line editor[15]**). The latter removes from the given context the given key binding, both of which may be the wildcard ‘*’, so that `unbind * *` removes all bindings of all contexts. Due to initialization order issues built-in sequences cannot become unbound upon program startup, instead see *line-editor-no-defaults[461]*.

Bindings are context-sensitive. This is not true for the shared`base` foundation upon which the other contexts are build. The other contexts are `default` that is used unless otherwise documented, and `compose` that is used in **Compose mode[8]**.

With zero arguments, or with only a context name (may be wildcard ‘*’) the former lists key bindings (*verbose[616]*ly). With two or more arguments a specific binding is shown or (re)established: the first is the context to which it shall apply, the second is a comma-separated list of the “keys” that make up the binding. All further arguments are joined to form the expansion, and cause the binding to be created or updated.

Normally the expansion of a triggered binding is immediately committed as an input line, shall the user instead have an editing option the last byte of the expansion must be a commercial at ‘@’. Reverse solidus cannot be the last character of an expansion. An empty expansion will be rejected.

Bindings are specified as a comma-separated sequence of keys. Each key list entry consists of a sequence of one or more bytes. Byte sequence input is forcefully terminated after *bind-inter-byte-timeout[385]* milliseconds, whereas key sequences can be timed out via *bind-inter-key-timeout[386]*, which allows to share an identical key prefix in between many bindings. Key list entries that start with colon ‘:’ name terminal capabilities: many names are built-in and may be specified either by their `terminfo`(5)[700], or, if existing, `termcap`(5)[701] name, regardless of the actually used terminal control library. But any capability resolvable by the control library or via *termcap[592]* may be used.

```
? bind default a,b echo one
? bind default a,b,c echo two
? bind default ab,c echo notbush city limit!
? bind base $'\E',d mle-snarf-word-fwd # Esc(ape)
? bind base $'\E',$'\c?' mle-snarf-word-bwd # Esc,Delete
? bind default $'\cA',:khome,w 'echo Editable binding@'
? bind default a,b,c,d rm -irf / @  # Also editable
? bind default :kf1 File %
? bind compose :kf1 ~v

? define kb-clear {
```

```
    # Careful to re-bind mle-commit in same line!
    \unbind * *;\
      \bind base $'\n' mle-commit;\
      \bind base $'\c?' mle-del-bwd;\
      \bind base $'\cE' mle-go-end
}
```

Notes. The entire comma-separated list is first parsed (over) as a shell-token with whitespace as the field separator, then parsed and expanded for real separated with comma, therefore whitespace must be properly quoted (**Shell-style argument quoting[24]**)! Using Unicode reverse solidus escape sequences renders bindings defunctional if the locale does not support Unicode (**Character sets[12]**). Unresolvable terminal capabilities render bindings defunctional, too. Adding, deleting or modifying key bindings invalidates the internal prebuilt lookup tree, it will be recreated as necessary: in *verbose[616]*=3 mode the resulting tree will be dumped. It is advisable to use an initial escape or other control character (like `\cA`) for user bindings in order to avoid ambiguities and reduce search time.

The following terminal capability names are built-in and can be used in `terminfo`(5)[702] or (if available) the two-letter `termcap`(5)[703] notation. See the respective manual for a list of capabilities. The program`infocmp`(1) [704] can be used to show all the capabilities of `TERM`[646], its **`-x`** option shows supported extensions.

`kbs` or **`kb`**	Backspace.
`kdch1` or **`kD`**	Delete character.
`kDC` or **`*4`**	— shifted variant.
`kel` or **`kE`**	Clear to end of line.
`kext` or **`@9`**	Exit.
`kich1` or **`kI`**	Insert character.
`kIC` or **`#3`**	— shifted variant.
`khome` or **`kh`**	Home.
`kHOM` or **`#2`**	— shifted variant.
`kend` or **`@7`**	End.
`knp` or **`kN`**	Next page.
`kpp` or **`kP`**	Previous page.
`kcub1` or **`kl`**	Left cursor (with more modifiers: see below).
`kLFT` or **`#4`**	— shifted variant.
`kcuf1` or **`kr`**	Right cursor (ditto).
`kRIT` or **`%i`**	— shifted variant.
`kcud1` or **`kd`**	Down cursor (ditto).
`kDN`	— shifted variant (only terminfo).
`kcuu1` or **`ku`**	Up cursor (ditto).
`kUP`	— shifted variant (only terminfo).
`kf0` or **`k0`**	Function key 0. Add one for each function key up to **`kf9`** and **`k9`**, respectively.
`kf10` or **`k;`**	Function key 10.
`kf11` or **`F1`**	Function key 11. Add one for each function key up to **`kf19`** and **`F9`**, respectively.

Some terminals support key-modifier combination extensions, e.g., `Alt+Shift+xy`. For example, the delete key, **`kdch1`**: in its shifted variant, the name is mutated to **`kDC`**, then a number is appended for the states `Alt` (**`kDC3`**), `Shift+Alt` (**`kDC4`**), `Control` (**`kDC5`**), `Shift+Control` (**`kDC6`**), `Alt+Control` (**`kDC7`**), finally `Shift+Alt+Control` (**`kDC8`**). The same for the left cursor key, **`kcub1`**: **`KLFT`**, **`KLFT3`**, **`KLFT4`**, **`KLFT5`**, **`KLFT6`**, **`KLFT7`**, **`KLFT8`**.

call [Only new quoting rules] Evaluate the given macro as created via **define**[174] (otherwise an *ˆERR*[356]-NOENT error occurs). Calling macros recursively excesses the stack size limit at some point, resulting in a hard program abortion; (if usable) this can be avoided by **xcall**[310]. Both commands support **local**[130] scoping.

call_if
Does not fail for non-existing macros, otherwise identical to **call**[155].

cd Synonym for **chdir**[161].

certsave
[Option] Takes an optional message list and a filename and saves the certificates of the message signatures to the named file in both human-readable and PEM format, for example, to use them for *smime-encrypt-USER@HOST*[566] specializations.

charsetalias, **uncharsetalias**
[Only new quoting rules] Manage alias mappings of **Character sets**[12]. The former command lists all mappings when used without arguments, or only the given one with one; with two arguments, hyphen-minus ‘-’ being the first, the second is instead expanded recursively. In all other cases the given arguments are treated as pairs of source and target character sets, creating new or updating existing mappings. Mappings do not apply to **INTERNAL VARIABLES**[31] like *charset-8bit*[397], and are generally ignored if character set conversion is not available. Mappings are expanded recursively (recursion limit: 8).

chdir [Only new quoting rules](ch) Change the working directory to the given argument or HOME[627]. Synonym for **cd**[157].

collapse, **uncollapse**
Takes a message list and hides all replies to these messages (for **headers**[214]), except for new messages and the “dot”. Also when a message with collapsed replies is displayed, these are automatically uncollapsed. The latter command undoes collapsing. Only applicable to threaded **sort**[282] mode.

colour, **uncolour**
[Option][Only new quoting rules] Manage colour mappings of and for a **Coloured display**[16]. Without arguments the former shows all defined mappings, otherwise a colour type is expected (case-insensitively), one of 256 for 256-colour terminals, ‘8’, ansi or iso for the standard 8-colour ANSI / ISO 6429 colour palette, and ‘1’ ormono for monochrome terminals: these only support (some) font attributes. Without further arguments the list of all defined mappings of the given type is shown.

Otherwise the second argument defines a slot name, the third a (comma-separated list of) colour and font attribute(s), and the optionally supported fourth argument can be used to specify a precondition: if conditioned mappings exist they are tested in (creation) order unless a (case-insensitive) match has been found, and the default mapping (if it exists) will only be chosen as a last resort. Preconditions depend on slots, the following of which exist:

Slot names prefixed with mle- are used for the [Option]al built-in Mailx-Line-Editor **MLE**[93] (see **On terminal control and line editor**[15]), and do not support preconditions.

mle-position The position indicator that is visible when a line cannot be fully displayed on the screen.
mle-prompt *prompt*[525].
mle-error [v15 behaviour may differ] Used for error messages written on standard error only.

Slot names prefixed with `sum-` are used by **`headers`**[214]. Preconditions are `dot` (current message) and `older` for elder messages (only in conjunction with *datefield-markout-older*[411]).

`sum-dotmark` Used for "dotmark" creatable with the *headline*[439] formats '`%>`' or '`%<`'.
`sum-header` For all summary line content except "dotmark" and the threaded **`sort`**[282] tree.
`sum-thread` For the threaded **`sort`**[282] tree creatable with the *headline*[439] format '`%i`'.

Slot names prefixed with `view-` are used when messages are **`type`**[297]d.

`view-from_` Used for so-called `From_` lines, which are MBOX file format specific header lines (also see *mbox-rfc4155*[470]).
`view-header` For header lines. A comma-separated list of headers to which the mapping shall apply may be given as a precondition; if the [Option]al regular expression support is available then it is evaluated as one if it contains **magic regular expression characters**[92].
`view-msginfo` For the introductional message info line.
`view-partinfo` For MIME part info lines.

The following (case-insensitive) colour definitions and font attributes are understood, multiple of which can be specified in a comma-separated list:

`ft=` a font attribute: `bold`, `reverse` or `underline`.

`fg=` foreground colour attribute, in order (numbers 0 - 7) `black`, `red`, `green`, `brown`, `blue`, `magenta`, `cyan` or `white`. To specify a 256-colour mode a decimal number colour specification in the range 0 to 255, inclusive, is supported, and interpreted as follows:

0 - 7	the standard ISO 6429 colours, as above.
8 - 15	high intensity variants of the standard colours.
16 - 231	216 colours in tuples of 6.
232 - 255	grayscale from black to white in 24 steps.

```
#!/bin/sh -
fg() { printf "\033[38;5;${1}m($1)"; }
bg() { printf "\033[48;5;${1}m($1)"; }
i=0
while [ $i -lt 256 ]; do fg $i; i=$(($i + 1)); done
printf "\033[0m\n"
i=0
while [ $i -lt 256 ]; do bg $i; i=$(($i + 1)); done
printf "\033[0m\n"
```

`bg=` background colour attribute (same values as **`fg=`**).

The command **`uncolour`**[165] will remove for the given colour type (or all with '`*`') the given mapping; if the optional precondition argument is given only the exact tuple of mapping and precondition is removed. The special name '`*`' will remove all mappings (no precondition allowed), thus `uncolour * *` will remove all established mappings.

`commandalias`, **`uncommandalias`**

[Only new quoting rules] Define or list, and remove, respectively, command aliases. Aliases can always be used in place of normal command names, and any given arguments are joined onto the alias expansion. Without arguments the former shows the list of all aliases and their expansion, with one argument only the given one.

With two or more arguments an alias is defined or updated: the first is the name, the remains can be just about anything. An alias may expand to another alias, but same-name expansion is prevented (after first level), and there is a recursion depth limit. Explicit expansion prevention is available via reverse solidus **\[126]** (**Command modifiers[22]**).

```
? commandalias xx
s-nail: 'commandalias': no such alias: xx
? commandalias xx echo hello,
? commandalias xx
commandalias xx 'echo hello,'
? xx world
hello, world
? commandalias q q; commandalias x q; help x
x -> q -> q (quit): Exit session [.]
```

Copy (C) Similar to **copy[169]**, but copy messages to a file named after the local part of the sender of the first message instead of taking a filename argument; *outfolder[493]* is inspected to decide on the actual storage location.

copy (c) Copy messages to the named file and do not mark them as being saved; otherwise identical to **save[268]**.

csop [Option][Only new quoting rules] Multiplexer that provides string operations on 8-bit US-ASCII bytes without a notion of **Character sets[12]**. For file operations refer to **fop[208]**, for numeric and other operations to **vexpr[306]**. **vput[132]** (**Command modifiers[22]**) is supported. The error result is '-1' for usage errors and numeric results, the empty string otherwise; missing data errors, as for unsuccessful searches, set *![351]* to *^ERR[356]*-NODATA. When the modifier suffix question mark '?' is supported, a case-insensitive mode is supported; the keyword `case` is optional so that `find?` and `find?case` are identical.

length Query the argument length.

hash, **hash32** Calculate a (32-bit) hash of the argument. '?' modifier is supported. These use Chris Torek's hash algorithm, the resulting hash value is bit mixed as shown by Bret Mulvey.

find Search for the second in the first argument, show 0-based offset upon success. '?' modifier is supported.

substring Create a substring of the first argument. The optional second argument could specify a 0-based starting offset, a negative one counts from the end. The optional third argument defines the desired result length, a negative one leaves off the given number of bytes at the end of the original string; by default the entire string is used. This operation tries to work around faulty arguments (**set[271]** *verbose[616]* for error logs), but reports them via the error number *![351]* as *^ERR[356]*-OVERFLOW.

trim Trim away whitespace from both ends of the argument.

trim-front Trim away whitespace from the begin of the argument.

trim-end Trim away whitespace from the end of the argument.

cwd Show the current working directory name as reported by `getcwd`(3)[705]. Supports **vput[132]** (**Command modifiers[22]**). The return status is tracked via *?[350]*.

Decrypt, **decrypt**
[Option] For unencrypted messages identical to **Copy[168]**; Encrypted messages are first decrypted, if possible, and then copied.

define, **undefine**

The latter deletes the given macro, '∗' discards all existing ones. Deletion, including self-deletion, is always possible. The former lists all or only the given macro(s), with two arguments it defines or replaces a macro. Macro names are free-form but must form a single token (**Shell-style argument quoting**[24]).

Macros can be evaluated by **call**[155], **call_if**[156] or **xcall**[310], optionally with **local**[130] scoping. Inside a running macro positional parameters can be accessed via ∗*[361]*, *@[362]*, *#[363]* and *1[365]* as well as any positive unsigned decimal number less than or equal to *#[363]*; they can be **shift**[278]ed away, and become exchanged via **vpospar**[307]. **vexpr**[306] is a numeric computation helper. **return**[266] can be used to explicitly give back control to the caller.

```
? unset prompt
define name {
  command1
  ...
  commandN
}

define exmac {
  echo Parameter 1 of ${#} is ${1}, all: ${∗} / ${@}
  return 1000 0
}

set prompt='? '
? call exmac Hello macro exmac!
? echo ${?}/${!}/${^ERRNAME}
```

delete, **undelete**

(d, u) Mark the **Message list arguments**[26] (not) `deleted`, one of the **Message states**[13]. If *autoprint[381]* is set, the new "dot" or the last message restored, respectively, is automatically **type**[297]d; also see **dp**[180], **dt**[181].

digmsg [Only new quoting rules] An object that digs (information out of) a message can be **create**d for the given message number; in **Compose mode**[8] the hyphen-minus '-' instead uses the draft message. The optional third argument hyphen-minus redirects object output to standard output instead of making it consumable by **readall**[255], **read**[253] or **readsh**[254]. Without redirection output must always be consumed (until end of file, EOF)! Objects should be **remove**d to prevent resource exhaustion; it happens automatically when their **folder**[203] (or compose mode) is left.

In all other use cases the first argument is the object's message number, the remaining arguments are interpreted as for **~^**[322] (**COMMAND ESCAPES**[30]):

```
? vput = msgno; digmsg create $msgno
? digmsg $msgno header list;   readall x;   echon $x
210 Subject From To Message-ID References In-Reply-To
? digmsg $msgno header show Subject;readall x;echon $x
212 Subject
'Hello, world'

? digmsg remove $msgno
```

discard
(di) Identical to **ignore[219]**. Superseded by the multiplexer **headerpick[212]**.

dp, **dt** Delete given messages and thereafter **type[297]** a possible new “dot” regardless of *autoprint[381]*.

dotmove
Move the “dot” up ‘+’ or down ‘-’ by one message if possible, then update**headers [214]**.

draft, **undraft**
Mark each given message as (not) being a draft (**Message states[13]**).

echo [Only new quoting rules](ec) Print arguments after expanding their **Shell-style argument quoting[24]**, then a newline. **vput[132]** (**Command modifiers[22]**) is supported, the error number *![351]* is managed: with **vput[132]** the result length is returned on success, ‘-1’ on error. Remarks: in BSD Mail this command also performed**Filename transf ormations[28]**, which is standard incompatible and hard to handle because quoting transformation patterns is not possible; the subcommand **file-expand** of **vexpr[306]** can be used to expand filenames.

echoerr
[Only new quoting rules] Like **echo[185]**, but write to standard error and prefix lines by *log-prefix[462]*. Also see**echoerrn [188]**. [Option] In interactive sessions and if **vput[132]** was not used message duplicates are stored in the **errors[194]** queue.

echon [Only new quoting rules] Like **echo[185]**, but do not append a newline.

echoerrn
[Only new quoting rules] like **echoerr[186]**, but do not append a newline.

edit (e) Successively open EDITOR[626] on each entry of the given message list. Modified contents are discarded unless *writebackedited[623]* is set, the mailbox can be written to, and the editor returned success. **visual[308]** can be used instead for a more display oriented editor.

elif Part of the **if[218]** (see there), **elif[190]**, **else[191]**, **endif[192]** conditional. An else-if block is evaluated if former blocks were not and its condition arguments are true.

else (el) Part of the **if[218]** (see there), **elif[190]**, **else[191]**, **endif[192]** conditional. An else block is evaluated only if none of the condition blocks matched.

endif (en) Marks the end of an **if[218]** (see there), **elif[190]**, **else[191]**, **endif[192]** conditional execution block.

environ
[Only new quoting rules] The process **ENVIRONMENT[34]** and **INTERNAL VARIABLES[31]** may be linked together as long as SHELL[643] **variable name rules[133]** are honoured. The subcommands **set** and **unset** do this inclusive value updates, whereas **link** and **unlink** only manage link state: if an according internal variable already exists (**if[218]**), the environment is synchronized immediately, if it only exists in the environment instead an internal variable is created, otherwise an error occurs. Changes are scopeable via **local[130]**; **local[130]** **set[271]** same-name (still) shadows a linked environment variable. The subcommand**lookup** looks up the given variable in the process environment only; it supports **vput[132]** and sets *![351]* to *^ERR[356]*-NOENT upon lookup failure.

```
# With *errexit*, do not exit if PERL5LIB does not exist
? ignerr environ link PERL5LIB
? if $? -ne 0
?   echoerr PERL5LIB not in environment, using ~/.perl5
?   environ set PERL5LIB=~/.perl5
```

```
? endif
? environ set PERL5LIB=$PERL5LIB:/home/shared/perl5
? vput environ storvar lookup PERL5LIB; echo $storvar
```

errors [Option] Manages the queue of error messages. Error messages sometimes fly by and scroll off the screen fast; In interactive sessions they are duplicated into this queue, more so with *verbose[616]*. *show* or no argument will display and clear the queue, *clear* will only clear it. As the queue becomes filled with *errors-limit[420]* entries the eldest entries are being dropped. There are also *ˆERRQUEUE-COUNT[359]* and *ˆERRQUEUE-EXISTS[360]*.

exit (ex or x) Exit without updating the active mailbox, skip any saving of messages in the **secondary mailbox[137]** MBOX[638], do not save the line editor *history-file[442]*, among others. *on-account-cleanup[494]* is evaluated when set. The optional status number argument is passed through to exit(3)[706]. [v15 behaviour may differ] For now it can happen that the given status will be overwritten, later this will only occur if a later error needs to be reported onto an otherwise success indicating status.

File (Fi) Like **folder[203]**, but open the mailbox read-only.

file (fi) See **folder[203]**.

filetype, **unfiletype**

[Only new quoting rules] Define, list, and remove, file handler hooks. These provide (shell) commands that enable loading and saving MBOX files from and to files with registered file extensions, as shown and described for **folder[203]**. The extensions are used case-insensitively (US-ASCII), yet the auto-completion feature of for example **folder[203]** will only work case-sensitively. An intermediate temporary file will be used to store the expanded data. The latter command removes hooks for the given extensions, asterisk ‘*’ will remove all existing handlers.

When used without arguments the former shows a list of all types, with one argument the expansion of the given one. Otherwise three arguments are expected: the first specifies the file extension, the second and third define load- and save commands; both must read from standard input and write to standard output. Changing hooks will not affect already opened mailboxes ([v15 behaviour may differ] except below). [v15 behaviour may differ] For now too much work is done, and files are oftened read in twice where once would be sufficient: this can cause problems if a filetype is changed while such a file is opened; this was already so with the built-in support of .gz etc. in Heirloom, and will vanish in v15. [v15 behaviour may differ] For now all handler strings are passed to the SHELL[643] for evaluation purposes; in the future a ‘!’ prefix to load and save commands may mean to bypass this shell instance: placing a leading space will avoid any possible misinterpretations.

```
? filetype bz2 'bzip2 -dc' 'bzip2 -zc' \
    gz 'gzip -dc' 'gzip -c'  xz 'xz -dc' 'xz -zc' \
    zst 'zstd -dc' 'zstd -19 -zc' \
    zst.pgp 'gpg -d | zstd -dc' 'zstd -19 -zc | gpg -e'
? set record=+sent.zst.pgp
```

flag, **unflag**

Take message lists and mark the messages as being flagged, or not being flagged, respectively, for urgent/special attention. See the section **Message states[13]**.

Folder (Fold) Like **folder[203]**, but open the mailbox read-only.

folder (fold) Open a new, or show informations about the current mailbox. When opening the current mailbox is synchronized and closed first. *name* under goes **Filename transformations[28]**, and URL (**On URL syntax and credential lookup[18]**) mailbox-type:// prefixes are understood, as in mbox:///tmp/somefolder, or, in general:

```
protocol://[user[:password]@]host[:port][/path]
```

Without mailbox type fixation opening none-existing **folders[205]** uses the *newfolders[491]* protocol. After the mailbox has been opened*mailbox-r esolved[465]* and *mailbox-display[464]* are updated, a set *on-mailbox-open[501]* is evaluated, and optionally a summary of **headers[214]** is displayed if *header[438]* is set.

For the protocols *mbox* and *file* (MBOX database), as well as *eml* (electronic mail message [v15 behaviour may differ] read-only) the list of all registered **filetype[198]**s is traversed to check whether hooks shall be used to load (and save) data from (and to) the given *name* (except for the "-" special case as documented for **-f [68]**). Changing hooks will not affect already opened mailboxes.

[Obsolete] For historical reasons **filetype[198]**s provide limited (case-sensitive) auto-completion capabilities. For example `mbox.gz` will be found for `? file mbox`, provided that corresponding handlers are installed. It will neither find `mbox.GZ` nor `mbox.Gz` however, but an explicit `? file mbox.GZ` will find and use the handler for '`gz`'. This will vanish in v15!

EML files consist of only one mail message, [v15 behaviour may differ] and can only be opened read-only. When reading MBOX files tolerant POSIX rules are used by default. Invalid message boundaries that can be found quite often in historic MBOX files will be complained about (even more with *debug[412]*): in this case the method described for *mbox-rfc4155[470]* may be used to create a valid MBOX database.

During file operations MBOX databases and EML files are protected by file-region locks (`fcntl`(2)[707]). [Option] Unless*dotloc k-disable[415]*d an MBOX *inbox[453]* (`MAIL`[634]) and any **primary system mailbox[136]** are additionally protected by so-called dotlock files, the traditional way of mail spool file locking: for any file '`x`' a lock file`x.lock` will be created dur - ing the synchronization, in the same directory and with the same user and group identities as the file of interest — as necessary created by an external privileged dotlock helper. Also see**F AQ[43]**: **Howto handle stale dotlock files[49]**.

[Option] If no protocol has been fixated, and *name* refers to a directory with the subdirectories `tmp`, `new` and `cur`, then it is treated as a "Maildir" folder. The maildir format stores each message in its own file, and has been designed so that file locking is not necessary when reading or writing files.

[Option] Network mail protocol resources may be addressed, securely via **Encrypted network communication[19]** if so supported: *pop3* (POP3) and *pop3s* (POP3 via TLS), see *pop3-auth[518]*, as well as *imap* and *imaps* (IMAP via TLS), see **IMAP CLIENT[50]**. For IMAP the *[/path]* URL part defaults to `INBOX`. All network traffic may be proxied over a SOCKS5 server via *socks-proxy[578]*. Network communication socket timeouts are configurable via *socket-connect-timeout[577]*.

folders
: Lists the names of all folders below the given argument or *folder[427]*. For file-based protocols `LISTER`[632] will be used for display purposes.

Followup, **followup**
: (Compose mode)(F,fo) Similar to **Reply[259]** aka **reply[261]**, but save the message in a file named after the local part of the first recipient's address (like **-F[67]**), overwriting *record[537]*, but honouring *outfolder[493]*. Also see**Copy [168]** and **Save[267]**.

fop
: [Option][Only new quoting rules] A multiplexer command for file operations. For C-style byte string operations refer to **csop[170]**, for numeric and other operations to **vexpr[306]**. The first argument defines the number, type, and meaning of the remaining ones. Unless otherwise noted subcommands expect file descriptors opened via **flock**, **lflock**, **lock**, **llock**,

open; standard input and output are accepted, regardless of their validity for the operation. Supports **vput[132]** (see **Command modifiers[22]**). Except when otherwise noted errors will be reported via *![351]*, and the error result is the empty string. Filename arguments undergo **Filename transformations[28]**.

close Close the given file descriptor. Standard descriptors cannot be closed.

expand Only perform name transformations.

ftruncate Truncate the given file descriptor to its current file position (also see **rewind**).

flock, **lflock**, **lock**, **llock** Open the given filename, then apply a shared read (‘`R`, `r`’), or an exclusive read and (appending) write (‘`W`, `w`, `A`, `a`’) lock according to mode argument (second); uppercase variants log retries to gain the lock, writer ones create the file first as necessary, they will truncate the file to zero size first if a ‘`0`’ is part of the mode; further operation-success-echoes on the open descriptor can be suppressed by adding a circumflex ‘`^`’ (for example `fop lock ./a-file.txt A0^`). On success the descriptor number is printed, and must be **close**d again explicitly. **llock** differs by not following symbolic links.

[Option](*features[425]* includes `,+flock,`) Different to the `fcntl`(2)[708] based locks **flock** uses the system call `flock`(2)[709], and creates locks which are inherited by child processes. (**lflock** does not follow symbolic links.) Giving a third argument turns these to one-shots that automatically close opened descriptors: it is passed to a newly spawned `SHELL`[643] that inherits the descriptor as standard input for shared locks, as standard input and output otherwise; If the shell could be started its exit status code is returned, and the normal result is suppressed but for **vput[132]**.

mkdir Create a new directory. The optional second argument is a boolean that denotes whether parent directories shall be created as necessary.

mktmp Create a temporary file and output its name. The optional first (and non-empty) argument will be used as a filename suffix, the (optional) second a target directory to be used instead of `TMPDIR`[647].

open Open the given file with the mode given second, either in read-only (‘`r`’), or read- and (appending) write (‘`W`, `w`, `A`, `a`’) mode; writer ones create the file first as necessary, uppercase versions fail if the file already exists, otherwise, the file will be truncated to zero size first if a ‘`0`’ is part of the mode; further operation-success-echoes on the open descriptor can be suppressed by adding a circumflex ‘`^`’ (for example `fop open ./a-file.txt A0^`). On success the descriptor number is printed, and must be **close**d again explicitly.

pass Takes two file descriptor arguments to be passed as standard I/O to the `SHELL`[643] command given third; beside the usual numbered ones the hyphen-minus ‘`-`’ for passing through the according standard I/O, or commercial at ‘`@`’ for`/dev/null` [657]; passing standard descriptors via number is not supported. If the shell could be started its exit status code is returned, the normal result is suppressed but for **vput.**

rename `rename`(2)[710]. **Remarks:** the first argument is the destination, the second the source.

rewind `rewind`(3)[711] the file descriptor with the given number.

rm `unlink`(2)[712].

rmdir `rmdir`(2)[713].

stat, **lstat** stat(2)[714] the given file and output values such that `vput fop x stat FILE; eval set $x` creates accessible v ariables. **lstat** differs by not following symbolic links when opening a file. The time fields are named `st_Xtime` for the second, and `st_Xtimensec` for the nanosecond (maybe 0), where '`X`' is one of a(ccess), c(hange) and m(odification). `st_type` uses solidus '`/`' to denote directories, commercial at '`@`' for links, number sign '`#`' for block devices, percent sign '`%`' for for character devices, vertical bar '|' for FIFOs, equal sign '`=`' for sockets, and the period '`.`' for the rest.

touch, **ltouch** Update file times, creating the file first as necessary. **ltouch** differs by not following symbolic links when creating the file.

Forward

(Compose mode) Similar to **forward[210]**, but save the message in a file named after the local part of the first recipient's address, overwriting *record[537]*, but honouring *outfolder[493]*.

forward

(Compose mode) Takes a message list and the address of a recipient, subject to *fullnames[437]*, to whom the messages shall be sent. The text of the original message is included in the new one, enclosed by the values of *forward-inject-head[434]* and *forward-inject-tail[435]*. *content-description-forwarded-message[404]* is inspected. The list of included headers can be filtered with the `forward` slot of **headerpick[212]**. Only the first part of a MIME multipart message is included but for *forward-as-attachment[433]*.

This may generate the errors *ˆERR[356]*-DESTADDRREQ if no recipient has been specified, or was rejected by *expandaddr[422]* policy, *ˆERR[356]*-IO if an I/O error occurs, *ˆERR[356]*-NOTSUP if a necessary character set conversion fails, and *ˆERR[356]*-INVAL for other errors. It can also fail with errors of **Specifying messages[14]**. Any error stops processing of further messages.

from (f) Takes a message list, and displays a summary as via **headers[214]**, making the first (the last with *showlast[553]*) message of the result the new "dot". An alias of this command is **search[269]**. Also see**Specifying messages [14]**.

headerpick, **unheaderpick**

[Only new quoting rules] Multiplexer that manages header field selection. The latter always takes three or more arguments and removes from the given context and the given type the given headers, all headers with '`*`'. Without arguments the former shows established settings of all contexts. Otherwise the first argument is the context, one of (case-insensitive) `type` for display purposes (for example **type[297]**), `save` for selecting which headers shall be stored persistently (**save[268]**, **copy[169]**, **move[235]**, **decrypt[173]**; *note:* ignoring MIME related etc. headers destroys message usability), `forward` for **forward[210]**ed (except with *forward-as-attachment[433]*), messages, and `top` for defining a user-defined list of fields for **top[294]**.

With only a context the current settings are shown. A restriction type may be given second, (a case-insensitive prefix of) `retain` for positive or `ignore` for negative filtering, inspected only if no positive one exists. Again, without further arguments the current settings are shown. Further arguments specify header fields, [Option]ally specified as regular expressions, to be added to the given type. The special wildcard field (asterisk, '`*`') will establish a (fast) shorthand setting which covers all fields.

headerpick supports creation and usage of user specified contexts. This mode is selected with (a case-insensitive prefix of) the "context names" **create**, **remove**, **assign**, **join**; the real name is given next, it is case-insensitive (US-ASCII), must not shadow a built-in name, and has to adhere to **variable name rules[133]**. Trying to create a context a second time is an error. The

name of another, constant template context is given third to **assign** and **join**, whereas the former clears the target context first, the latter does not; in case of errors both clear the target context.

headers
(h) Show the current group of headers, the size of which depends on *screen[547]* in interactive mode, and the format of which can be defined with *headline[439]*. Without a message list argument the group surrounding the “dot” is shown, otherwise the first (the last with *showlast[553]*) message becomes the new “dot”.

help (hel) A synonym for **?[142]**.

history
[Option] Without arguments or when given **show** all history entries are shown (possibly more *verbose[616]*ely). **load** replaces the history with the content of *history-file[442]*, and **save** dumps the history to said file, replacing former content; **clear** deletes all entries. The argument can also be a signed decimal *NUMBER*, the respective history entry is then evaluated; a negative number describes an offset to the current command so that ‘-1’ selects the last command, the history top. Last **delete** deletes the given entries (*:NUMBER:*). Also see**On terminal contr ol and line editor[15]**.

hold (ho, also **preserve[249]**) Marks the given message list for saving in the user’s system *inbox[453]* instead of in the **secondary mailbox[137]** MBOX[638]. Does not override the **delete[176]** command. S-nail deviates from the POSIX standard, because **next[241]** issued after **hold[217]** displays the following message, not the current one.

if [Only new quoting rules](i) Part of the **if**, **elif[190]**, **else[191]**, **endif[192]** conditional execution: if the given expression(s) evaluate to true the encapsulated block is executed. All arguments (values, operators, metacharacters for groups, AND-OR lists etc.) must be separate tokens (**Shell-style argument quoting[24]**). The expressions are parsed much like the [[..]] construct of some sh(1)[715]ells; quoting literals that equal syntax elements, especially ‘[’ and unary operators, avoids syntax ambiguities; quoting variable-only arguments is only needed if they may expand to the empty string. Syntax errors cause blocks to be skipped (as no-ops) until **endif[192]**. **Remarks:** in a whiteout the modifier **eval[127]** does not work, even for the conditional command that toggles that state.

```
if expression
  [commands ...]
eli[f]
  [commands ...]
el[se]
  [commands ...]
en[dif]

set i=!
if r && $i == !; echo yes; else; echo no; end

set i=-n
if -n == && ’-n’ == -n; echo yes; end
yes
if -n == && \-n == -n; echo yes; end
yes
if -n == && $i == -n; echo yes; end
yes
if -n == && -n == -n; echo yes; end
..this is an error
```

The only portable operators are **r** (receive mode) and **s** (send mode), all others are extensions. This includes case-insensitivity, and the (partially) spelled-out **receive** and **send**. Further no-argument operators are **t** (any word beginning with 't' not equal to`true`) that matches interactive terminal sessions (running attached to a terminal, and none of the "quickrun" command line options **-e**[66], **-H**[69] and **-L**[72] were used), as well as any boolean value (see **INTERNAL VARIABLES**[31] for textual boolean representations). Non-argument operators are detected as a last ressort.

Unary variable test operators are **-N** and **-Z** which test whether the given variable exists, or not, so that `-N editalong` is true when *editalong*[416] is set, whereas `-Z editalong` is if it is not. **-n** and **-z** test whether the length of the given string is nonzero or zero.

Integer binary operators interpret arguments as integral signed 64-bit numbers, and compare them arithmetically. Invalid numbers are errors, unset variables and the empty string are 0. Operators may be suffixed with the question mark **?** modifier to enforce `saturated` operation mode where numbers linger at the minimum or maximum value instead of causing an error, the keyword is optional, `-lt?`, `-lt?satu` and `-lt?saturated` are identical. The operators are **-lt** (less than), **-le** (less than or equal to), **-eq** (equal), **-ne** (not equal), **-ge** (greater than or equal to), and **-gt** (greater than), for example `if 3 -gt 2;echo yes;end`.

8-bit US-ASCII byte and regular expression binary operators interpret arguments textually. Unset variables expand to the empty string. Operators can be suffixed with the question mark **?** modifier to choose a `case-insensitive` operation mode, the keyword is optional, `==?`, `==?case` and `==?case-insensitive` are identical. Available 8-bit US-ASCII byte binary operators are **<** (less than), **<=** (less than or equal to), **==** (equal), **!=** (not equal), **>=** (greater than or equal to), **>** (greater than), as well as **=%** (is substring of) and **!%** (is not substring of), for example `if Water =% at;echo yes;end`.

[Option] Regular expression binary operators work in the active locale (see **Character sets**[12]). **=~** and **!~** interpret the right hand side argument as an extended regular expression. Match groups (in parenthesis, see `re_format`(7)[716] or `regex`(7)[717], dependent on host system) can be accessed in **local**[130]-most scope via the *^*[352] multiplexer sub-variables *^#*[353], *^0*[354], *^1*[355], for example `if knoedel =~? .*(OEDE).*; echo yes, $^1 is in $^0; end`.

Unary file operators test an aspect of the given filename argument. **Filename transformations**[28] are not performed. Unless noted the same `saturated` question mark **?** modifier like for integer binary operators may be given to ignore the argument and (re)use the file status cache that persists until expression evaluation ends. A non-existing cache is treated as a non-existing file. Without modifier the cache is updated. The operators are **-b** (file exists and is block special), **-c** (exists, character special), **-d** (exists, directory), **-e** (exists), **-f** (exists, regular file), **-g** (exists, set-group-ID), **-k** (exists, sticky bit set), **-L** (exists, symbolic link), **-O** (exists, owned by effective user ID), **-p** (exists, named pipe), **-r** (exists, user can read, no modifier), **-S** (exists, socket), **-s** (exists, size greater than zero), **-t** (file descriptor number argument is open on a terminal, no modifier), **-u** (exists, set-user-ID), **-w** (exists, user can write, no modifier), **-x** (exists, user can execute / search, no modifier). All operators resolve symbolic links except **-L**. For example `if -f /etc/motd && -s? '' ; echo yes; end`.

Binary file operators interpret the left and right hand side arguments as filenames. **Filename transformations**[28] are not performed. The operators are **-ef** (both files refer to same device and inode), **-nt** (left hand modification is newer, or it exists and right hand does not), **-ot** (left hand modification is older, or it does not exist and right hand side does). All operators resolve symbolic links. For example `if /etc/passwd -nt /etc/shadow; echo yes; end`.

Expressions can be joined via AND-OR lists (where AND is **&&** and OR is **||**), which have equal precedence and left associativity. Further groups can be created via interlockable pairs of brackets **[...]**, themselve joinable with AND-OR lists. The unary operator **!** reverses the result of individual expressions or entire groups.

```
if -N debug;echo *debug* set;else;echo not;endif
if "$ttycharset" == UTF-8 || "$ttycharset" ==?cas UTF8
  echo ttycharset is UTF-8, the former case-sensitive!
endif
set t1=one t2=one
if ${t1} == ${t2}
  echo The non-empty variables are byte-wise equal
endif
if $features =% ,+regex, && "$TERM" =~?case ^xterm.*
  echo ..in an X terminal
endif
if [ [ true ] && [ [ "${t1}" != '' ] || \
    [ "$t2" != '' ] ] ]
  echo Noisy, noisy
endif
if true && [ -n "$t1" || -n "${t2}" ]
  echo Left associativity as known from the shell
endif
```

ignore (ig) Superseded by the multiplexer **headerpick**[212].

list Shows all built-in commands in a lookup order that does not always correlate to the alphabetical one: names can be abbreviated, and POSIX standardized some abbreviations. [Option] In *verbose*[616] mode argument types and documentation strings will be shown, and the set of command flags will show up:

- `'local'` supports the modifier **local**[130].
- `'vput'` supports the modifier **vput**[132].
- `*!*` the error number is tracked in *!*[351].
- `needs-box` whether the command needs an active mailbox, a **folder**[203].
- `ok:` indicators whether command is . . .
  - `batch/interactive` usable in interactive or batch mode (**-#**[90]).
  - `send-mode` usable in send mode.
  - `subprocess` allowed when running in a subprocess, for example via *on-compose-splice*[498].
- `not ok:` indicators whether command is not . . .
  - `compose mode` available in **Compose mode**[8].
  - `startup` available during program startup, for example in **Resource files**[36] (usually given as “nay”).
- `gabby` The command produces *history-gabby*[443] **history**[216] entries.

localopts
[Obsolete] Please just use the **local**[130] modifier, as in `local set`, `local environ set` or `local call`.

Lfollowup, **Lreply**
(Compose mode) Reply to messages that come in via known (**mlist**[230]) or subscribed (**mlsubscribe**[232]) mailing lists, or pretend to do so (see **Mailing lists**[11]): on top of the usual **followup**[207] and **reply**[261] functionality this will actively resort and even

remove message recipients in order to generate a message that is supposed to be sent to a mailing list. It will also implicitly generate a`Mail-Followup-To:` header if that seems useful, re gardless of *followup-to[429]*. For more documentation refer to **On sending mail, and non-interactive mode[7]**.

This may generate the errors *ˆERR[356]*-DESTADDRREQ if no recipient has been specified, *ˆERR[356]*-PERM if some recipients where rejected by *expandaddr[422]*, *ˆERR[356]*-IO if an I/O error occurs, *ˆERR[356]*-NOTSUP if a necessary character set conversion fails, and *ˆERR[356]*-INVAL for other errors. It can also fail with errors of **Specifying messages[14]**. Any error stops processing of further messages.

`Mail` (Compose mode) Similar to **`mail`[224]**, but save the message in a file named after the local part of the first recipient's address, overwriting *record[537]*, but honouring *outfolder[493]*.

`mail` (Compose mode)(m) Send mail to given (or asked for) list of recipients. Unless *fullnames[437]* is set recipient addresses are skinned, and see **On sending mail, and non-interactive mode[7]**.

This may generate the errors *ˆERR[356]*-DESTADDRREQ if no recipient has been specified, *ˆERR[356]*-PERM if some recipients where rejected by *expandaddr[422]*, *ˆERR[356]*-NOTSUP if multiple messages have been specified, *ˆERR[356]*-IO if an I/O error occurs, *ˆERR[356]*-NOTSUP if a necessary character set conversion fails, and *ˆERR[356]*-INVAL for other errors. It can also fail with errors of **Specifying messages[14]**.

`mailcap`

[Option] Without arguments, or with only *`show`*, the content of **The Mailcap files[38]** cache is shown, (re-)initializing it as necessary. If it is *`load`* the cache will only be (re-)initialized, and *`clear`* will remove its contents. Only one attempt to load the files is made, *`clear`* has to be used for a retry. The load and parse step reacts upon*verbose [616]*.

`mbox` (mb) The given message list is to be sent to the **secondary mailbox[137]** `MBOX[638]` when the folder is left; this is default except with *hold[446]*. [v15 behaviour may differ] Can only be used in a **primary system mailbox[136]**.

`mimetype`, **`unmimetype`**

[Only new quoting rules] Without arguments the MIME type cache is shown, maybe more *verbose[616]*. Otherwise arguments are joined, and interpreted as shown in **The mime.types files[37]** (also see **HTML mail and MIME attachments[10]**), and the result will be prepended (last in first out) to the cache. In any event MIME type sources are loaded first as necessary – *mimetypes-load-control[481]* fine-tunes loading.

The latter command deletes all entries of the given MIME types, asterisk '`*`' discards all defined MIME types, just like `reset`, but that also reenables cache initialization via *mimetypes-load-control[481]*.

`mimeview`

[v15 behaviour may differ] Only available in interactive mode, this command allows execution of external MIME type handlers which do not integrate into the normal **`type`[297]** output (**HTML mail and MIME attachments[10]**). ([v15 behaviour may differ] No syntax to directly address parts, this restriction may vanish.) The user will be asked for each supported part in turn whether the registered handler shall be used to display the part.

`mlist`, **`unmlist`**

[Only new quoting rules] Manage the list of known **Mailing lists[11]**; subscriptions are controlled via **`mlsubscribe`[232]**. The former shows all currently known lists if used without arguments, otherwise the given specifications become known. [Option] Many addresses can be matched with a single one by using **magic regular expression characters[92]** (matched sequentially via linked lists instead of dictionaries, though). The latter deletes all given specifications, or

all with asterisk ‘*’.

mlsubscribe, **unmlsubscribe**
Building upon the command pair **mlist[230]**, **unmlist[231]**, but only managing the subscription attribute of mailing lists. (The former also creates yet unknown mailing lists.)

Move Similar to **move[235]**, but move messages to a file named after the local part of the sender of the first message instead of taking a filename argument; *outfolder[493]* is inspected to decide on the actual storage location.

move Acts like **copy[169]**, but marks the messages for deletion if they were transferred successfully.

More Like **more[237]**, but display all headers and MIME parts. Identical to **Page[245]**.

more Show the given messages in PAGER[640], even in non-interactive mode (only **if[218]** terminal). Identical to**page [246]**.

mtaaliases
[Option] Without an argument or with *show* the *mta-aliases[483]* cache is initialized if necessary and then shown; the output is normalized and properly quoted and fitted to COLUMNS[624]; it is usable as an aliases(5)[718] input file. If the argument is *clear* the cache is removed, whereas *load* will also reinitialize it. Otherwise the expansion of the given MTA alias is shown in one line. With two arguments, hyphen-minus ‘-’ being the first, show the recursive expansion of the second.

netrc [Option] Without arguments or with *show* the ~/.netrc[656] cache is shown, (re-)initializing it (as necessary). If the argument is *load* then the cache will (only) be (re-)initialized, *clear* removes its contents. Loading and parsing can be made more*verbose [616]*. *lookup* will query the cache for the URL given as the second argument ([USER@]HOST). See*netr c-lookup[489]*, *netrc-pipe[490]* and the section **On URL syntax and credential lookup[18]**; the section **The .netrc file[39]** documents the file format in detail.

newmail
Checks for new mail in the current **folder[203]**, and shows a message if so; if *header[438]* is set the **headers[214]** of new messages are shown. This command is not available for all mailbox types.

next (n) Go to the next (the first matching) message and **type[297]** it on success. This is the default command for an empty interactive input line.

New Same as **Unread[299]**.

new Same as **unread[300]**.

noop If the current **folder[203]** is accessed via a network connection, a no-op(eration) command is sent over the wire (for keep-alive purposes).

Page Like **page[246]**, but display all headers and MIME parts. Identical to **More[236]**.

page Show the given messages in PAGER[640], even in non-interactive mode (only **if[218]** terminal). Identical to**more [237]**.

Pipe Like **pipe[248]**, but pipe all headers and MIME parts.

pipe (pi) Takes an optional message list and a shell command (defaults to *cmd[399]*), and pipes the messages in visual form (as via **page[246]**) through the command invoked by the SHELL[643] If the *page[507]* variable is set, every message is followed by a formfeed character.

preserve
(pre) A synonym for **hold**[217].

Print (P) Alias for **Type**[296].

print (p) Research UNIX equivalent of **type**[297].

quit (q) Initiate a full program exit that includes updating the active mailbox, preserving of messages marked **hold**[217] or **preserve**[249], or never being referenced in the system *inbox*[453] (also see **primary system mailbox**[136]), saving of messages in the **secondary mailbox**[137] MBOX[638], saving of the the line editor *history-file*[442], among others. *on-account-cleanup*[494] is evaluated when set. If new mail has arrived during the session, the message "You have new mail" will be shown. A return to the shell is effected. The optional status number argument will be passed through to exit(3)[719]. [v15 behaviour may differ] For now it can happen that the given status will be overwritten, later this will only occur if a later error needs to be reported onto an otherwise success indicating status.

read [Only new quoting rules] Read a line from standard input, or the **readctl**[256] activated descriptor, split as indicated by *ifs*[449], into to the given **local**[130]izable variable(s), which must honour **variable name rules**[133] (*!*[351] error codes apply); the exit status *?*[350] reports the number of bytes read, or '-1' on error, with *!*[351] set to *^ERR*[356]-BADF in case of I/O errors, or *^ERR*[356]-NONE upon End-Of-File (with no more bytes to read). With more fields than variables, successive data is assigned to the last; with less fields than variables, unused are assigned empty strings. (Behaves like SHELL[643]s read(1)[720] command with **-r** [79] option used.)

```
? read a b c
   H  e  l  l  o
? echo "$? <$a> <$b> <$c>"
16 <H> <e> <l  l  o>
? set ifs=:; read a b c;unset ifs
hey2.0,:"'you    ",:world!:mars.:
? echo <$a><$b><$c><$d>
<hey2.0,><"'you    ",><world!:mars.:><>
? set ifs=:; read a b c d;unset ifs
hey2.0,:"'you    ",:world!:mars.:
? echo <$a><$b><$c><$d>
<hey2.0,><"'you    ",><world!><mars.>
```

readsh [Only new quoting rules] Like **read**[253], but splits on shell token boundaries (**Shell-style argument quoting**[24]) rather than at *ifs*[449]. [v15 behaviour may differ] Could become a **commandalias**[166], maybe read --tokenize --.

readall
[Only new quoting rules] Read anything unexpanded from standard input, or the **readctl**[256] activated descriptor, into to the given **local**[130]izable variable, which must honour **variable name rules**[133] (*!*[351] error codes apply); the exit status *?*[350] reports the number of bytes read, or '-1' on error, with *!*[351] set to *^ERR*[356]-BADF in case of I/O errors, or *^ERR*[356]-NONE upon End-Of-File (with no more bytes to read). [v15 behaviour may differ] The input data length is restricted to 31-bits.

readctl
[Only new quoting rules] Manage input channels for **read**[253], **readsh**[254] and **readall**[255]. Errors are reported via error number*!* *[351]* and exit status *?*[350]. Channel names are first checked for standard input hyphen-minus '-', which always exists, then for a numeric file descriptor number, otherwise a filename is assumed; filename based channels undergo

minimal **Filename transformations[28]** (no meta expansion are performed).

Without arguments known channels are listed, as with **show**. New active ones can be **create**d, existing ones can be **set** active and **remove**d by giving their name. Shared (fcntl(2)[721]) **lock**s can be applied but to standard I/O channels: this is tried several times and can thus block and fail, an uppercase first letter will log iterations (case-insensitivity still applies besides). [Option](*features[425]* includes ,+flock,) The otherwise identical **flock** lock type uses flock(2)[722]. Also see**fop [208]**.

```
$ printf 'echon "hey, "\nread a\nyou\necho $a' |\
  s-nail -R#
hey, you

$ LC_ALL=C printf 'echon "hey, "\nread a\necho $a' |\
  LC_ALL=C 6<<< 'you' s-nail -R#X'readctl create 6'
hey, you

? set fn=/tmp/members.dat
? readctl create $fn
? set rv=$? es=$! ed=$^ERRDOC
? if $rv -eq 0
?   readctl Lock $fn # ignore error
?   call .read-and-handle-until-eof "$fn"
?   readctl remove $fn
? else
?   echoerr $'Cannot read file \$fn: \$ed'
? endif
```

remove [Only new quoting rules] Remove named **folders[205]**. The given names are mailbox type classified, and type specific removal will be applied; removal of unknown types is refused. In interactive mode the user is asked for confirmation.

rename [Only new quoting rules] Rename the **folder[203]** given first to the name given second. **Filename transformations[28]** including shell pathname wildcard pattern expansions (glob(7)[723]) are performed on both arguments. Both folders must be of the same type.

Reply, **Respond**
(Compose mode)(R) Identical to **reply[261]** except that it replies to only the senders of the given messages, by using the first message as the template to quote, for the Subject: etc.; setting *flipr[426]* will exchange this command with **reply[261]**.

reply, **respond**
(Compose mode)(r) Successively group reply to each of the given messages by addressing the sender and all recipients, subject to *fullnames[437]* and **alternates[149]** processing. *followup-to[429]*, *followup-to-honour[431]*, *reply-to-honour[543]* as well as *recipients-in-cc[536]* influence response behaviour. *quote[528]* as well as *quote-as-attachment[530]* configure whether responded-to messages shall be quoted etc. *record[537]* controls saving the response message. Setting *flipr[426]* will exchange this command with **Reply[259]**. **Lreply[222]** offers special support for replying to mailing lists. For more documentation refer to **On sending mail, and non-interactive mode[7]**.

This may generate the errors *^ERR[356]*-DESTADDRREQ if no recipient has been specified, or was rejected by *expandaddr[422]* policy, *^ERR[356]*-IO if an I/O error occurs, *^ERR[356]*-NOTSUP if a necessary character set conversion fails, and *^ERR[356]*-INVAL for other errors. It can also fail with errors of **Specifying messages[14]**. Any error stops processing

of further messages.

Resend Like **resend[264]**, but does not add any header lines. This is not a way to hide the sender's identity, but useful for sending a bounced message again to the same recipients.

resend Send all given messages to the given recipient (*fullnames[437]*). `Resent-From:` and related header fields are prepended to sent messages. Saving in *record[537]* is only performed if *record-resent[539]* is set. [v15 behaviour may differ](Compose mode) is not entered, the only supported hooks are *on-resend-enter[506]* and *on-resend-cleanup[505]*.

This may generate the errors *ˆERR[356]*-DESTADDRREQ if no recipient has been specified, or was rejected by *expandaddr[422]* policy, *ˆERR[356]*-IO if an I/O error occurs, *ˆERR[356]*-NOTSUP if a necessary character set conversion fails, and *ˆERR[356]*-INVAL for other errors. It can also fail with errors of **Specifying messages[14]**. Any error stops processing of further messages.

retain (ret) Superseded by the multiplexer **headerpick[212]**.

return Return control of execution to the outer scope. Two optional positive decimal number arguments (default 0): the first is the 32-bit ([v15 behaviour may differ] later 64-bit) return value (*?[350]*), the second the 32-bit error number (*![351]*). As documented for*? [350]* a non-0 exit status may cause the program to exit. Only available inside of a **define[174]**d macro or an **account[144]**.

Save (S) Similar to **save,** but save messages to a file named after the local part of the sender of the first message instead of taking a filename argument; *outfolder[493]* is inspected to decide on the actual storage location.

save (s) Append the given messages to the given filename, which undergoes **Filename transformations[28]** including shell pathname wildcard pattern expansions (`glob`(7)[724]). If no filename is given, the **secondary mailbox[137]** `MBOX[638]` is used. The quoted filename and the generated byte count is shown on success. If editing a **primary system mailbox[136]** the messages are marked for deletion. The `save` slot of the white- and blacklisting command **headerpick[212]** filters header fields to be saved. Also see**Copy [168]**.

search Displays a header summary of all messages given, as via **headers[214]**. This is an alias of **from[211]**. Also see**Specifying messages [14]**.

seen Marks all given messages as having been read.

set, **unset**

(se, [Only new quoting rules] uns) Manage (set and clear) built-in and free-form custom **INTERNAL VARIABLES[31]**. Without arguments the former lists currently existing ones, with *debug[412]* or *verbose[616]* their attributes are included as comments. **Note**: this mode does not automatically establish **environ[193]** **link**for b uilt-in **ENVIRONMENT[34]** variables: only explicit addressing via **varshow[303]** (with arguments), usage in an **if[218]** condition or an argument to **echo[185]**, explicit **set**ting (not necessarily via **environ[193]**), as well as some program-internal use cases (look-ups) do this.

With arguments the given `name`s or `name=value` pairs (no space before or after the equal sign '=') are set or adjusted. Prefixing '`no`', for example `set noname`, equals calling **unset** (`unset name`), which erases all given variable names. They both support **local[130]** scoping. **Note: local[130]**ized free-form custom names use an isolated namespace and do not affect **environ[193]**ment links; on the other hand changes to built-in variables are passed down the call chain.

```
? set atab=$'' aspace=' ' zero=0 noprompt
? define mboxfix {
  local set mbox-rfc4155
```

```
    File "${1}"; copy * "${2}"
}
```

setdot Set the “dot” to the given message. Error number *![351]* is supported.

shcodec
: Apply *[+]e[ncode]* or *d[ecode]* shell quoting rules to the following raw-data. The result of `+encode` is not roundtrip enabled (Unicode etc.: only local decode); also see **mle-quote-rndtrip[108]**. On error*! [351]* is set to *ˆERR[356]*-CANCELED, and the unmodified input is the result. (Error may change again due to output or result storage errors.) Supports **vput[132]** (**Command modifiers[22]**).

shell (sh) Invokes an interactive `SHELL[643]`, then returns its exit status.

shortcut, **unshortcut**
: [Only new quoting rules] Manage the filename shortcuts of **folder[203]**. The latter deletes all shortcuts given as arguments (all with asterisk ‘*’). Without arguments the former shows all currently defined shortcuts, with one the target of the given. Otherwise arguments specify pairs of shortcut names and expansions, creating new or updating already existing ones.

shift [Only new quoting rules] Shift positional parameter stack (starting at *1[365]*) by the given positive decimal number, or 1 without argument. Giving 0 causes no action, successfully. Shifting more than the number of positional parameters is an error. The stack as such can be managed via **vpospar[307]**.

show Show raw message content (like **type[297]**, but perform neither MIME decoding nor decryption).

size (si) For all given messages, show their number, and the lines and bytes of the raw message content.

sleep [Only new quoting rules] Sleep the given number of seconds (and optionally milliseconds), With any third argument the sleep is uninterruptible, otherwise *![351]* will be set to *ˆERR[356]*-INTR upon interruption. If the given duration(s) overflow the time datatype an *ˆERR[356]*-OVERFLOW error occurs, if the given durations are no valid integers *ˆERR[356]*-INVAL.

sort, **unsort**
: Without arguments the former only shows, otherwise it sets the current sorting criterion; the latter is the same as `sort none`. The criterion affects the visual **folder[203]** representation, as well as addressing modes when **Specifying messages[14]** and the meaning of **next[241]**. The unique message numbers remain unchanged. A newly (un)ordered **headers[214]** summary is displayed if *header[438]* is set. For automatic **folder[203]** sorting set *autosort[382]*, as in **set[271]** autosort=thread. Possible sorting criterions are:

  date Sort by `Date:` field (message send time).
  from Sort by `From:` field (address of sender). The sender’s real name (if any) is used if *showname[554]* is set.
  none Do not sort (same as **unsort**).
  size Sort by size.
  spam [Option] Sort by spam score classified by **spamrate[289]**.
  status Sort by **Message states[13]**.
  subject Sort by subject.
  thread Create a threaded display. *autocollapse[380]* may be set to **collapse[162]** threads automatically.
  to Sort by `To:` field (recipient address). The recipient’s real name (if any) is used if *showname[554]* is set.

source [Only new quoting rules](so) Reads commands from the given file after doing **Filename transformations[28]**. A filename with a trailing vertical bar '|' is interpreted as a `SHELL[643]` command, the output of which will be read. Dependent on *posix[523]* and *errexit[419]*, as well as on the modifier **ignerr[129]**, encountered errors will stop reading (or even cause program exit). [v15 behaviour may differ] **source** cannot be used from within hook macros like *on-mailbox-open[501]*, nor from **account[144]**s, but only from macros that were **call[155]**ed.

source_if
[Only new quoting rules] Different to **source[284]** (beside not supporting pipe aka shell command input) is that no error is generated if the given file cannot be opened successfully.

spamclear
[Option] Clears the `is-spam` flag of the given messages.

spamforget
[Option] Pass the given messages to the *spam-interface[579]* so that it can un-train its Bayesian filter. Unless otherwise noted the`is-spam` flag is inspected to decide what shall be for gotten, "ham" or "spam".

spamham
[Option] Pass the given messages to the *spam-interface[579]* to inform it they are "ham". This also clears the `is-spam` flag.

spamrate
[Option] Pass the given messages to the *spam-interface[579]* for rating purposes, without modifying messages, but only setting the `is-spam` accordingly. The rating and flag will be forgotten once the **folder[203]** is left. **Handling spam[20]** shows the complete picture.

spamset
[Option] Sets the `is-spam` flag of the given messages.

spamspam
[Option] Pass the given messages to the *spam-interface[579]* to inform it they are "spam". This also sets the flag `is-spam`.

tls [Option][Only new quoting rules] TLS information and management command multiplexer to aid in **Encrypted network communication[19]** with the given URL (**On URL syntax and credential lookup[18]**). The port of the URL's `server:port` defaults to HTTPS (443); only protocols which establish TLS directly are supported, not those which upgrade an insecure channel. Subcommands support**vput [132]** (**Command modifiers[22]**). The error result is the empty string, the error itself is stored in *![351]*. For example, string length overflows are caught and set *![351]* to *^ERR[356]*-OVERFLOW. The TLS configuration is honoured, for example *tls-verify[608]* and *tls-config-pairs[601]*.

```
? vput tls result fingerprint pop3s://ex.am.ple
? echo $?/$!/$^ERRNAME: $result
```

certchain Show the complete verified peer certificate chain of the given URL. Includes informational fields in conjunction with *verbose[616]*.

certificate Show only the peer certificate, without any signers, of the given URL. Includes informational fields in conjunction with *verbose[616]*.

fingerprint Show the *tls-fingerprint-digest[606]*ed fingerprint of the certificate of the given URL. *tls-fingerprint[605]* is actively ignored for the runtime of this command.

Top Like **top**[294] but uses the **headerpick**[212] `type` slot for white- and blacklisting header fields.

top (to) **type**[297]s out the first *toplines*[609] lines of each of the given messages. Unless `top` selection has been established with **headerpick**[212], only `From:`, `To:`, `Cc:`, and `Subject:` are shown. The message content display can be compressed via*topsqueeze* [610].

touch (tou) Marks the given messages for saving in the **secondary mailbox**[137] `MBOX`[638]. POSIX deviation: an adjacent **next**[241] will **type**[297] the following, not the current message.

Type (T) Like **type**[297] but do not filter header fields through **headerpick**[212], and visualize all parts of MIME `multipart/alternative` messages.

type (t) Show the given messages. Whether and when `PAGER`[640] is used for display instead of the *screen*[547] is controlled by *crt*[408]. (**more**[237] always uses `PAGER`[640].) Message headers are filtered as chosen by the `type` selection of **headerpick**[212]. For MIME multipart messages all `text` message parts, all parts with a registered MIME type handler (**HTML mail and MIME attachments**[10]) that produces **copiousoutput**[661], and all `message` parts are shown, others are hidden except for their headers. Messages are decrypted and converted to *ttycharset*[611] as necessary. **mimeview**[229] can be used to display parts of other sort.

unaccount[145]
See **account**[144].

unalias[148]
(una) See **alias**[147].

unanswered[152]
See **answered**[151].

unbind[154]
See **bind**[153].

uncollapse[163]
See **collapse**[162].

uncolour[165]
See **colour**[164].

undefine[175]
See **define**[174].

undelete[177]
See **delete**[176].

undraft[184]
See **draft**[183].

unflag[201]
See **flag**[200].

unignore
Superseded by the multiplexer **headerpick**[212].

unmimetype[228]
See **mimetype**[227].

unmlist[231]
See **mlist**[230].

unmlsubscribe[233]
: See **mlsubscribe[232]**.

Unread Same as **unread[300]**.

unread Marks all given messages as not being read (**Message states[13]**).

unretain
: Superseded by the multiplexer **headerpick[212]**.

unset[272]
: [Only new quoting rules](uns) See **set[271]**.

unshortcut[277]
: See **shortcut[276]**.

unsort[283]
: See **sort[282]**.

urlcodec
: Apply *e[ncode]*, *d[ecode]*, *p[ath]enc[ode]* or *p[ath]dec[ode]* URL percent codec (RFC 3986) operations on the following raw-data. The latter two are slightly modified to better adhere to filenames: tilde ‘~’ is not allowed, and the initial character can neither be hyphen-minus ‘-’ nor dot ‘’. ([v15 behaviour may differ] The path-aware decoder is yet identical to the normal one.) This is a character set agnostic operation which could decode bytes that are invalid in the current *ttycharset[611]*.

  Supports **vput[132]** (see **Command modifiers[22]**), and manages the error number *![351]*. If the operation fails *![351]* is set to *^ERR[356]*-CANCELED, and the unmodified input is the result (error number may change again due to output or result storage errors). [v15 behaviour may differ] This command does not know about URLs beside what is documented. (**vexpr[306]** offers a **makeprint** subcommand, shall the URL be displayed.)

varshow
: [Only new quoting rules] Show only the given variables, or all **INTERNAL VARIABLES[31]**. In the latter case **local[130] scope** settings are not covered, **vpospar[307]** and other positional parameters are never covered. The output is subject to*verbose [616]*.

verify [Option] Verifies all given messages. If a message is not a S/MIME signed message, verification will fail for it. The process checks if the message was signed using a valid certificate, whether the sender address matches one of those certificate presented, and whether the message content has not been altered.

version
: Show *version[617]* and *features[425]*, optionally in a more *verbose[616]* form that includes build and running system environment informations. Supports **vput[132]** (**Command modifiers[22]**).

vexpr [Option][Only new quoting rules] Multiplexer which offers signed 64-bit numeric calculations, as well as other, mostly string-based operations. C-style byte string operations are available via **csop[170]**, and file operations via **fop[208]**. The first argument defines number, type, and meaning of the remaining arguments. An empty number argument is treated as 0. Supports **vput[132]** (**Command modifiers[22]**). The result in case of errors is ‘-1’ for usage errors and numeric operations, the empty string otherwise; “soft” errors, like when a search operation failed, will also set the *![351]* error number to *^ERR[356]*-NODATA. Except when otherwise noted numeric arguments are parsed as signed 64-bit numbers, and errors will be reported in the error number *![351]* as the numeric error *^ERR[356]*-RANGE.

Numeric operations work on one or two signed 64-bit integers according to **number syntax rules[135]**. Unsigned interpretation of a number can be enforced with an ‘u’ prefix (case-insensitive), as in `u-110`; power-of-two bases (2,4,8,16,32,64) are unsigned by default, but regarding overflow detection and error constants it still makes a difference. To enforce signed interpretation (instead) prefix ‘s’ (case-insensitive). One integer is expected by assignment (equals sign ‘=’), which does nothing but validity and overflow detection, unary not (tilde ‘~’), which creates the bitwise complement, and unary plus and minus. Two integers are used by addition (plus sign ‘+’), subtraction (hyphen-minus ‘-’), multiplication (asterisk ‘*’), division (solidus ‘/’) and modulo (percent sign ‘%’), as well as for the bitwise operators logical or (vertical bar ‘|’, to be quoted) , bitwise and (ampersand ‘&’), bitwise xor (circumflex ‘^’), the bitwise signed left- and right shifts (‘<<’, ‘>>’), as well as for the unsigned right shift `>>>`.

Another numeric operation is **`pbase`**, which takes a number base in between 2 and 64, inclusive, and will act on the second number given just the same as what equals sign ‘=’ does, but the number result will be formatted in the base given, as a signed 64-bit number unless unsigned interpretation of the input number had been forced.

Numeric operations support a saturated mode via the question mark ‘?’ modifier suffix; the keyword `saturated` is optional, ‘+?’, `+?satu`, and `+?saturated` are therefore identical. In saturated mode overflow errors and division and modulo by zero are no longer reported via the return status, but the result will linger at the minimum or maximum possible value, instead of overflowing (or trapping). This is true also for the argument parse step. For the bitwise shifts, the saturated maximum is 63. Any caught overflow will be reported via the error number *![351]* as *^ERR[356]*-OVERFLOW.

```
? vput vexpr res -? +1 -9223372036854775808
? echo $?/$!/$^ERRNAME:$res
0/75/OVERFLOW:-9223372036854775808
```

Character set agnostic string functions have no notion of locale settings and character sets.

`date-utc` Outputs the current date and time in UTC (Coordinated Universal Time) with values named such that `vput vexpr x date-utc; eval set $x` creates accessible variables. An optional argument denotes the UNIX seconds-since-the-epoch to be used instead of the current time. (The algorithm does not know about leap seconds and works until 65535-12-31T23:59:59.)

`date-stamp-utc` Outputs the current UTC time in RFC 3339 internet date/time format.

`epoch` The seconds and nanoseconds since the Unix epoch (1970-01-01T00:00:00) named `epoch_sec` and `epoch_nsec` such that `vput vexpr x epoch; eval set $x` creates accessible variables. May be given one to six optional arguments naming (in order) year, month, day (of month), hour, minute and second to be used instead of the current time. (The algorithm works until 65535-12-31T23:59:59.)

`random` Generates a random string of the given length, or of `PATH_MAX` bytes (a constant from `/usr/include`) if the value 0 is given; the random string will be base64url encoded according to RFC 4648, and thus be usable as a (portable) filename.

String operations work, sufficient support provided, according to the active user’s locale encoding and character set (**Character sets[12]**). Where the question mark ‘?’ modifier suffix is supported, a case-insensitive operation mode is available; the keyword `case` is optional, `regex?` and `regex?case` are therefore identical.

`makeprint` (One-way) Converts the argument to something safely printable on the terminal.

regex [Option] A string operation that tries to match the first argument with the regular expression (re_format(7)[725] or regex(7)[726], dependent on host system) given as the second argument. '?' modifier suffix is supported. With the optional third argument the match group accessors *^#[353]*, *^0[354]*, *^1[355]* are set (in **local[130]**-most scope) for successful matches, and the argument is interpreted as if specified within dollar-single-quote (**Shell-style argument quoting[24]**):

```
? vput vexpr res regex bananarama \
    (.*)NanA(.*) '\${^1}au\$^2'
? echo $?/$!/$^ERRNAME:$res:
1/61/NODATA::
? vput vexpr res regex?case bananarama \
    (.*)NanA(.*) '\${^1}uauf\$^2'
? echo $?/$!/$^ERRNAME:$res:
0/0/NONE:bauauframa:
```

vpospar

[Only new quoting rules] Manage the scope's positional parameter stack (see *1[365]*, *#[363]*, **[361]*, *@[362]* as well as **shift[278]**). The global scope can be enforced with the modifier **global[128]**. If the first argument is clear the stack is cleared. otherwise, is cleared. If it is set the remaining arguments will be used to (re)create the stack, if the parameter stack size limit is excessed an *^ERR[356]*-OVERFLOW error will occur. evalset acts likewise, but only takes one argument to furtherly **eval[127]**uate in a special mode where number sign '#' is an ordinary character (**Shell-style argument quoting[24]**).

If the first argument is quote, a round-trip capable representation of the stack contents is created, with each quoted parameter separated from each other with the first character of *ifs[449]*, and followed by the first character of *if-ws*, if that is not empty and not identical to the first. If that results in no separation at all a **space** character is used. This mode supports **vput[132]** (**Command modifiers[22]**). The subcommandsset and quote can be used (in conjunction with **eval[127]**) to losslessly (re)create an argument stack from and to a single variable.

```
? vpospar set hey, "'you    ", world!
? echo $#: <$1><$2><$3>
? vput vpospar x quote
? vpospar clear
? echo $#: <$1><$2><$3>
? eval vpospar set $x
? echo $#: <$1><$2><$3>
?
? set x=$'a b\n#c d\ne f\n'
? set ifs=$'\n'; vpospar set $x; unset ifs
? echo $#: <$1><$2><$3>
1: <a b
#c d
e f
><><>
? set ifs=$'\n'; eval vpospar set $x; unset ifs
? echo $#: <$1><$2><$3>
1: <a b><><>
? set ifs=$'\n'; vpospar evalset $x; unset ifs
? echo $#: <$1><$2><$3>
3: <a b><#c d><e f>
```

visual (v) Successively invoke the VISUAL[650] display editor on the given messages. Modified contents are discarded unless *writebackedited[623]* is set, and are not used unless the mailbox can be written to, and the editor returns a successful exit status. **edit [189]** can be used instead for a less display oriented editor.

write (w) For conventional messages the body without all headers is written. The original message is never marked for deletion in the originating mail folder. The output is decrypted and converted to its native format as necessary. If the output file exists, the text is appended. If a message is in MIME multipart format its first part is written to the specified file as for conventional messages, handling of the remains depends on the execution mode. No special handling of compressed files is performed.

In interactive mode the user is consecutively asked for the filenames of the processed MIME parts. For convenience saving of each part may be skipped by giving an empty value, the same result as writing it to /dev/null[657]. Shell piping the part content by specifying a leading vertical bar '|' character for the filename is supported. Other user input undergoes the usual **Filename transformations[28]**, including shell pathname wildcard pattern expansions (glob(7)[727]) and shell variable expansion for the message as such, not the individual parts, and contents of the destination file are overwritten if the file previously existed. Character set conversion to *ttycharset[611]* is performed when saving text data.

[v15 behaviour may differ] In non-interactive mode any part which does not specify a filename is ignored, and suspicious parts of filenames of the remaining parts are URL percent encoded (as via **urlcodec[302]**) to prevent injection of malicious character sequences, resulting in a filename that will be written into the current directory. Existing files will not be overwritten, instead the part number or a dot are appended after a number sign '#' to the name until file creation succeeds (or fails due to other reasons).

xcall [Only new quoting rules] The sole difference to **call[155]** is that the new macro is executed in place of the current one, which will not regain control: all resources of the current macro will be released first, except the re-parented settings covered by **local[130]** scoping. If this command is not used from within a **call[155]**ed macro it will silently be (a more expensive variant of) **call[155]**.

xit (x) A synonym for **exit[195]**.

z [Only new quoting rules] Message **headers[214]** are shown in *screen[547]*fuls. Without arguments this command scrolls to the next screen of messages, likewise if given '+'. An argument of '-' scrolls to the last, '^' scrolls to the first, and '$' to the last screen. A number argument prefixed by '+' or '−' indicates that the window is calculated in relation to the current position, and a number without a prefix specifies an absolute position.

Z [Only new quoting rules] Similar to **z[312]**, but scrolls to the next or previous *screen[547]* that contains at least one new or **flag[200]**ged message.

COMMAND ESCAPES

Compose mode[8] escapes offer access to attachments, header editing, invocation of **COMMANDS[21]** and more. They are available when interactive, when requested via **−~ [89]**, as well as in batch mode (**−#[90]**). They are only recognized in the leftmost column and consist of an *escape[421]* character (default tilde '~'), optional modifiers (that act like **Command modifiers[22]**), and a command character. Interspersed whitespace is ignored. To write an *escape[421]* character in the leftmost column, double it. [Option] Addition to the **history[216]** is prevented by placing any number of whitespace after *escape[421]*. [Option] Key **bind[153]**ings support a command escape specific context. Modifiers are

- Hyphen-minus ‘`-`’ acts like **`ignerr`**[129] does for **COMMANDS**[21], and overrides *errexit*[419].

```
? set errexit; reply .
~ - : illegal
s-nail: illegal: unknown command
~:illegal
s-nail: illegal: unknown command
s-nail: Failed to prepare composed message
$
```

- Dollar ‘`$`’ **`eval`**[127]uates the remains of the line as often as it is used (**Shell-style argument quoting**[24]). [v15 behaviour may differ] For now the entire input line is evaluated as a whole; to avoid that control operators like semicolon **`;`** are interpreted unintentionally, they must be quoted.

```
? set var='bla bla bla'; reply .
~:echo '$var'
$var
~$:echo '$var'
bla bla bla
```

Unless otherwise documented escapes manage the error number *!*[351] and the exit/return status *?*[350], and so *errexit*[419] may cause leaving compose mode and program exits.

`~!` *`command`*
: Execute the given `SHELL`[643] *`command`*, replacing unescaped exclamation marks with the previously executed command if *bang*[383] is set.

`~.`
: **Compose mode**[8] is left, and the message is sent. The hooks *on-compose-splice-shell*[499] and *on-compose-splice*[498], in order, are called when set, after which, in interactive mode, *askatend*[370] (leading to *askcc*[372], *askbcc*[373]) and *askattach*[371] will be checked as well as *asksend*[374], after which a set *on-compose-leave*[497] hook will be called, *autocc*[379] and *autobcc*[378] will be joined in if set, finally a given *message-inject-tail*[474] will be incorporated.

`~:` *`command`* or **`~_`** *`command`*
: Can be used to execute **COMMANDS**[21] (not all are allowed in compose mode).

`~<` *`filename`*
: Identical to **`~r`**[341].

`~<!` *`command`*
: *`command`* is executed using the `SHELL`[643], and its standard output is inserted into the message.

`~?`
: [Option] Write a summary of command escapes.

`~@` [*`filename...`*]
: Append or edit the list of attachments. Does not manage error number *!*[351] and exit status *?*[350] (use **`~^`**[322] if error handling is necessary). The append mode expects a list of *`filename`* arguments as shell tokens (**Shell-style argument quoting**[24]; token-separating commas are also ignored), to be interpreted as documented for the command line option **`-a`**[58], with the message number exception as below.

  Without *`filename`* arguments the attachment list is edited, entry by entry; if a filename is left empty, that attachment is deleted from the list; once the end of the list is reached either new attachments may be entered or the session can be quit by committing an empty input. In non-interactive as well as in batch mode (**`-#`**[90]) the list of attachments is effectively not edited but instead recreated; again, an empty input ends list creation.

For all modes, if a given filename solely consists of the number sign '#' followed by either a valid message number of the currently active mailbox, or by a period '.', referring to the current message of the active mailbox, the so-called "dot", the given message is attached as a `message/rfc822` MIME message part. The number sign must be quoted to avoid misinterpretation as a shell comment character.

~| *command*

Filter the message text through a `SHELL`[643] *command*; its standard output forms the new message text. If no output or a non-0 exit status is generated, the original message text is retained. The command `fmt`(1)[728] is often used as a rejustifying filter (`~| /usr/bin/fmt -tuw11`).

`~||` instead filters the entire message including header fields, so that `~|| echo Fcc: /tmp/test; cat`will prepend a file-carbon-cop y message header. Also see**~e** [328], **~v**[346].

~^ *cmd* [*subcmd* [*arg3* [*arg4*]]]

Inspect and modify the message using the semantics of **digmsg**[178], therefore arguments are evaluated according to **Shell-style argument quoting**[24]. Error number*!* *[351]* and exit status *?[350]* are not managed: errors are handled via the protocol, and hard errors like I/O failures cannot be handled.

The protocol consists of command lines followed by (a) response line(s). The first field of the response line represents a status code which specifies whether a command was successful or not, whether result data is to be expected, and if, the format of the result data. Response data will be shell quoted as necessary for consumption by **readsh**[254], or **vpospar**[307] and **eval**[127], to name a few. Error status code lines may optionally contain additional context:

`210` Status ok; the remains of the line are the result.

`211` Status ok; the rest of the line is optionally used for more status. What follows are lines of result addresses, terminated by an empty line. All the input, including the empty line, must be consumed before further commands can be issued. Address lines consist of two token, first the plain network address, e.g., `bob@exam.ple`, followed by the (quoted) full address as known: `'(Lovely) Bob <bob@exam.ple>'`. Non-network addresses use the first field to indicate the type (hyphen-minus '-' for files, vertical bar '|' for pipes, and number sign '#' for names which will undergo **alias**[147] processing) instead, the actual value will be in the second field.

`212` Status ok; the rest of the line is optionally used for more status. What follows are lines of furtherly unspecified (quoted) string content, terminated by an empty line. All the input, including the empty line, must be consumed before further commands can be issued.

`500` Syntax error; invalid command.

`501` Syntax error or otherwise invalid parameters or arguments.

`505` Error: an argument fails verification. For example an invalid address has been specified (also see *expandaddr[422]*), or an attempt was made to modify anything in S-nail's own namespace, or a modifying subcommand has been used on a read-only message.

`506` Error: an otherwise valid argument is rendered invalid due to context. For example, a second address is added to a header which may consist of a single address only.

If a command indicates failure then the message will have remained unmodified. Most commands can fail with `500` if required arguments are missing, or excessive arguments have been given (false command usage). ([v15 behaviour may differ] The latter does not yet occur regularly, because as stated in **Shell-style argument quoting**[24] our argument parser is not yet smart enough to work on subcommand base; for example one might get excess argument error for a three argument subcommand that receives four arguments, but not for a four argument subcommand which receives six arguments: here excess will be joined.) The following (case-insensitive) commands are

supported:

attachment This command allows listing, removal and addition of message attachments. The second argument specifies the subcommand to apply, one of:

attribute This uses the same search mechanism as described for **remove** and prints any known attributes of the first found attachment via 212 upon success or 501 if no such attachment can be found. The attributes are written as lines with a keyword and a value token.

attribute-at This uses the same search mechanism as described for **remove-at** and is otherwise identical to **attribute**.

attribute-set This uses the same search mechanism as described for **remove**, and will set the attribute given as the fourth to the value given as the fifth token argument. If the value is an empty token, then the given attribute is removed, or reset to a default value if existence of the attribute is crucial.

It returns via 210 upon success, with the index of the found attachment following, 505 for message attachments or if the given keyword is invalid, and 501 if no such attachment can be found. The following keywords may be used (case-insensitively):

filename Sets the filename of the MIME part, i.e., the name that is used for display and when (suggesting a name for) saving (purposes).

content-description Associate some descriptive information to the attachment's content, used in favour of the plain filename by some MUAs.

content-id May be used for uniquely identifying MIME entities in several contexts; this expects a special reference address format as defined in RFC 2045 and generates a 505 upon address content verification failure.

content-type Defines the media type/subtype of the part, which is managed automatically, but can be overwritten.

content-disposition Automatically set to the string attachment.

attribute-set-at This uses the same search mechanism as described for **remove-at** and is otherwise identical to **attribute-set**.

insert Adds the attachment given as the third argument, specified exactly as documented for the command line option **-a [58]**, and supporting the message number extension as documented for ~@**[320]**. This reports210 upon success, with the index of the new attachment following, 505 if the given file cannot be opened, 506 if an on-the-fly performed character set conversion fails, otherwise 501 is reported; this is also reported if character set conversion is requested but not available.

list List all attachments via 212, or report 501 if no attachments exist. This command is the default command of **attachment** if no second argument has been given.

remove This will remove the attachment given as the third argument, and report 210 upon success or 501 if no such attachment can be found. If there exists any path component in the given argument, then an exact match of the path which has been used to create the attachment is used directly, but if only the basename of that path matches then all attachments are traversed to find an

exact match first, and the removal occurs afterwards; if multiple basenames match, a `506` error occurs. Message attachments are treated as absolute pathnames.

If no path component exists in the given argument, then all attachments will be searched for `filename=` parameter matches as well as for matches of the basename of the path which has been used when the attachment has been created; multiple matches result in a `506`.

`remove-at` This will interpret the third argument as a number and remove the attachment at that list position (counting from one!), reporting `210` upon success or `505` if the argument is not a number or `501` if no such attachment exists.

`header` This command allows listing, inspection, and editing of message headers. Header name case is not normalized, so that case-insensitive comparison should be used when matching names. The second argument specifies the subcommand to apply, one of:

`insert` Create a new or an additional instance of the header given in the third argument, with the header body content as given in the fourth token. It may return `501` if the third argument specifies a free-form header field name that is invalid, or if body content extraction fails to succeed, `505` if any extracted address does not pass syntax and/or security checks or on S-nail namespace violations, and `506` to indicate prevention of excessing a single-instance header — note that `Subject:` can be appended to (a space separator will be added automatically first). `To:`, `Cc:` and `Bcc:` support the `?single` modifier to enforce treatment as a single recipient, for example `header insert To?single: 'exa, <m@ple>'`; the word `single` is optional.

`210` is returned upon success, followed by the name of the header and the list position of the newly inserted instance. The list position is always 1 for single-instance header fields. All free-form header fields are managed in a single list; also see *customhdr[409]*.

`headerpick` Takes the name of a **`headerpick`**`[212]` *`context`* and filters accordingly, giving `210` on success, and `501` for read-only messages or an invalid *`context`*.

`list` Without a third argument a list of all yet existing headers is given via `210`; this command is the default command of **`header`** if no second argument has been given. A third argument restricts output to the given header only, which may fail with `501` if no such field is defined.

`remove` This will remove all instances of the header given as the third argument, reporting `210` upon success, `501` if no such header can be found, and `505` on S-nail namespace violations.

`remove-at` This will remove from the header given as the third argument the instance at the list position (counting from one!) given with the fourth argument, reporting `210` upon success or `505` if the list position argument is not a number or on S-nail namespace violations, and `501` if no such header instance exists.

`show` Shows the content of the header given as the third argument. The content will be converted to *ttycharset[611]* ([v15 behaviour may differ] and then be made printable according to `LC_ALL`[628]). Dependent on the header type this may respond with `211` or `212`; any failure results in `501`.

In compose mode read-only access to optional pseudo headers in the S-nail private namespace is available:

`Mailx-Command:`
: The name of the command that generates the message, one of `forward`, `Lreply`, `mail`, `Reply`, `reply`, `resend`. This pseudo header always exists (in compose mode).

`Mailx-Edited-Sender:`
`Mailx-Edited-Origin:`
: Different to the RFC 5322 originator fields of the otherwise identical `-Orig-` series below these might take into account *reply-to-honour[543]* and *reply-to-swap-in[544]*. This sender field will however be identical to `Mailx-Orig-Sender:` unless the edited (replaced) originator field is unambiguous.

`Mailx-Orig-Sender:`
`Mailx-Orig-From:`
`Mailx-Orig-To:`
`Mailx-Orig-Cc:`
`Mailx-Orig-Bcc:`
: The values of said headers of the original message which has been addressed by any of **`reply`**`[261]`, **`forward`**`[210]`, **`resend`**`[264]`. The sender field is filled in according to what is described for *from[436]*.

`Mailx-Raw-To:`
`Mailx-Raw-Cc:`
`Mailx-Raw-Bcc:`
: Represent the frozen initial state of these headers before any transformation (**`alias`**`[147]`, **`alternates`**`[149]`, *recipients-in-cc[536]* etc.) took place.

`epoch` Show the message date as seconds since epoch via `210`. Error for an invalid date is `501`, for using this command in compose mode `505`. For converting times you may want to look at **`vexpr`**`[306]`.

`help`, **`?`** Show an abstract of the above commands via `211`.

`version` This command will print the protocol version via `210`.

`~A` The same as **`~i`**`[334]` *Sign[556]*.

`~a` The same as **`~i`**`[334]` *sign[557]*.

`~b` *`name ...`*
: Add the given names to the list of blind carbon copy recipients.

`~c` *`name ...`*
: Add the given names to the list of carbon copy recipients.

`~d` Read the file specified by the `DEAD`[625] variable into the message.

`~e` Invoke the text `EDITOR`[626] on the message collected so far, then return to compose mode. **`~v`**`[346]` can be used for a more display oriented editor, and **`~|`**`[321]`| offers a pipe-based editing approach.

`~F` *`messages`*
: Read the named messages into the message being sent, including all message headers and MIME parts, and honouring*forwar d-add-cc[432]* as well as *forward-inject-head[434]* and *forward-inject-tail[435]*. If no messages are specified, read in the current message, the “dot”.

~f *messages*
: Read the named messages into the message being sent. If no messages are specified, read in the current message, the “dot”. Strips down the list of header fields according to the `forward` (with *posix[523]*: `type`) white- and blacklist selection of **headerpick[212]**, and honours *forward-add-cc[432]* as well as *forward-inject-head[434]* and *forward-inject-tail[435]*. For MIME multipart messages, only the first displayable part is included.

~H
: In interactive mode, edit the message header fields `From:`, `Reply-To:` and `Sender:` by typing each one in turn and allowing the user to edit the field. The default values for these fields originate from the *from[436]*, *reply-to[542]* and *sender[551]* variables. In non-interactive mode this sets *^ERR[356]*-NOTTY.

~h
: In interactive mode, edit the message header fields `To:`, `Cc:`, `Bcc:` and `Subject:` by typing each one in turn and allowing the user to edit the field. In non-interactive mode this sets *^ERR[356]*-NOTTY.

~I *variable*
: Insert the value of the specified variable into the message. The message remains unaltered if the variable is unset or empty. Any embedded character sequences ‘`\t`’ horizontal tabulator and ‘`\n`’ line feed are expanded in *posix[523]* mode; otherwise the expansion should occur at **set[271]** time (*v15-compat[615]*, **wysh[134]**).

~i *variable*
: Like **~I[333]**, but appends a newline character.

~M *messages*
: Read the named messages into the message being sent, indented by *indentprefix[454]*. If no messages are specified, read the current message, the “dot”. Honours *forward-add-cc[432]* as well as *forward-inject-head[434]* and *forward-inject-tail[435]*.

~m *messages*
: Read the named messages into the message being sent, indented by *indentprefix[454]*. If no messages are specified, read the current message, the “dot”. Strips down the list of header fields according to the `type` white- and blacklist selection of **headerpick[212]**. Honours *forward-add-cc[432]* as well as *forward-inject-head[434]* and *forward-inject-tail[435]*. For MIME multipart messages, only the first displayable part is included.

~p
: Display the message collected so far, prefaced by the message header fields and followed by the attachment list, if any.

~Q
: Read in the given / current message(s) using the algorithm of *quote[528]* (except that is implicitly assumed, even if not set), honouring *quote-add-cc[529]*.

~q
: Abort the message being sent, copying it to the file specified by the `DEAD[625]` variable if *save[546]* is set.

~R *filename*
: Identical to **~r[341]**, but indent each line that has been read by *indentprefix[454]*.

~r *filename* [*HERE-delimiter*]
: Read the named file, object to **Filename transformations[28]** excluding shell globs and variable expansions, into the message; if *filename* is the hyphen-minus ‘`-`’ then standard input is used (for pasting, for example). Only in this latter mode*HERE-delimiter* may be gi ven: if it is data will be read in until the given *HERE-delimiter* is seen on a line by itself, and encountering EOF is an error; the *HERE-delimiter* is a required argument in non-interactive mode; if it is single-quote quoted then the pasted content will not be expanded, [v15 behaviour may differ] otherwise a future version of S-nail may perform shell-style expansion on the content.

~s *string*
: Cause the named string to become the current subject field. Newline (NL) and carriage-return (CR) bytes are invalid and will be normalized to space (SP) characters.

~t *name ...*
: Add the given name(s) to the direct recipient list.

~U *messages*
: Read in the given / current message(s) excluding all headers, indented by *indentprefix[454]*. Honours *forward-add-cc[432]* as well as *forward-inject-head[434]* and *forward-inject-tail[435]*.

~u *messages*
: Read in the given / current message(s), excluding all headers. Honours *forward-add-cc[432]* as well as *forward-inject-head[434]* and *forward-inject-tail[435]*.

~v
: Invoke the VISUAL[650] editor on the message collected so far, then return to compose mode. **~e[328]** can be used for a less display oriented editor, and **~|[321]**| offers a pipe-based editing approach.

~w *filename*
: Write the message onto the named file, which is object to the usual **Filename transformations[28]**. If the file exists, the message is appended to it.

~x
: Same as **~q[339]**, except that the message is not saved at all.

INTERNAL VARIABLES

Variables are names that exist or not, and optionally expand to values. They can be created or changed with **set[271]** and erased with **unset[272]**. There are built-in variables which may have typed values and attributes, as explained below. Custom variables with optional (string) values may be defined freely. **varshow[303]** will inspect all built-in or the given variables, **set[271]** without arguments all currently existing ones; both support a detailed *verbose[616]* listing mode. Some built-in variables are synchronized from and with the program **ENVIRONMENT[34]**, others can be linked or created with **environ[193]** to henceforth have said property; these have to honour SHELL[643] **variable name rules[133]**.

```
? set one=val\ 1 two="val 2" \
    three='val "3"' four=$'val \'4\''; \
  environ set FIVE=val\ 5; \
  varshow one two three four FIVE; \
  unset one two three four FIVE; \
  varshow one two three four FIVE
```

Boolean variables have no value and can only be in the states “set” and “unset”. Values need proper quoting upon assignment time, the quoting rules are documented for **COMMANDS[21]**. Dependent upon the built-in variable values may become interpreted as colour names, command specifications, normal text, etc. They may be treated as numbers, in which case decimal values are expected if so documented, otherwise the usual **number syntax rules[135]** apply.

“Boolean string” is a special kind of string value, either a decimal integer (with ‘0’ being false and ‘1’ or any other value being true), or one of the (case-insensitive) strings off, ‘no’, ‘n’ andfalse for a f alse boolean and ‘on’, yes, ‘y’ andtrue for a true boolean. A special kind of boolean string is the “quadoption” which is optionally prefixed with the (case-insensitive) term ask-, as in ask-yes: in interactive mode the user will be prompted, otherwise the actual boolean is used.

Some built-in variables exist as so-called “chains” which extend the plain variable with variable-HOST and variable-USER@HOST variants. HereHOST will be con verted to all lowercase when looked up (but not when the variable is set or unset!), [Option]ally IDNA converted, and indeed means server:port if a port had been specified in the contextual Uniform Resource Locator URL, see **On**

URL syntax and credential lookup[18]. Even though this mechanism is based on URLs no URL percent encoding (**urlcodec[302]**) may be applied to neither of USER nor HOST, variable chains need to be specified using raw data; the mentioned section contains examples. Variables which support chains are explicitly documented as such, and since chains are special users should not create custom names like variable-xyz in order to avoid false classifications and treatment of such variables.

Initial settings

The standard POSIX 2008/Cor 2-2016 mandates the following initial settings for built-in internal variables: no*allnet[368]*, no*append[369]*, *asksub[376]*, no*askbcc[373]*, no*autoprint[381]*, no*bang[383]*, no*cmd[399]*, no*crt[408]*, no*debug[412]*, no*dot[414]*, *escape[421]* set to '~', no*flipr[426]*, no*folder[427]*, *header[438]*, no*hold[446]*, no*ignore[451]*, no*ignoreeof[452]*, no*keep[455]*, no*keepsave[457]*, no*metoo[475]*, no*outfolder[493]*, no*page[507]*, *prompt[525]* set to '? ', no*quiet[527]*, no*record[537]*, *save[546]*, no*sendwait[552]*, no*showto[555]*, no*Sign[556]*, no*sign[557]*, *toplines[609]* set to '5'.

However, some initial (and some default) settings are built-in, and (may) diverge, others may become adjusted by one of the **Resource files[36]**. Displaying the former is accomplished via**set [271]**: `$ s-nail -:/ -v -Xset -Xx`. In general this implementation sets (and has extended the meaning of) *sendwait[552]*, and does not support the no*onehop* variable – use command line options or *mta-arguments[484]* to pass options through to a *mta[482]*. The system-wide resource file as shipped, sets, among others, *hold[446]*, *keep[455]* and *keepsave[457]*, establishes a default **headerpick[212]** selection etc., and should thus be taken into account.

Built-in variables

? (Read-only) The exit status of the last command, or the **return[266]** value of the macro **call[155]**ed last. This status has a meaning in the state machine: in conjunction with *errexit[419]* any non-0 exit status will cause a program exit, and in *posix[523]* mode any error while loading (any of the) resource files will have the same effect. **ignerr[129]**, one of the **Command modifiers[22]**, can be used to instruct the state machine to ignore errors.

! (Read-only) The current error number (errno(3)[729]), which is set after an error occurred; it is also available via *^ERR[356]*, and the error name and documentation string can be queried via *^ERRNAME[358]* and *^ERRDOC[357]*. [v15 behaviour may differ] This machinery is new and the error number is only really usable if a command explicitly states that it manages the variable *![351]*, for others errno will be used in case of errors, or *^ERR[356]*-INVAL if that is 0: it thus may or may not reflect the real error. The error number may be set with the command**return [266]**.

^ (Read-only) This is a multiplexer variable which performs dynamic expansion of the requested state or condition, of which there are:

^#, ^0, ^1 Access to match groups of the last regular expression evaluation in **local[130]**-most scope, created for example by the **if[218]** command. The former denotes the number of match groups including *^0* that holds the entire matching string (0 for no match); for example

```
\if abrakadabra =~ (.+)ka.*
  \echo $^#: <$^0> <$^1> <$^2>
\end
# ->
2: <abrakadabra> <abra> <>
```

^ERR, ^ERRDOC, ^ERRNAME
The number, documentation, and name of the current errno(3)[730], respectively, usually set after an error occurred. The former as shown is identical to *![351]*. Documentation is an [Option], the name is used as a fallback. Each of these can be suffixed with a hyphen minus followed by a name or number, in which case the expansion refers to the

given error. Note this is a direct mapping of (a subset of) the system error values, with (high numbered) fallbacks for unsupported constants:

```
define work {
  \eval echo \$1: \$^ERR-$1: \
    \$^ERRNAME-$1: \$^ERRDOC-$1
  \eval if \$(($1 + 1)) -lt 16
    \eval xcall work \$(($1 + 1))
  \end
}
\call work $^ERR-NONE
```

^ERRQUEUE-COUNT, *^ERRQUEUE-EXISTS*
: The number of messages in the [Option]al queue of **errors[194]**, and a string indicating queue state: empty or (translated) “ERROR”. Always 0 and the empty string, respectively, unless *features[425]* includes `,+errors,`.

*
: (Read-only) Expands all positional parameters (see *1[365]*), separated by the first character of the value of *ifs[449]*. [v15 behaviour may differ] The special semantics of the equally named special parameter of the SHELL[643] are not yet supported.

@
: (Read-only) Expands all positional parameters (see *1[365]*), separated by a space character. If placed in double quotation marks, each positional parameter is properly quoted to expand to a single parameter again.

#
: (Read-only) Expands to the number of positional parameters, i.e., the size of the positional parameter stack in decimal.

0
: (Read-only) Inside the scope of a **define[174]**d and **call[155]**ed macro this expands to the name of the calling macro, or to the empty string if the macro is running from top-level or as a hook. For the [Option]al regular expression search and replace operator of **vexpr[306]** this expands to the entire matching expression. It represents the program name in global context, and “compose mode” in **Compose mode[8]**.

1
: (Read-only) Access of the positional parameter stack. All further parameters can be accessed with this syntax, too, ‘2’, ‘3’ etc.; positional parameters can be shifted off the stack by calling **shift[278]**. The parameter stack contains, for example, the arguments of a **call[155]**ed **define[174]**d macro, the matching groups of the [Option]al regular expression search and replace expression of **vexpr[306]**, and can be explicitly created or overwritten with the command **vpospar[307]**.

account
: (Read-only) Is set to the active **account[144]**.

add-file-recipients
: (Boolean) When file or pipe recipients have been specified, mention them in the corresponding address fields of the message instead of silently stripping them from their recipient list. By default such recipients are not mentioned.

allnet
: (Boolean) Causes only the local part to be evaluated when comparing addresses.

append
: (Boolean) Causes messages saved in the **secondary mailbox[137]** MBOX[638] to be appended to the end rather than prepended. This should always be set.

askatend
: (Boolean) Causes the prompts for `Cc:` and `Bcc:` lists to appear after the message has been edited.

askattach
: (Boolean) If set, S-nail asks an interactive user for files to attach at the end of each message; An empty line finalizes the list.

askcc (Boolean) Causes the interactive user to be prompted for carbon copy recipients (at the end of each message if *askatend[370]* or *bsdcompat[387]* are set).

askbcc (Boolean) Causes the interactive user to be prompted for blind carbon copy recipients (at the end of each message if *askatend[370]* or *bsdcompat[387]* are set).

asksend (Boolean) Causes the interactive user to be prompted for confirmation to send the message or reenter compose mode after having been shown a preliminary envelope summary.

asksign (Boolean)[Option] Causes the interactive user to be prompted if the message is to be signed at the end of each message. The *smime-sign[568]* variable is ignored when this variable is set.

asksub (Boolean) Causes S-nail to prompt the interactive user for the subject upon entering compose mode unless a subject already exists.

attrlist A sequence of characters to display in the `attribute` column of the *headline[439]* as shown in the display of **`headers`[214]**; each for one type of messages (see **Message states[13]**), with the default being `NUROSPMFAT+-$~` or `NU *HMFAT+-$~` if the *bsdflags[388]* variable is set, in the following order:

‘N’	new.
‘U’	unread but old.
‘R’	new but read.
‘O’	read and old.
‘S’	saved.
‘P’	preserved.
‘M’	mboxed.
‘F’	flagged.
‘A’	answered.
‘T’	draft.
‘+’	[v15 behaviour may differ] start of a (collapsed) thread in threaded **`sort`[282]** mode;
‘-’	[v15 behaviour may differ] an uncollapsed thread in threaded **`sort`[282]** mode; only used in conjunction with **`-L`[72]**.
‘$’	classified as spam.
‘~’	classified as possible spam.

autobcc Specifies a list of recipients to which a blind carbon copy of each outgoing message will be sent automatically.

autocc Specifies a list of recipients to which a carbon copy of each outgoing message will be sent automatically.

autocollapse
(Boolean) Causes threads to be **`collapse`[162]**d automatically when `thread`ed **`sort`[282]** mode is entered.

autoprint
(Boolean) Enable automatic **`type`[297]**ing of a(n existing) “successive” message after **`delete`[176]** and **`undelete`[177]** commands: the message that becomes the new “dot” is shown automatically, as via **`dp`[180]** or **`dt`[181]**.

autosort Causes sorted mode (see the **`sort`[282]** command) to be entered automatically with the value of this variable as sorting method when a folder is opened, for example `set autosort=thread`.

bang (Boolean) Enables the substitution of all not (reverse-solidus) escaped exclamation mark ‘!’ characters by the contents of the last executed command for the **`!`[138]** shell escape command and **`~!`[314]**, one of the compose mode **COMMAND ESCAPES[30]**. If this variable is not set no reverse solidus stripping is performed.

bind-timeout
: [Obsolete] Predecessor of *bind-inter-byte-timeout[385]*. [v15 behaviour may differ] Setting this automatically sets the successor.

bind-inter-byte-timeout
: [Option] Terminals may generate multi-byte sequences for special function keys, for example, but these sequences may not become read as a unit. Multi-byte sequences may also be defined freely via **`bind`[153]**. This variable specifies the timeout in milliseconds that the **MLE[93]** (see **On terminal control and line editor[15]**) waits for more bytes to arrive unless it considers a sequence "complete". The default is 200, the maximum is about 10 seconds. In the following example the comments state which sequences are affected by this timeout:

```
? bind base abc echo 0 # abc
? bind base ab,c echo 1 # ab
? bind base abc,d echo 2 # abc
? bind base ac,d echo 3 # ac
? bind base a,b,c echo 4
? bind base a,b,c,d echo 5
? bind base a,b,cc,dd echo 6 # cc and dd
```

bind-inter-key-timeout
: [Option] Multi-key **`bind`[153]** sequences do not time out by default. If this variable is set, then the current key sequence is forcefully terminated once the timeout (in milliseconds) triggers. The value should be (maybe significantly) larger than *bind-inter-byte-timeout[385]*, but cannot excess the maximum, too.

bsdcompat
: (Boolean) Sets some cosmetical features to traditional BSD style; has the same affect as setting *askatend[370]* and all other variables prefixed with `bsd`; it also changes the behaviour of *emptystart[418]* (which does not exist in BSD).

bsdflags
: (Boolean) Changes the letters shown in the first column of a header summary to traditional BSD style.

bsdheadline
: (Boolean) Changes the display of columns in a header summary to traditional BSD style.

bsdmsgs
: (Boolean) Changes some informational messages to traditional BSD style.

bsdorder
: (Boolean) Causes the `Subject:` field to appear immediately after the `To:` field in message headers and with the **`~h`[332] COMMAND ESCAPES[30]**.

build-cc, *build-ld*, *build-os*, *build-rest*
: (Read-only) The build environment, including the compiler, the linker, the operating system S-nail has been build for, usually taken from `uname`(1)[731] via `uname -s`, and then lowercased, as well as all the possibly interesting rest of the configuration and build environment. This information is also available in the *verbose[616]* output of the command **`version`[305]**.

charset-7bit
: The value that should appear in the `charset=` parameter of `Content-Type:` MIME header fields when no character set conversion of the message data was performed. This defaults to US-ASCII, and the chosen character set should be US-ASCII compatible.

charset-8bit
: [Option] The default 8-bit character set that is used as an implicit last member of the variable *sendcharsets[549]*. This defaults to UTF-8 if character set conversion capabilities are available, and to ISO-8859-1 otherwise (unless the operating system environment is known to always and

exclusively support UTF-8 locales), in which case the only supported character set is *ttycharset[611]* and this variable is effectively ignored.

charset-unknown-8bit
: [Option] RFC 1428 specifies conditions when internet mail gateways shall "upgrade" the content of a mail message by using a character set with the name `unknown-8bit`. Because of the unclassified nature of this character set S-nail will not be capable to convert this character set to any other character set. If this variable is set any message part which uses the character set `unknown-8bit` is assumed to really be in the character set given in the value, otherwise the (final) value of *charset-8bit[397]* is used for this purpose.

  This variable will also be taken into account if a MIME type (see **The mime.types files[37]**) of a MIME message part that uses the `binary` character set is forcefully treated as text.

cmd
: The default value for the **`pipe`[248]** command.

colour-disable
: (Boolean)[Option] Forcefully disable usage of colours. Also see the section **Coloured display[16]**.

colour-pager
: [Obsolete](Boolean)[Option] This is now an implied setting when *colour-disable[400]* is not set!

contact-mail, *contact-web*
: (Read-only) Addresses for contact per email and web, respectively, for bug reports, suggestions, or anything else regarding S-nail. The former can be used directly: '`? eval`[127] `mail`[224] $contact-mail'.

content-description-forwarded-message, *content-description-quote-attachment*, *content-description-smime-message*, *content-description-smime-signature*
: [Option](partially) Strings which will be placed in according `Content-Description:` headers if non-empty. They all have default values, for example `Forwarded message`.

crt
: If set it defines the threshold that determines how many lines of output there have to be before the `PAGER`[640] will be used for display. Usage of the`PAGER` [640] can be forced by setting this to the value '`0`', setting it without a value will deduce the current height of the terminal screen to compute the threshold (see `LINES`[631], *screen[547]* and `stty`(1)[732]). [v15 behaviour may differ] At the moment this may use the count of lines of the message in wire format, which, dependent on the *mime-encoding[479]* of the message, is unrelated to the number of display lines. (The software is old and historically the relation was a given thing.)

customhdr
: Define a set of custom headers to be injected into newly composed or forwarded messages. A custom header consists of the field name followed by a colon '`:`' and the field content body. Standard header field names cannot be overwritten by a custom header, with the exception of `Comments:` and `Keywords:`. Different to the command line option **`-C`**[61] the variable value is interpreted as a comma-separated list of custom headers: to include commas in header bodies they need to become escaped with reverse solidus '`\`'. Headers can be managed more freely in**Compose mode[8]** via `~^`[322].

  ```
  ? set customhdr='Hdr1: Body1-1\, Body1-2, Hdr2: Body2'
  ```

datefield
: Controls the appearance of the '`%d`' date and time format specification of the*headline [439]* variable, that is used, for example, when viewing the summary of **`headers`[214]**. If unset, then the local receiving date is used and displayed unformatted, otherwise the message sending `Date:`. It is possible to assign a `strftime`(3)[733] format string and control formatting, but embedding newlines via the '`%n`' format is not supported, and will result in display errors. The default is

`%Y-%m-%d %H:%M`, and also see *datefield-markout-older[411]*.

datefield-markout-older
: Only used in conjunction with *datefield[410]*. Can be used to create a visible distinction of messages dated more than a day in the future, or older than six months, a concept comparable to the **`-l`** option of the POSIX utility `ls`(1)[734]. If set to the empty string, then the plain month, day and year of the `Date:` will be displayed, but a `strftime`(3)[735] format string to control formatting can be assigned. The default is `%Y-%m-%d`.

debug
: (Boolean) (Almost) Enter a debug-only sandbox mode which generates many log messages, disables the actual delivery of messages, and also implies no*record[537]* as well as no*save[546]*. Also see *verbose[616]*.

disposition-notification-send
: (Boolean)[Option] Emit a `Disposition-Notification-To:` header (RFC 3798) with the message. This requires the*fr om[436]* variable to be set.

dot
: (Boolean) When dot is set, a period '`.`' on a line by itself during message input in (interactive or batch **`-#`[90]**) **Compose mode[8]** will be treated as end-of-message (in addition to the normal end-of-file condition). This behaviour is implied in *posix[523]* mode with a set *ignoreeof[452]*.

dotlock-disable
: (Boolean)[Option] Disable creation of **dotlock files[204]** for MBOX databases.

editalong
: If this variable is set then the editor is started automatically when a message is composed in interactive mode. If the value starts with the letter '`v`' then this acts as if**`~v`[346]**, otherwise as if **`~e`[328]** (see **COMMAND ESCAPES [30]**) had been specified. The *editheaders[417]* variable is implied for this automatically spawned editor session.

editheaders
: (Boolean) When a message is edited while being composed, its header is included in the editable text.

emptystart
: (Boolean) When entering interactive mode S-nail normally writes "No mail for user" and exits immediately if a mailbox is empty or does not exist. If this variable is set S-nail starts even with an empty or non-existent mailbox (the latter behaviour furtherly depends upon *bsdcompat[387]*, though).

errexit
: (Boolean) Let each command with a non-0 exit status, including every **`call`[155]**ed macro which **`return`[266]**s a non-0 status, cause a program exit unless prefixed by **`ignerr`[129]** (see **Command modifiers[22]**). This also affects **COMMAND ESCAPES[30]**, but which use a different modifier for ignoring the error. For more on this topic refer to the variable *?[350]*.

errors-limit
: [Option] Maximum number of entries in the **`errors`[194]** queue.

escape
: The first character of this value defines the escape character for **COMMAND ESCAPES[30]** in **Compose mode[8]**. The default value is the character tilde '`~`'. If set to the empty string, command escapes are disabled.

expandaddr
: If unset only user name and email address recipients are allowed **On sending mail, and non-interactive mode[7]**. If set without value all possible recipient types will be accepted. A value is parsed as a comma-separated list of case-insensitive strings, and if that contains `restrict` behaviour equals the former except when in interactive mode or if **COMMAND ESCAPES[30]** were enabled via **`-~`** [89] or **`-#`[90]**, in which case it equals the latter, allowing all address types.

`restrict` really acts like `restrict,-all,+name,+addr`, so care for ordering issues must be taken.

Recipient types can be added and removed with a plus sign '`+`' or hyphen-minus '`-`' prefix, respectively. By default invalid or disallowed types are filtered out and cause a warning, hard send errors need to be enforced by including `fail`. The value `all` covers all types, `fcc` whitelists `Fcc:` header targets regardless of other settings, `file` file targets (it includes `fcc`), `pipe` command pipeline targets, `name` user names still unexpanded after **`alias`**[147] and *mta-aliases*[483] processing and thus left for expansion by the *mta*[482] (invalid for the built-in SMTP one), and `addr` network addresses. Targets are interpreted in the given order, so that `restrict,fail,+file,-all,+addr` will cause hard errors for any non-network address recipient address unless running interactively or having been started with the option **`-~`** [89] or **`-#`**[90]; in the latter case(s) any type may be used.

Plain user name recipients addressing valid local users can be expanded to fully qualified network addresses (also see *hostname*[447]) by including `nametoaddr` in the list. Historically invalid recipients were stripped off without causing errors, this can be changed by making `failinvaddr` an entry of the list (it really acts like `failinvaddr,+addr`). Likewise, `domaincheck` (really `domaincheck,+addr`) compares address domain names against a whitelist and strips off (`fail` for hard errors) recipients which fail this test; the domain name `localhost` and the non-empty value of *hostname*[447] (the real hostname otherwise) are always whitelisted, *expandaddr-domaincheck*[423] can be set to extend this list. Finally some address providers (for example **`-b`**[60], **`-c`**[62] and all other command line recipients) will be evaluated as if specified within dollar-single-quotes (see **Shell-style argument quoting**[24]) if the value list contains the string `shquote`.

expandaddr-domaincheck
: Can be set to a comma-separated list of domain names which should be whitelisted for the evaluation of the `domaincheck` mode of *expandaddr*[422]. IDNA encoding is not automatically performed, **`addrcodec`**[146] can be used to prepare the domain (of an address).

expandargv
: Unless this variable is set additional *mta*[482] arguments from the command line, as can be given after a **`--`** separator, results in a program termination with failure status. The same can be accomplished by using the special (case-insensitive) value `fail`. A lesser strict variant is the otherwise identical `restrict`, which does accept such arguments in interactive mode, or if tilde commands were enabled explicitly by using one of the command line options **`-~`** [89] or **`-#`**[90]. The empty value will allow unconditional usage.

features
: (Read-only) String giving a list of optional features. Features are preceded with a plus sign '`+`' if they are available, with a hyphen-minus '`-`' otherwise. To ease substring matching the string starts and ends with a comma. The output of the command **`version`**[305] includes this information in a more pleasant output.

flipr
: (Boolean) This setting reverses the meanings of a set of reply commands, turning the lowercase variants, which by default address all recipients included in the header of a message (**`reply`**[261], **`respond`**[262], **`followup`**[207]) into the uppercase variants, which by default address the sender only (**`Reply`**[259], **`Respond`**[260], **`Followup`**[206]) and vice versa.

folder
: The default path under which mailboxes are to be saved: filenames that begin with the plus sign '`+`' will have the plus sign replaced with the value of this variable if set, otherwise the plus sign will remain unchanged when doing **Filename transformations**[28]; also see **`folder`**[203] for more on this topic, and know about standard imposed implications of *outfolder*[493]. The value supports a subset of transformations itself, and if the non-empty value does not start with a solidus

'/', then the value of HOME[627] will be prefixed automatically. Once the actual value is evaluated first, the internal variable *folder-resolved[428]* will be updated for caching purposes.

folder-hook-FOLDER, *folder-hook*
: [Obsolete] Predecessor of both of *on-mailbox-newmail[502]* (with a different **local[130]** scope!) and *on-mailbox-open[501]*.

folder-resolved
: (Read-only) Set to the fully resolved path of *folder[427]* once that evaluation has occurred; rather internal.

followup-to
: (Boolean) Controls whether a `Mail-Followup-To:` header is generated when sending messages to known mailing lists. The user as determined via *from[436]* (or, if that contains multiple addresses, *sender[551]*) will be placed in there if any list recipient is not a subscribed list. Also see *followup-to-honour[431]* and the commands **mlist[230]**, **mlsubscribe[232]**, **reply[261]** and **Lreply[222]**.

followup-to-add-cc
: (Boolean) Controls whether the user will be added to the messages' `Cc:` list in addition to placing an entry in `Mail-Followup-To:` (see *followup-to[429]*).

followup-to-honour
: Controls whether a `Mail-Followup-To:` header is honoured when group-replying to a message via **reply[261]** or **Lreply[222]**. This is a**quadoption [349]**; if set without a value it defaults to "yes", and see *followup-to[429]*.

forward-add-cc
: (Boolean) Whether senders of messages forwarded via **~F[329]**, **~f[330]**, **~m[336]**, **~U[344]** or **~u[345]** shall be made members of the carbon copies `Cc:` list.

forward-as-attachment
: (Boolean) Original messages are normally sent as inline text with the **forward[210]** command, and only the first part of a multipart message is included. With this setting enabled messages are sent as unmodified MIME `message/rfc822` attachments with all of their parts included.

forward-inject-head, *forward-inject-tail*
: The strings to put before and after the text of a message with the **forward[210]** command, respectively. The former defaults to `-------- Original Message --------\n`. Special format directives in these strings will be expanded if possible, and if so configured the output will be folded according to *quote-fold[532]*; for more refer to *quote-inject-head[533]*. Injections will not be performed by **forward[210]** if the variable *forward-as-attachment[433]* is set — the **COMMAND ESCAPES[30]** **~F[329]**, **~f[330]**, **~M[335]**, **~m[336]**, **~U[344]**, **~u[345]** always inject.

from
: The address, or a list of addresses to put into the `From:` field of the message header, quoting RFC 5322: the author(s) of the message, that is, the mailbox(es) of the person(s) or system(s) responsible for the writing of the message. If multiple addresses are used specifying a `Sender:` is required according to that RFC, which can be done via *sender[551]*. [v15 behaviour may differ] Expect automatic management of the *from[436]* and *sender[551]* relationship (requiring an address order in the former). Dependent on the context these addresses are handled as if they were in the list of **alternates[149]**.

  If a file-based MTA is used, then *from* (or, if that contains multiple addresses, *sender[551]*) can nonetheless be used as the envelope sender address at the MTA protocol level (the RFC 5321 reverse-path), either via the **-r** [79] command line option (without argument; see there for more), or by setting *r-option-implicit[535]*.

If the machine's hostname is not valid at the Internet (for example at a dialup machine) either this or *hostname[447]* have to be set (a SMTP-based *mta[482]* adds onto this *smtp-from[574]*): if so the message and MIME part related unique ID fields `Message-ID:` and `Content-ID:` will be created (except when disallowed by *message-id-disable[472]* or *stealthmua[590]*).

fullnames
(Boolean) Due to historical reasons comments and name parts of email addresses are removed by default when sending mail, replying to or forwarding a message. If this variable is set such stripping is not performed.

header (Boolean) Causes the header summary to be written at startup and after commands that affect the number of messages or the order of messages in the current **`folder`**[203]. Unless in *posix[523]* mode a header summary will also be displayed on folder changes. The command line option **`-N`**[75] can be used to set no*header[438]*.

headline A format string to use for the summary of **`headers`**[214]. Format specifiers in the given string start with a percent sign '`%`' and may be followed by an optional decimal number indicating the field width — if that is negative, the field is to be left-aligned. Names and addresses are subject to modifications according to *showname[554]* and *showto[555]*. Valid format specifiers are:

- '`%%`' A plain percent sign.
- '`%>`' "Dotmark": a space character but for the current message ("dot"), for which it expands to '`>`' (dependent on*headline-plain [441]*).
- '`%<`' "Dotmark": a space character but for the current message ("dot"), for which it expands to '`<`' (dependent on*headline-plain [441]*).
- '`%$`' [Option] The spam score of the message, as has been classified via the command **`spamrate`**[289]. Shows only a replacement character if there is no spam support.
- '`%a`' Message attribute character (status flag); the actual content can be adjusted by setting *attrlist[377]*.
- '`%d`' The date found in the `Date:` header of the message when *datefield[410]* is set (the default), otherwise the date when the message was received. Formatting can be controlled by assigning a `strftime`(3)[736] format string to *datefield[410]* (and *datefield-markout-older[411]*).
- '`%e`' The indenting level in threaded **`sort`**[282] mode.
- '`%f`' The address of the message sender.
- '`%i`' The message thread tree structure. (Note that this format does not support a field width, and honours *headline-plain[441]*.)
- '`%L`' Mailing list status: is the recipient of the message a known '`l`' (**`mlist`**[230]) or '`L`' **`mlsubscribe`**[232]d mailing list? The letter '`P`' announces the presence of a RFC 2369 `List-Post:` header, which makes a message a valuable target of **`Lreply`**[222].
- '`%l`' The number of lines of the message, if available.
- '`%m`' Message number.
- '`%o`' The number of octets (bytes) in the message, if available.
- '`%S`' Message subject (if any) in double quotes.
- '`%s`' Message subject (if any).
- '`%t`' The position in sorted order.
- '`%U`' The value 0 except in an IMAP mailbox, where it expands to the UID of the message.

The default is `%>%a%m %-18f %16d %4l/%-5o %i%-s`, or `%>%a%m %20-f %16d %3l/%-5o %i%-S` if *bsdcompat[387]* is set. Also see *attrlist[377]*, *headline-plain[441]* and *headline-bidi[440]*.

headline-bidi

Bidirectional text requires special treatment when displaying headers, because numbers (in dates or for file sizes etc.) will not affect the current text direction, in effect resulting in ugly line layouts when arabic or other right-to-left text is to be displayed. On the other hand only a minority of terminals is capable to correctly handle direction changes, so that user interaction is necessary for acceptable results. Note that extended host system support is required nonetheless, e.g., detection of the terminal character set is one precondition; and this feature only works in an Unicode (i.e., UTF-8) locale.

In general setting this variable will cause S-nail to encapsulate text fields that may occur when displaying *headline[439]* (and some other fields, like dynamic expansions in *prompt[525]*) with special Unicode control sequences; it is possible to fine-tune the terminal support level by assigning a value: no value (or any value other than '1', '2' and '3') will make S-nail assume that the terminal is capable to properly deal with Unicode version 6.3, in which case text is embedded in a pair of U+2068 (FIRST STRONG ISOLATE) and U+2069 (POP DIRECTIONAL ISOLATE) characters. In addition no space on the line is reserved for these characters.

Weaker support is chosen by using the value '1' (Unicode 6.3, but reserve the room of two spaces for writing the control sequences onto the line). The values '2' and '3' select Unicode 1.1 support (U+200E, LEFT-TO-RIGHT MARK); the latter again reserves room for two spaces in addition.

headline-plain

(Boolean) On Unicode (UTF-8) aware terminals enhanced graphical symbols are used by default for certain entries of *headline[439]*. If this variable is set only basic US-ASCII symbols will be used.

history-file

[Option] The (expandable) location of a permanent **history[216]** file for the **MLE[93]** line editor (**On terminal control and line editor[15]**). Also see*history-size [445]*.

history-gabby

[Option] Add more entries to the MLE **history[216]** as is normally done. A comma-separated list of case-insensitive strings can be used to fine-tune which gabby entries shall be allowed. If it contains `errors`, erroneous commands will also be added. `all` adds all optional entries, and is the fallback chattiness identifier of *on-history-addition[500]*.

history-gabby-persist

(Boolean)[Option] The *history-gabby[443]* entries will not be saved in persistent storage unless this variable is set. The knowledge of whether a persistent entry was gabby is not lost. Also see *history-file[442]*.

history-size

[Option] Setting this variable imposes a limit on the number of concurrent **history[216]** entries. If set to the value 0 then no further history entries will be added, and loading and incorporation of the *history-file[442]* upon program startup can also be suppressed by doing this. Runtime changes will not be reflected before the **history[216]** is saved or loaded (again).

hold (Boolean) This setting controls whether messages are held in the system *inbox[453]*, and it is set by default.

hostname

Used instead of the value obtained from `uname`(3)[737] and `getaddrinfo`(3)[738] as the hostname when expanding local addresses, for example in `From:` (also see **On sending mail, and non-interactive mode[7]**, for expansion of addresses that have a valid user-, but no domain name in angle brackets). If either of*fr om[436]* or this variable is set the message and MIME part related unique ID fields `Message-ID:` and `Content-ID:` will be created (except when disallowed by

message-id-disable[472] or *stealthmua[590]*). If the [Option]al IDNA support is available (see *idna-disable[448]*) variable assignment is aborted when a necessary conversion fails.

Setting it to the empty string will cause the normal hostname to be used, but nonetheless enables creation of said ID fields. One should produce some test messages with the desired combination of *hostname*, and/or *from[436]*, *sender[551]*, *smtp-from[574]* etc. first.

idna-disable
(Boolean)[Option] Can be used to turn off the automatic conversion of domain names according to the rules of IDNA (internationalized domain names for applications). Since the IDNA code assumes that domain names are specified with the *ttycharset[611]* character set, an UTF-8 locale charset is required to represent all possible international domain names (before conversion, that is).

ifs The input field separator that is used ([v15 behaviour may differ] by some functions) to determine where to split input data.

1. Unsetting is treated as assigning the default value, `\t\n`.
2. If set to the empty value, no field splitting will be performed.
3. If set to a non-empty value, all whitespace characters are extracted and assigned to the variable *ifs-ws[450]*.

a. *ifs-ws* will be ignored at the beginning and end of input. Diverging from POSIX shells default whitespace is removed in addition, which is owed to the entirely different line content extraction rules.
b. Each occurrence of a character of *ifs* will cause field-splitting, any adjacent *ifs-ws* characters will be skipped.

ifs-ws (Read-only) Automatically deduced from the whitespace characters in *ifs[449]*.

ignore (Boolean) Ignore interrupt signals from the terminal while entering messages; instead echo them as '@' characters and discard the current line.

ignoreeof
(Boolean) Ignore end-of-file conditions (`control-D`) in **Compose mode[8]** on message input and in interactive command input. If set an interactive command input session can only be left by explicitly using one of the commands **`exit`[195]** and **`quit`[252]**, and message input in compose mode can only be terminated by entering a period '`.`' on a line by itself or by using the **`~.`[315] COMMAND ESCAPES[30]**; Setting this implies the behaviour that *dot[414]* describes in *posix[523]* mode.

inbox If this is set to a non-empty string it will specify the user's **primary system mailbox[136]**, overriding `MAIL`[634] and the system-dependent default, and (thus) be used to replace '%' when doing **Filename transformations[28]**; also see **`folder`[203]** for more on this topic. The value supports a subset of transformations itself.

indentprefix
String used by the **`~m`[336]**, **`~M`[335]** and **`~R`[340] COMMAND ESCAPES[30]** and by the *quote[528]* option for indenting messages, in place of the POSIX mandated default tabulator character '`\t`'. Also see*quote-c hars[531]*.

keep (Boolean) If set, an empty **primary system mailbox[136]** file is not removed. Note that, in conjunction with *posix[523]* mode any empty file will be removed unless this variable is set. This may improve the interoperability with other mail user agents when using a common folder directory, and prevents malicious users from creating fake mailboxes in a world-writable spool directory. [v15 behaviour may differ] Only local regular (MBOX) files are covered, Maildir and other mailbox types will never be removed, even if empty.

keep-content-length
(Boolean) When (editing messages and) writing MBOX mailbox files S-nail can be told to keep the `Content-Length:` and `Lines:` header fields that some MUAs generate by setting this variable. Since S-nail does neither use nor update these non-standardized header fields (which in itself shows one of their conceptual problems), stripping them should increase interoperability in between MUAs that work with with same mailbox files. Note that, if this is not set but *writebackedited[623]*, as below, is, a possibly performed automatic stripping of these header fields already marks the message as being modified. [v15 behaviour may differ] At some future time S-nail will be capable to rewrite and apply an *mime-encoding[479]* to modified messages, and then those fields will be stripped silently.

keepsave (Boolean) When a message is saved it is usually discarded from the originating folder when S-nail is quit. This setting causes all saved message to be retained.

line-editor-config
[Option] Some aspects of the **MLE[93]** are dynamically adjustable. Interpreted as a comma-separated list of case-insensitive keywords. `quote-rndtrip` denotes the default setting of **`mle-quote-rndtrip`[108]**. `srch-case` defines whether **`mle-hist-srch-bwd`[109]** and **`mle-hist-srch-fwd`[110]** match case-insensitively, `srch-any` whether they match any substring or only at the beginning of lines, and [Option]ally `srch-regex` will instead search through history based on a regular expression. The cursor is placed at the end of the expanded history search entry, with `srch-pos0` only if it fits on the line: like this the command name is always visible.

line-editor-cpl-word-breaks
[Option] List of bytes which are used by the **`mle-complete`[102]** tabulator completion to decide where word boundaries exist, by default `"'@=;|:` [v15 behaviour may differ] This mechanism is yet restricted.

line-editor-disable
(Boolean) Turn off any line editing capabilities (from S-nails POW, see **On terminal control and line editor[15]** for more).

line-editor-no-defaults
(Boolean)[Option] Do not establish any default key binding.

log-prefix
Error log message prefix string (`s-nail: `).

mailbox-basename
(Read-only) The “last component” of the name of the current mailbox (**`folder`[203]**).

mailbox-display
(Read-only) The name of the current mailbox (**`folder`[203]**), possibly abbreviated for display purposes.

mailbox-resolved
(Read-only) The fully resolved path of the current mailbox.

mailcap-disable
(Boolean)[Option] Turn off consideration of MIME type handlers from, and implicit loading of **The Mailcap files[38]**.

mailx-extra-rc
An additional startup file that is loaded as the last of the **Resource files[36]**. Use this file for commands that are not understood by other POSIX `mailx`(1)[739] implementations, i.e., mostly anything which is not covered by **Initial settings[32]**.

markanswered
(Boolean) When a message is replied to and this variable is set, it is marked as having been **answered[151]**. See the section**Message states [13]**.

mbox-fcc-and-pcc
(Boolean) By default all file and pipe message recipients (see *expandaddr[422]*) will be fed valid MBOX database entry message data (see **folder[203]**, *mbox-rfc4155[470]*), and existing file targets will become extended in compliance to RFC 4155. If this variable is unset then a plain standalone RFC 5322 message will be written, and existing file targets will be overwritten.

mbox-rfc4155
(Boolean) When opening MBOX mailbox databases, and in order to achieve compatibility with old software, the very tolerant POSIX standard rules for detecting message boundaries (so-called `From_` lines) are used instead of the stricter rules from the standard RFC 4155. This behaviour can be switched by setting this variable.

This may temporarily be handy when S-nail complains about invalid `From_` lines when opening a MBOX: in this case setting this variable and re-opening the mailbox in question may correct the result. If so, copying the entire mailbox to some other file, as in `copy * SOME-FILE`, will perform proper, all-compatible `From_` quoting for all detected messages, resulting in a valid MBOX mailbox. ([v15 behaviour may differ] The better and non-destructive approach is to re-encode invalid messages, as if it would be created anew, instead of mangling the `From_` lines; this requires the structural code changes of the v15 rewrite.) Finally the variable can be unset again:

```
? define mboxfix {
  local set mbox-rfc4155; File "${1}"; copy * "${2}"
}
? call mboxfix /tmp/bad.mbox /tmp/good.mbox
```

memdebug
(Boolean) Internal development variable. Auto-enabled if*deb ug[412]* is set, but can individually be disabled.

message-id-disable
(Boolean) By setting this variable the generation of `Message-ID:` and `Content-ID:` message and MIME part headers can be completely suppressed, effectively leaving this task up to the *mta[482]*. Note that according to RFC 5321 a SMTP server is not required to add this field by itself, so it should be ensured that it accepts messages without `Message-ID`.

message-inject-head
A string to put at the beginning of each new message, followed by a newline. [Obsolete] The escape sequences tabulator '`\t`' and newline '`\n`' are understood: expand when **set[271]**ing instead (*v15-compat[615]*, **wysh[134]**).

message-inject-tail
A string to put at the end of each new message, followed by a newline. [Obsolete] The escape sequences tabulator '`\t`' and newline '`\n`' are understood: expand when **set[271]**ing instead (*v15-compat[615]*, **wysh[134]**). Also see*on-compose-leave [497]*.

```
? set sign=$'\n  Tony'  Sign=$'\n  Tony\n  I am fixing things!'
? set message-inject-tail=${Sign}
```

metoo (Boolean) Usually, when an **alias[147]** expansion contains the sender, the sender is removed from the expansion. Setting this option suppresses these removals. Note that a set*metoo [475]* also causes a '`-m`' option to be passed through to the*mta [482]*; though most of the modern MTAs no longer document this flag, no MTA is known which does not support it (for historical compatibility).

mime-allow-text-controls

(Boolean) When sending messages, each part of the message is MIME-inspected in order to classify the `Content-Type:` and `Content-Transfer-Encoding:` (see *mime-encoding[479]*) that is required to send this part over mail transport, i.e., a computation rather similar to what the `file`(1)[740] command produces when used with the `--mime` option.

This classification however treats text files which are encoded in UTF-16 (seen for HTML files) and similar character sets as binary octet-streams, forcefully changing any `text/plain` or `text/html` specification to `application/octet-stream`: If that actually happens a yet unset charset MIME parameter is set to `binary`, effectively making it impossible for the receiving MUA to automatically interpret the contents of the part.

If this variable is set, and the data was unambiguously identified as text data at first glance (by a `.txt` or `.html` file extension), then the original `Content-Type:` will not be overwritten.

mime-alternative-favour-rich

(Boolean) If this variable is set then rich MIME alternative parts (e.g., HTML) will be preferred in favour of included plain text versions when displaying messages, provided that a handler exists which produces output that can be (re)integrated into S-nail's normal visual display.

mime-counter-evidence

Normally the `Content-Type:` field is used to decide how to handle MIME parts. Some MUAs, however, do not use **The mime.types files[37]** (also see **HTML mail and MIME attachments[10]**) or a similar mechanism to correctly classify content, but specify an unspecific MIME type (`application/octet-stream`) even for plain text attachments. If this variable is set then S-nail will try to re-classify such MIME message parts, if possible, for example via a possibly existing attachment filename. A non-empty value may also be given, in which case a number is expected, actually a carrier of bits, best specified as a binary value, like `mime-counter-evidence=0b1110`.

- If bit two is set (counting from 1, decimal 2) then the detected **`mimetype`[227]** will be carried along with the message and be used for deciding which MIME handler is to be used, for example; when displaying such a MIME part the part-info will indicate the overridden content-type by showing a plus sign '`+`'.
- If bit three is set (decimal 4) then the counter-evidence is always produced and a positive result will be used as the MIME type, even forcefully overriding the parts given MIME type.
- If bit four is set (decimal 8) as a last resort the actual content of `application/octet-stream` parts will be inspected, so that data which looks like plain text can be treated as such. This mode is even more relaxed when data is to be displayed to the user or used as a message quote (data consumers which mangle data for display purposes, which includes masking of control characters, for example).

mime-encoding

The MIME `Content-Transfer-Encoding` to use in outgoing text messages and message parts, where applicable (7-bit clean text messages are without an encoding if possible):

`8bit` (Or '`8b`'.) 8-bit transport effectively causes the raw data be passed through unchanged, but may cause problems when transferring mail messages over channels that are not ESMTP (RFC 1869) compliant. Also, several input data constructs are not allowed by the specifications and may cause a different transfer-encoding to be used. By established rules and popular demand occurrences of `^From_` (see *mbox-rfc4155[470]*) will be MBOXO quoted (prefixed with greater-than sign '`>`') instead of causing a non-destructive encoding like `quoted-printable` to be chosen, unless context (like message signing) requires otherwise.

`quoted-printable`
: (Or ‘`qp`’.) Quoted-printable encoding is 7-bit clean and has the property that ASCII characters are passed through unchanged, so that an english message can be read as-is; it is also acceptable for other single-byte locales that share many characters with ASCII, for example ISO-8859-1. The encoding will cause a large overhead for messages in other character sets: for example it will require up to twelve (12) bytes to encode a single UTF-8 character of four (4) bytes. It is the default encoding.

`base64` (Or `b64`.) This encoding is 7-bit clean and will always be used for binary data. This encoding has a constant input:output ratio of 3:4, regardless of the character set of the input data it will encode three bytes of input to four bytes of output. This transfer-encoding is not human readable without performing a decoding step.

mime-force-sendout
: (Boolean)[Option] Whenever it is not acceptable to fail sending out messages because of non-convertible character content this variable may be set. It will, as a last resort, classify the part content as `application/octet-stream`. Refer to the section**Character sets [12]** for the complete picture of character set conversion, and **HTML mail and MIME attachments[10]** for how to internally or externally handle part content.

mimetypes-load-control
: Can be used to control which of **The mime.types files[37]** are loaded: if the letter ‘`u`’ is part of the option value, then the user’s personal `~/.mime.types[654]` file will be loaded (if it exists); likewise the letter ‘`s`’ controls loading of the system-wide`/etc/mime.types [655]`; directives found in the user file take precedence, letter matching is case-insensitive. If this variable is not set S-nail will try to load both files. Incorporation of the S-nail-built-in MIME types cannot be suppressed, but they will be matched last (the order can be listed via **`mimetype[227]`**).

  More sources can be specified by using a different syntax: if the value string contains an equals sign ‘`=`’ then it is instead parsed as a comma-separated list of the described letters plus `f=FILENAME` pairs; the given filenames will be expanded and loaded, and their content may use the extended syntax that is described in the section **The mime.types files[37]**. Directives found in such files always take precedence (are prepended to the MIME type cache).

mta
: Select an alternate Message-Transfer-Agent by either specifying the full pathname of an executable (a `file://` prefix may be given), or [Option]ally a SMTP aka SUBMISSION protocol URL:

  ```
  submissions://[user[:password]@]server[:port]
  ```

  The default has been chosen at compile time. MTA data transfers are always performed in asynchronous child processes, and without supervision unless either the *sendwait[552]* or the *verbose[616]* variable is set. Also see *mta-bcc-ok[488]*. [Option]ally expansion of `aliases`(5)[741] can be performed by setting *mta-aliases[483]*.

  For testing purposes there is the `test` pseudo-MTA, which dumps to standard output or optionally to a file, and honours *mbox-fcc-and-pcc[469]*:

  ```
  $ echo text | s-nail -:/ -Smta=test -s ubject ex@am.ple
  $ </dev/null s-nail -:/ -Smta=test://./xy ex@am.ple
  ```

  For a file-based MTA it may be necessary to set *mta-argv0[487]* in in order to choose the right target of a modern `mailwrapper`(8)[742] environment. It will be passed command line arguments from several possible sources: from the variable *mta-arguments[484]* if set, from the command line if given and the variable *expandargv[424]* allows their use. Argument processing of the MTA will be terminated with a **`--`** separator.

The otherwise occurring implicit usage of the following MTA command line arguments can be disabled by setting the boolean variable *mta-no-default-arguments[485]* (which will also disable passing **--** to the MTA): **-i** (for not treating a line with only a dot '**.**' character as the end of input), **-m** (shall the variable *metoo[475]* be set) and **-v** (if the *verbose[616]* variable is set); in conjunction with the **-r** [79] command line option or *r-option-implicit[535]* **-f**as well as possibly **-F** will (not) be passed.

[Option]ally S-nail can send mail over SMTP aka SUBMISSION network connections to a single defined smart host by setting this variable to a corresponding URL (see **On URL syntax and credential lookup[18]**). Server interaction (TLS, authentication type, etc.) is configurable via *smtp-config[573]*. An overview on TLS and links to more information can be found under **Encrypted network communication[19]**. Note that with some mail providers it may be necessary to set the *smtp-from[574]* variable in order to use a specific combination of *from[436]*, *hostname[447]* and *mta[482]*. Network communication socket timeouts are configurable via *socket-connect-timeout[577]*. All generated network traffic may be proxied over a SOCKS *socks-proxy[578]*, it can be logged by setting *verbose[616]* twice. The following SMTP variants may be used:

- The plain SMTP protocol (RFC 5321) that normally lives on the server port 25, which will [Option]ally be upgraded to a TLS encrypted session unless disallowed by *smtp-config[573]*. Assign a value like `smtp://[user[:password]@]server[:port]` to choose this protocol.
- [Option] The so-called SMTPS which is supposed to live on server port 465 and is automatically TLS secured. Unfortunately it never became a standardized protocol and may thus not be supported by your hosts network service database – in fact the port number has already been reassigned to other protocols!

  SMTPS is nonetheless a commonly offered protocol and thus can be chosen by assigning a value like `smtps://[user[:password]@]server[:port]`; due to the mentioned problems it is usually necessary to explicitly specify the port as `:465`, however.
- The SUBMISSION protocol (RFC 6409) lives on server port 587 and shares the semantics with SMTP from S-nail's point of view: `submission://[user[:password]@]server[:port]`.
- [Option] The SUBMISSIONS protocol (RFC 8314) that lives on server port 465 and is TLS secured by default. It can be chosen by assigning a value like `submissions://[user[:password]@]server[:port]`. Due to the problems mentioned for SMTPS above and the fact that SUBMISSIONS is new and a successor that lives on the same port as the historical engineering mismanagement named SMTPS, it is usually necessary to explicitly specify the port as `:465`.

mta-aliases

[Option] If set to a path pointing to a text file in valid MTA (Postfix) `aliases`(5)[743] format, the file is loaded and cached (manageable with**mtaaliases [238]**), and henceforth plain `name` (see *expandaddr[422]*) message recipient names are recursively expanded as a last expansion step, after the distribution lists which can be created with **alias[147]**. Constraints on `aliases`(5)[744] content support: only local addresses (names) which are valid usernames (`[a-z_][a-z0-9_-]*[$]?`) are treated as expandable aliases, and [v15 behaviour may differ] `:include:/file/name` directives are not supported. By including `-name` in *expandaddr[422]* it can be asserted that only expanded names (mail addresses) are passed through to the MTA.

mta-arguments
: Arguments to pass through to a file-based *mta[482]*, parsed according to **Shell-style argument quoting[24]** into an array of arguments which will be joined onto MTA options from other sources, for example `? set mta-arguments='-t -X "/tmp/my log"'`.

mta-no-default-arguments
: (Boolean) Avoids passing standard command line options to a file-based *mta[482]* (see there).

mta-no-recipient-arguments
: (Boolean) By default all recipient addresses will be passed as command line options to a file-based *mta[482]*. Setting this variable disables this behaviour to aid those MTAs which employ special treatment of such arguments. Doing so can make it necessary to pass a **`-t`** via *mta-arguments[484]*, to testify the MTA that it should use the passed message as a template.

mta-argv0
: Many systems use a so-called `mailwrapper`(8)[745] environment to ensure compatibility with `sendmail`(1)[746]. This works by inspecting the name that was used to invoke the mail delivery system. If this variable is set then the mailwrapper (the program that is actually executed when calling the file-based *mta[482]*) will treat its contents as that name.

mta-bcc-ok
: (Boolean) In violation of RFC 5322 some MTAs do not remove `Bcc:` header lines from transported messages after having noted the respective recipients for addressing purposes. (The MTAs Exim and Courier for example require the command line option **`-t`** to enforce removal.) Unless this is set corresponding recipients are addressed by protocol-specific means or MTA command line options only, the header itself is stripped before being sent over the wire.

netrc-lookup-USER@HOST, *netrc-lookup-HOST*, *netrc-lookup*
: (Boolean)[Option] Used to control usage of the user's `~/.netrc[656]` file for lookup of account credentials, as documented in the section **On URL syntax and credential lookup[18]** and for the command **`netrc[239]`**; the section **The .netrc file[39]** documents the file format. Also see *netrc-pipe[490]*.

netrc-pipe
: [Option] When `~/.netrc[656]` is loaded (see **`netrc[239]`** and *netrc-lookup[489]*) then S-nail will read the output of a shell pipe instead of the user's `~/.netrc[656]` file if this variable is set (to the desired shell command). This can be used to, for example, store `~/.netrc[656]` in encrypted form: `? set netrc-pipe='gpg -qd ~/.netrc.pgp'`.

newfolders
: [Option] If this variable has the value `maildir`, newly created local folders will be in Maildir instead of MBOX format.

newmail
: Checks for new mail in the current folder each time the prompt is shown. A Maildir folder must be re-scanned to determine if new mail has arrived. If this variable is set to the special value `nopoll` then a Maildir folder will not be rescanned completely, but only timestamp changes are detected. Maildir folders are [Option]al.

outfolder
: (Boolean) Causes a non-absolute filename specified in *record[537]*, as well as the sender-based filenames of the **`Copy[168]`**, **`Save[267]`**, **`Followup[206]`** and **`followup[207]`** commands to be interpreted relative to the *folder[427]* directory rather than relative to the current directory.

on-account-cleanup-ACCOUNT, *on-account-cleanup*
: Macro hook which will be called once an **`account[144]`** is left, as the very last step before unrolling the per-account scope **`local[130]`**. This hook is run even in case of fatal errors,

including those generated by switching to the account as such, and it is advisable to perform only absolutely necessary actions, like cleaning up **alternates**[149], for example. The specialized form is used in favour of the generic one if found.

on-compose-cleanup
: Macro hook which will be called after the message has been sent (or not, in case of failures), as the very last step before unrolling compose mode **local** scope. This hook is run even in case of fatal errors, and it is advisable to perform only absolutely necessary actions, like cleaning up **alternates**[149], for example.

  For compose mode hooks that may affect the message content see *on-compose-enter*[496], *on-compose-leave*[497], *on-compose-splice*[498]. [v15 behaviour may differ] This hook exists because **alias**[147], **alternates**[149], **commandalias**[166], **shortcut**[276], to name a few, are not yet covered by **local**[130]: changes applied in compose mode will continue to be in effect thereafter.

on-compose-enter, *on-compose-leave*
: Macro hooks which will be called once compose mode is entered, and after composing has been finished, respectively; the exact order of the steps taken is documented for **~.**[315], one of the **COMMAND ESCAPES**[30]. Context about the message being worked on can be queried via **digmsg**[178]. *on-compose-cleanup*[495] can be used to perform other necessary cleanup steps.

  Here is an example that injects a signature (also see *Sign*[556], *sign*[557] as well as **~A**[323], **~a**[324]) via *message-inject-tail*[474]; instead using *on-compose-splice*[498] to simply inject the file of desire via **~<**[317] or **~<!**[318] may be a better approach.

```
define t_ocl {
  vput ! i cat ~/.mysig
  if $? -eq 0
     vput csop message-inject-tail trim-end $i
  end

  # Alternatively
  readctl create ~/.mysig
  if $? -eq 0
    readall i
    if $? -eq 0
      vput csop message-inject-tail trim-end $i
    end
    readctl remove ~/.mysig
  end
}
set on-compose-leave=t_ocl
```

on-compose-splice, *on-compose-splice-shell*
: These hooks run after compose mode is finished, but before *on-compose-leave*[497]. Both are executed in a subprocess, with their input and output connected such that they can act like an interactive user: S-nail's output they **read**[253], input for S-nail they can **echo**[185]. Whereas the latter is a SHELL[643] command, the former is a **define**[174]d macro that is evaluated in a restricted mode with only a small set of commands available (the *verbose*[616] output of for example **list**[220] indicates the subprocess capability).

  In the subprocess (a restricted set of) **COMMAND ESCAPES**[30] will always be available. For guaranteed reproducibilities' sake *escape*[421] and *ifs*[449] are set to their defaults. The escape **~^**[322] has been especially designed for scriptability: the first line these hooks will read on

standard input is the escape's protocol version ("0 0 2"), backward incompatible protocol changes have to be expected.

Care must be taken to avoid deadlocks and other false control flow: if subprocess and S-nail both wait for more input, or if one does not expect more input, whereas the other waits for consumption of its output. Hooks are not automatically synchronized: if a hook emits '`~x`' to cause S-nail to exit compose mode, the subprocess will keep running nonetheless until the macro is completely worked or **xit**[311] is called explicitly. They will however receive a termination signal if the parent enters an error condition. [v15 behaviour may differ] Protection against and interaction with signals is not yet given; it is likely that in the future these scripts will be placed in an isolated session, which is signalled in its entirety as necessary.

```
define ocs_signature {
  read proto_version
  echo '~< ~/.mysig' # '~<! fortune pathtofortunefile'
}
set on-compose-splice=ocs_signature

set on-compose-splice-shell=$'\
  read proto_version;\
  printf "hello $version!  Headers: ";\
  echo \'~^header list\';\
  read status result;\
  echo "status=$status result=$result";\
  '

define ocsm {
  read proto_version
  echo Splice protocol version is $proto_version
  echo '~^h l'; read hl; vput csop es subs "${hl}" 0 1
  if "$es" != 2
    echoerr 'Cannot read header list'; echo '~x'; xit
  endif
  if "$hl" !%?case ' cc'
    echo '~^h i cc "Diet is your <mirr.or>"'; read es;\
      vput csop es substring "${es}" 0 1
    if "$es" != 2
      echoerr 'Cannot insert Cc: header'; echo '~x'
      # (no xit, macro finishes anyway)
    endif
  endif
}
set on-compose-splice=ocsm
```

on-history-addition

This hook will be called if an entry is about to be added to the **history**[216] of the **MLE**[93], as documented in **On terminal control and line editor**[15]. It will be called with three arguments: the first is the name of the input context (see **bind**[153]), the second is either an empty string or the matching *history-gabby*[443] type, and the third being the complete command line to be added. The entry will not be added to history if the hook uses a non-0 **return**[266]. [v15 behaviour may differ] A future version will give the expanded command name as the third argument, followed by the tokenized command line as parsed in the remaining arguments, the first of which is the original unexpanded command name; i.e., one may do **shift**[278] 4 and will then

be able to access the positional parameters as usual via ***[361], *#*[363], *1*[365] etc.

on-mailbox-open-FOLDER, *on-mailbox-open*, *on-mailbox-newmail-FOLDER*, *on-mailbox-newmail*
: Macros which are evaluated when a **folder**[203] is opened or a successful **newmail**[240] check is performed, respectively. **local**[130] scoping is activated by default, causing the covered settings to be reverted no sooner but when *folder-resolved*[428] is left again. The specialization is matched against the fully expanded name, without metacharacters; however, if `FOLDER` resides under *folder*[427] the usual '`+`' (**Filename transformations**[28]) is also tried: for example, if *folder* is "mail" (therefore relative to `HOME`[627]) then `/home/usr1/mail/sent` will be tried as `on-mailbox-open-/home/usr1/mail/sent` first, followed by `on-mailbox-open-+sent` (and lastly `on-mailbox-open`).

on-main-loop-tick
: This hook will be called each time before the main event loop will read an input line. Note variable and other changes it performs are not scoped as via **local**[130]! The main event loop never ticks in (Send mode), *on-compose-enter*[496] can be used instead.

on-program-exit
: This hook will be called when the program exits, whether via **exit**[195] or **quit**[252], or because the send mode is done. **Note:** this runs late and so terminal settings etc. are already teared down.

on-resend-cleanup
: [v15 behaviour may differ] Identical to *on-compose-cleanup*[495], but is only triggered by **resend**[264].

on-resend-enter
: [v15 behaviour may differ] Identical to *on-compose-enter*[496], but is only triggered by **resend**[264]; currently there is no **digmsg**[178] support, for example.

page
: (Boolean) If set, each message feed through the command given for **pipe**[248] is followed by a formfeed character '`\f`'.

password-USER@HOST, *password-HOST*, *password*
: Variable chain that sets a password, which is used in case none has been given in the protocol and account-specific URL; as a last resort S-nail will ask for a password on the user's terminal if the authentication method requires a password. Specifying passwords in a startup file is generally a security risk; the file should be readable by the invoking user only.

piperaw
: (Boolean) Send messages to the **pipe**[248] command without performing MIME and character set conversions.

pipe-EXTENSION
: Identical to *pipe-TYPE/SUBTYPE*[511] except that `EXTENSION` (normalized to lowercase using character mappings of the ASCII charset) denotes a file extension, for example `xhtml`. Handlers registered using this method take precedence.

pipe-TYPE/SUBTYPE
: Whenever a `TYPE/SUBTYPE` (case-insensitive, normalized to lowercase using character mappings of the ASCII charset) MIME message part is displayed or quoted, its data is filtered through the given value interpreted as a shell command. Unless noted only **copiousoutput**[661] parts (see **The Mailcap files**[38]) are covered, other parts are solely considered by **mimeview**[229].

  The value question mark '`?`' forces plain text interpretation of the part, for example `set pipe-application/xml=?`. (**mimetype**[227] type-markers achieve the same.) [Option]ally MIME type handlers may be defined via **The Mailcap files**[38]. It is indeed also a

trigger character to indicate the following flags:

```
? set pipe-X/Y='?!++=? vim ${MAILX_FILENAME_TEMPORARY}'
```

‘*’ The command’s output is reintegratable: **copiousoutput[661]**. This is implied when using a plain ‘’.

‘#’ Only use this handler for display, not for quoting a message: **x-mailx-noquote[664]**.

‘&’ Run the command asynchronously, do not wait for the handler to exit: **x-mailx-async[663]**. The standard output of the command will go to /dev/null[657].

‘!’ The command must be run on an interactive terminal, the terminal will temporarily be released for it to run: **needsterminal[662]**.

‘+’ Request creation of a zero-sized temporary file, the absolute pathname of which will be made accessible via the environment variable MAILX_FILENAME_TEMPORARY[517]: **x-mailx-tmpfile[666]**. If given twice the file will be unlinked automatically: **x-mailx-tmpfile-unlink[668]**; it is an error to use automatic deletion in conjunction with **x-mailx-async[663]**.

‘=’ by default part content is passed to the handler via standard input; if set data is written into MAILX_FILENAME_TEMPORARY[517] (**x-mailx-tmpfile-fill[667]**) instead, the creation of which is implied. Automatic deletion still requires two plus signs ‘++’!

‘t’ Plain text display type-marker (for type-markers: **The mime.types files[37]**). Implies **copiousoutput[661]**.

‘h’ [Option] HTML type-marker: display via built-in HTML-to-text filter. Implies **copiousoutput[661]**.

‘?’ To avoid ambiguities a second question mark can be used to forcefully terminate interpretation of the remaining characters as flags. (Any character not in this list will have the same effect.)

Some information about the MIME part to be displayed is embedded into the environment of the shell command:

MAILX_CONTENT The MIME content-type of the part, if known, the empty string otherwise.

MAILX_CONTENT_EVIDENCE The detected MIME content-type if the carry-around-bit (2) is set in *mime-counter-evidence[478]*, identical to MAILX_CONTENT otherwise.

MAILX_EXTERNAL_BODY_URL MIME parts of type message/external-body access-type=url will store the access URL in this variable, it is empty otherwise. URL targets should not be activated automatically, without supervision.

MAILX_FILENAME The filename, if any is set, the empty string otherwise.

MAILX_FILENAME_GENERATED A random string.

MAILX_FILENAME_TEMPORARY If temporary file creation was requested it will contain the absolute pathname of the temporary file.

pop3-auth-USER@HOST, *pop3-auth-HOST*, *pop3-auth*

[Option] Variable chain that sets the POP3 authentication method. Supported are the default plain, oauthbearer (see **FAQ[43]** entry **But, how about XOAUTH2 / OAUTHBEARER?[46]**), as well as external and externanon for TLS secured connections which pass a client certificate via *tls-config-pairs[601]*. There may be the [Option]al method

`gssapi`. `externanon` does not need any user credentials, `external` and `gssapi` need a *user[614]*, the remains also require a *password[508]*. Unless*pop3-no-apop [521]* is set the `plain` method will [Option]ally be replaced with APOP if possible (see there).

pop3-bulk-load-USER@HOST, *pop3-bulk-load-HOST*, *pop3-bulk-load*
(Boolean)[Option] When accessing a POP3 server S-nail loads the headers of the messages, and only requests the message bodies on user request. For the POP3 protocol this means that the message headers will be downloaded twice. If this variable is set then S-nail will download only complete messages from the given POP3 server(s) instead.

pop3-keepalive-USER@HOST, *pop3-keepalive-HOST*, *pop3-keepalive*
[Option] POP3 servers close the connection after a period of inactivity; the standard requires this to be at least 10 minutes, but practical experience may vary. Setting this variable to a numeric value greater than '`0`' causes a`NOOP` command to be sent each v alue seconds if no other operation is performed.

pop3-no-apop-USER@HOST, *pop3-no-apop-HOST*, *pop3-no-apop*
(Boolean)[Option] Unless this variable is set the MD5 based `APOP` authentication method will be used instead of a chosen `plain` *pop3-auth[518]* when connecting to a POP3 server that advertises support. The advantage of `APOP` is that only a single packet is sent for the user/password tuple. (Originally also that the password is not sent in clear text over the wire, but for one MD5 does not any longer offer sufficient security, and then today transport is almost ever TLS secured.)

pop3-use-starttls-USER@HOST, *pop3-use-starttls-HOST*, *pop3-use-starttls*
(Boolean)[Option] Causes S-nail to issue a `STLS` command to make an unencrypted POP3 session TLS encrypted. This functionality is not supported by all servers. Directly using encrypted communication channels should be preferred.

posix (Boolean) This flag enables POSIX mode, which changes behaviour of S-nail where that deviates from standardized behaviour. It is automatically squared with the environment variable `POSIXLY_CORRECT[642]`, changing the one will adjust the other. The following behaviour is covered and enforced by this mechanism:

- In non-interactive mode, any error encountered while loading resource files during program startup will cause a program exit, whereas in interactive mode such errors will stop loading of the currently loaded (stack of) file(s, i.e., recursively). These exits can be circumvented on a per-command base by using **`ignerr[129]`**, one of the **Command modifiers[22]**, for each command which shall be allowed to fail.
- **`alternates[149]`** will replace the list of alternate addresses instead of appending to it. In addition alternates will only be honoured for any sort of message **`reply[261]`**, and for aliases.
- The variable inserting **COMMAND ESCAPES[30]** **`~A[323]`**, **`~a[324]`**, **`~I[333]`** and **`~i[334]`** will expand embedded character sequences '`\t`' horizontal tabulator and '`\n`' line feed. [v15 behaviour may differ] For compatibility reasons this step will always be performed.
- Reading in messages via **`~f[330]`** (**COMMAND ESCAPES[30]**) will use the `type` not the `forward` **`headerpick[212]`** selection.
- Upon changing the active **`folder[203]`** no summary of **`headers[214]`** will be displayed even if *header[438]* is set.
- Setting *ignoreeof[452]* implies the behaviour described by *dot[414]*.
- The variable *keep[455]* is extended to cover any empty mailbox, not only empty **primary system mailbox[136]**es: they will be removed when they are left in empty state otherwise.
- The exit *?[350]* (and error *![351]*) status of each command replaces that of the former, the last becomes the exit status of the program itself. In POSIX mode the program exit status will instead have bit 2 (value 4) set unless all sent messages were successfully sent out to the *mta[482]*; also see *sendwait[552]*.

print-alternatives
(Boolean) When a MIME message part of type `multipart/alternative` is displayed and it contains a subpart of type `text/plain`, other parts are normally discarded. Setting this variable causes all subparts to be displayed, just as if the surrounding part was of type `multipart/mixed`.

prompt The string used as a prompt in interactive mode. Whenever the variable is evaluated the value is treated as if specified within dollar-single-quotes (see **Shell-style argument quoting**[24]). This (post-assignment, i.e., second) expansion can be used to embed status information, for example *?*[350], *!*[351], *account*[366] or *mailbox-display*[464].

In order to embed characters which should not be counted when calculating the visual width of the resulting string, enclose the characters of interest in a pair of reverse solidus escaped brackets: `\[\E[0m\]`; a slot for coloured prompts is also available with the [Option]al command **`colour`**[164]. Prompting may be prevented by setting this to the null string (aka `set noprompt`).

prompt2 This string is used for secondary prompts, but is otherwise identical to *prompt*[525]. The default is `.. `.

quiet (Boolean) Suppresses the printing of the version when first invoked.

quote If set messages processed by variants of **`followup`**[207] and **`reply`**[261] will start with the original message, lines of which prefixed by *indentprefix*[454], taking into account *quote-chars*[531] and *quote-fold*[532]. No headers will be quoted when set without value or for `noheading`, for `headers` the `type` **`headerpick`**[212] selection will be included in the quote, `allbodies` embeds the (body) contents of all MIME parts, and `allheaders` also includes all headers. The quoted message will be enclosed by the expansions of *quote-inject-head*[533] and *quote-inject-tail*[534]. Also see*quote-add-cc* [529], *quote-as-attachment*[530] and **`~Q`**[338], one of the **COMMAND ESCAPES**[30].

quote-add-cc
(Boolean) Whether senders of messages quoted via **`~Q`**[338] shall be made members of the carbon copies `Cc:` list.

quote-as-attachment
(Boolean) Add the original message in its entirety as a `message/rfc822` MIME attachment when replying to a message, announced as *content-description-quote-attachment*[405]. This works regardless of the setting of *quote*[528].

quote-chars
Can be set to a string consisting of non-whitespace ASCII characters which shall be treated as quotation leaders, the default being `>|}:`.

quote-fold
[Option] Can be set in addition to *indentprefix*[454], and creates a more fancy quotation in that leading quotation characters (*quote-chars*[531]) are compressed and overlong lines are folded. *quote-fold* can be set to either one, two or three (space separated) numeric values, which are interpreted as the maximum (goal) and the minimum line length, respectively, in a spirit rather equal to the `fmt`(1)[747] program, but line- instead of paragraph-based. The third value is used as the maximum line length instead of the first if no better break point can be found; it is ignored unless it is larger than the minimum and smaller than the maximum. If not set explicitly the minimum will reflect the goal algorithmically. The goal cannot be smaller than the length of*indentpr efix*[454] plus some additional pad; necessary adjustments take place silently.

quote-inject-head, *quote-inject-tail*
: The strings to put before and after the text of a *quote[528]*d message, if non-empty, and respectively. The former defaults to `%f wrote:\n\n`. Special format directives will be expanded if possible, and if so configured the output will be folded according to *quote-fold[532]*. Format specifiers in the given strings start with a percent sign '`%`' and expand values of the original message, unless noted otherwise. Note that names and addresses are not subject to the setting of *showto[555]*. Valid format specifiers are:

  '`%%`' A plain percent sign.
  '`%a`' The address(es) of the sender(s).
  '`%d`' The date found in the `Date:` header of the message when *datefield[410]* is set (the default), otherwise the date when the message was received. Formatting can be controlled by assigning a`strftime`(3) [748] format string to *datefield[410]* (and *datefield-markout-older[411]*).
  '`%f`' The full name(s) (name and address, as given) of the sender(s).
  '`%i`' The `Message-ID:`.
  '`%n`' The real name(s) of the sender(s) if there is one and *showname[554]* allows usage, the address(es) otherwise.
  '`%r`' The senders real name(s) if there is one, the address(es) otherwise.

r-option-implicit
: (Boolean) Setting this option evaluates the contents of *from[436]* (or, if that contains multiple addresses, *sender[551]*) and passes the results onto the used (file-based) MTA as described for the **`-r`**[79] option (empty argument case).

recipients-in-cc
: (Boolean) When doing a **`reply`[261]**, the original `From:` and `To:` as well as recipients which possibly came in via `Reply-To:` and `Mail-Followup-To:` are by default merged into the new `To:`. If this variable is set a sensitive algorithm tries to place in `To:` only the sender of the message being replied to, others are placed in `Cc:`.

record
: Unless this variable is defined, no copies of outgoing mail will be saved. If defined it gives the pathname, subject to the usual **Filename transformations[28]**, of a folder where all new, replied-to or forwarded messages are saved: when saving to this folder fails the message is not sent, but instead *save[546]*d to `DEAD`[625]. The standard defines that relative (fully expanded) paths are to be interpreted relative to the current directory (**`cwd`[171]**), to force interpretation relative to *folder[427]* *outfolder[493]* needs to be set in addition.

record-files
: (Boolean) If this variable is set the meaning of *record[537]* will be extended to cover messages which target only file and pipe recipients (see *expandaddr[422]*). These address types will not appear in recipient lists unless *add-file-recipients[367]* is also set.

record-resent
: (Boolean) If this variable is set the meaning of *record[537]* will be extended to also cover the **`resend`[264]** and **`Resend`[263]** commands.

reply-in-same-charset
: (Boolean) If this variable is set S-nail first tries to use the same character set of the original message for replies. If this fails, the mechanism described in **Character sets[12]** is evaluated as usual.

reply-strings
: Can be set to a comma-separated list of (case-insensitive according to ASCII rules) strings which shall be recognized in addition to the built-in strings as `Subject:` reply message indicators – built-in are `Re:`, which is mandated by RFC 5322, as well as the german `Aw:`, `Antw:`, and the `Wg:` which often has been seen in the wild; I.e., the separating colon has to be specified explicitly.

reply-to A list of addresses to put into the `Reply-To:` field of the message header. Members of this list are handled as if they were in the **`alternates`**`[149]` list.

reply-to-honour
Controls whether a `Reply-To:` header is honoured when replying to a message via **`reply`**`[261]` or **`Lreply`**`[222]`. This is a**quadoption [349]**; if set without a value it defaults to "yes".

reply-to-swap-in
Standards like DKIM and (in conjunction with) DMARC caused many **Mailing lists[11]** to use sender address rewriting in the style of `Name via List <list@address>`, where the original sender address often being placed in `Reply-To:`. If this is set and a`Reply-To:` e xists, and consists of only one recipient (!), then that is used in place of the pretended sender. This works independently from *reply-to-honour[543]*. The optional value, a comma-separated list of strings, offers more fine-grained control on when swapping shall be used; for now supported is *mlist*, here swapping occurs if the sender is a mailing-list as defined by **`mlist`**`[230]`.

*rfc822-body-from_*
(Boolean) This variable can be used to force displaying a so-called `From_` line for messages that are embedded into an envelope mail via the `message/rfc822` MIME mechanism, for more visual convenience, also see *mbox-rfc4155[470]*.

save (Boolean) Enable saving of (partial) messages in `DEAD`[625] upon interrupt or delivery error.

screen The number of lines that represents a "screenful" of lines, used in **`headers`**`[214]` summary display, **`from`**`[211]` **`search`**`[269]`ing, message **`top`**`[294]`line display and scrolling via **`z`**`[312]`. If this variable is not set a calculation based upon the detected terminal window size and the baud rate is used: the faster the terminal, the more will be shown. Overall screen dimensions and pager usage is influenced by the environment variables `COLUMNS`[624] and `LINES`[631] and the variable *crt[408]*.

searchheaders
(Boolean) Expand message list specifiers in the form `/x:y` to all messages containing the substring "y" in the header field '`x`'. The string search is case insensitive.

sendcharsets
[Option] A comma-separated list of character set names that can be used in outgoing internet mail. The value of the variable *charset-8bit[397]* is automatically appended to this list of character sets. If no character set conversion capabilities are compiled into S-nail then the only supported charset is *ttycharset[611]*. Also see*endc harsets-else-ttycharset[550]* and refer to the section **Character sets[12]** for the complete picture of character set conversion in S-nail.

sendcharsets-else-ttycharset
(Boolean)[Option] If this variable is set, but *sendcharsets[549]* is not, then S-nail acts as if *sendcharsets[549]* had been set to the value of the variable *ttycharset[611]*. In effect this combination passes through the message data in the character set of the current locale encoding: therefore mail message text will be (assumed to be) in ISO-8859-1 encoding when send from within a ISO-8859-1 locale, and in UTF-8 encoding when send from within an UTF-8 locale.

The 8-bit fallback *charset-8bit[397]* never comes into play as *ttycharset[611]* is implicitly assumed to be 8-bit and capable to represent all files the user may specify (as is the case when no character set conversion support is available in S-nail and the only supported character set is *ttycharset[611]*, see **Character sets[12]**). This might be a problem for scripts which use the suggested `LC_ALL=C` setting, since in this case the character set is US-ASCII by definition, so that it is better to also override *ttycharset[611]*, then; and/or do something like the following in the resource file:

```
# Avoid ASCII "propagates to 8-bit" when scripting
\if ! t && "$LC_ALL" != C && "$LC_CTYPE" != C
  \set sendcharsets-else-ttycharset
\end
```

sender An address that is put into the `Sender:` field of outgoing messages, quoting the standard RFC 5322: the mailbox of the agent responsible for the actual transmission of the message. For example, if Mary sends a mail for Alice, then Alice should be in `From:` whereas Mary should be present in `Sender:`. According to the standard this variable **must** be set if *from[436]* contains multiple addresses. [v15 behaviour may differ] Expect automatic management of the *from[436]* and *sender[551]* relationship (requiring an address order in the former). Dependent on the context this address is handled as if it were in the list of **`alternates`[149]**. Also see**`-r` [79]**, *r-option-implicit[535]*.

sendwait Sending messages to the chosen *mta[482]* or to command-pipe recipients (see **On sending mail, and non-interactive mode[7]**) will be performed asynchronously. This means that only startup errors of the respective program will be recognizable, but no delivery errors. Also, no guarantees can be made as to when the respective program will actually run, as well as to when they will have produced output.

If this variable is set then child program exit is waited for, and its exit status code is used to decide about success. Remarks: in conflict with the POSIX standard this variable is built-in to be initially set. Another difference is that it can have a value, which is interpreted as a comma-separated list of case-insensitive strings naming specific subsystems for which synchronousness shall be ensured (only). Possible values are `mta` for *mta[482]* delivery, and `pcc` for command-pipe recipients.

showlast (Boolean) This setting causes S-nail to start at the last message instead of the first one when opening a mail folder, as well as with **`from`[211]** and **`headers`[214]**.

showname
(Boolean) Causes S-nail to use the sender's real name instead of the plain address in the header field summary and in message specifications.

showto (Boolean) Causes the recipient of the message to be shown in the header summary if the message was sent by the user.

Sign The value backing **`~A`[323]**, one of the **COMMAND ESCAPES[30]**. Also see *message-inject-tail[474]*, *on-compose-leave[497]* and *on-compose-splice[498]*.

sign The value backing **`~a`[324]**, one of the **COMMAND ESCAPES[30]**. Also see *message-inject-tail[474]*, *on-compose-leave[497]* and *on-compose-splice[498]*.

skipemptybody
(Boolean) If an outgoing message has an empty first or only message part, do not send, but discard it, successfully (also see the command line option **`-E`[65]**).

smime-ca-dir, *smime-ca-file*
[Option] Specify the location of trusted CA certificates in PEM (Privacy Enhanced Mail) for the purpose of verification of S/MIME signed messages. *tls-ca-dir [595]* documents the necessary preparation steps to use the former. The set of CA certificates which are built into the TLS library can be explicitly turned off by setting *smime-ca-no-defaults[562]*, and further fine-tuning is possible via *smime-ca-flags[561]*.

smime-ca-flags
[Option] Can be used to fine-tune behaviour of the X509 CA certificate storage, and the certificate verification that is used. The actual values and their meanings are documented for *tls-ca-flags[597]*.

smime-ca-no-defaults
(Boolean)[Option] Do not load the default CA locations that are built into the used to TLS library to verify S/MIME signed messages.

smime-cipher-USER@HOST, *smime-cipher*
[Option] Specifies the cipher to use when generating S/MIME encrypted messages (for the specified account). RFC 5751 mandates a default of `aes128` (AES-128 CBC). Possible values are (case-insensitive and) in decreasing cipher strength: `aes256` (AES-256 CBC), `aes192` (AES-192 CBC), `aes128` (AES-128 CBC), `des3` (DES EDE3 CBC, 168 bits; default if `aes128` is not available) and `des` (DES CBC, 56 bits).

The actually available cipher algorithms depend on the cryptographic library that S-nail uses. [Option] Support for more cipher algorithms may be available through dynamic loading via `EVP_get_cipherbyname`(3)[749] (OpenSSL) if S-nail has been compiled to support this.

smime-crl-dir
[Option] Specifies a directory that contains files with CRLs in PEM format to use when verifying S/MIME messages.

smime-crl-file
[Option] Specifies a file that contains a CRL in PEM format to use when verifying S/MIME messages.

smime-encrypt-USER@HOST
[Option] If this variable is set, messages send to the given recipient are encrypted before sending. The value of the variable must be set to the name of a file that contains a certificate in PEM format.

If a message is sent to multiple recipients, each of them for whom a corresponding variable is set will receive an individually encrypted message; other recipients will continue to receive the message in plain text unless the *smime-force-encryption[567]* variable is set. It is recommended to sign encrypted messages, i.e., to also set the *smime-sign[568]* variable. *content-description-smime-message[406]* will be inspected for messages which become encrypted.

smime-force-encryption
(Boolean)[Option] Causes S-nail to refuse sending unencrypted messages.

smime-sign
(Boolean)[Option] S/MIME sign outgoing messages with the user's (*from[436]*) private key and include the users certificate as a MIME attachment. Signing a message enables a recipient to verify that the sender used a valid certificate, that the email addresses in the certificate match those in the message header and that the message content has not been altered. It does not change the message text, and people will be able to read the message as usual. *content-description-smime-signature[407]* will be inspected. Also see *smime-sign-cert[569]*, *smime-sign-include-certs[571]* and *smime-sign-digest[570]*.

smime-sign-cert-USER@HOST, *smime-sign-cert*
[Option] Points to a file in PEM format. For the purpose of signing and decryption this file needs to contain the user's private key followed by the certificate.

For message signing `USER@HOST` is always derived from the value of *from[436]* (or, if that contains multiple addresses, *sender[551]*). For the purpose of encryption the recipients public encryption key (certificate) is expected; the command **`certsave`[158]** can be used to save certificates of signed messages (the section **Signed and encrypted messages with S/MIME[17]** gives some details). This mode of operation is usually driven by the specialized form.

When decrypting messages the account is derived from the recipient fields (`To:` and `Cc:`) of the message, which are searched for addresses for which such a variable is set. S-nail always uses the

first address that matches, so if the same message is sent to more than one of the user addresses using different encryption keys, decryption might fail.

Password-encrypted keys may be used for signing and decryption. Automated password lookup is possible via the "pseudo-hosts" `USER@HOST.smime-cert-key` for the private key, and `USER@HOST.smime-cert-cert` for the certificate stored in the same file. For example, the hypothetical address `bob@exam.ple` could be driven with a private key / certificate pair path defined in *smime-sign-cert-bob@exam.ple*, and the needed passwords would then be looked up as `bob@exam.ple.smime-cert-key` and `bob@exam.ple.smime-cert-cert`. When decrypting the value of *from[436]* will be tried as a fallback to provide the necessary `USER@HOST`. To include intermediate certificates, use *smime-sign-include-certs[571]*. The possible password sources are documented in **On URL syntax and credential lookup[18]**.

smime-sign-digest-USER@HOST, *smime-sign-digest*

[Option] Specifies the message digest to use when signing S/MIME messages. Remember that for this use case `USER@HOST` refers to the variable *from[436]* (or, if that contains multiple addresses, *sender[551]*). The available algorithms depend on the used cryptographic library, but at least one usable built-in algorithm is ensured as a default. If possible the standard RFC 5751 will be violated by using `SHA512` instead of the mandated `SHA1` due to security concerns. This variable is ignored for very old (released before 2010) cryptographic libraries which do not offer the necessary interface: it will be logged if that happened.

S-nail will try to add built-in support for the following message digests, names are case-insensitive: `BLAKE2b512`, `BLAKE2s256`, `SHA3-512`, `SHA3-384`, `SHA3-256`, `SHA3-224`, as well as the widely available `SHA512`, `SHA384`, `SHA256`, `SHA224`, and the proposed insecure `SHA1`, finally `MD5`. More digests may [Option]ally be available through dynamic loading via the OpenSSL function `EVP_get_digestbyname`(3)[750].

smime-sign-include-certs-USER@HOST, *smime-sign-include-certs*

[Option] If used, this is supposed to a consist of a comma-separated list of files, each of which containing a single certificate in PEM format to be included in the S/MIME message in addition to the *smime-sign-cert[569]* certificate. This can be used to include intermediate certificates of the certificate authority, in order to allow the recipient's S/MIME implementation to perform a verification of the entire certificate chain, starting from a local root certificate, over the intermediate certificates, down to the *smime-sign-cert[569]*. Even though top level certificates may also be included in the chain, they will not be used for the verification on the recipient's side.

For the purpose of the mechanisms involved here, `USER@HOST` refers to the content of the internal variable *from[436]* (or, if that contains multiple addresses, *sender[551]*). The pseudo-host `USER@HOST.smime-include-certs` will be used for performing password lookups for these certificates, shall they have been given one, therefore the lookup can be automated via the mechanisms described in **On URL syntax and credential lookup[18]**.

smtp-auth-USER@HOST, *smtp-auth-HOST*, *smtp-auth*

[Option][Obsolete] Use the authentication slots of *smtp-config[573]*.

smtp-config-USER@HOST, *smtp-config-HOST*, *smtp-config*

[Option] When a SMTP based *mta[482]* is contacted a list of supported SMTP service extensions will (optionally) be announced by the server. This comma-separated (case-insensitive) list configures which extensions shall be used, and which of the available ones shall not. Order matters, whitespace is ignored, an optional plus sign '`+`' prefix enables, a hyphen-minus '`-`' prefix disables usage of an extension, for example `-all, ehlo,+starttls, gssapi`.

all This special word enables (default) or disables all extensions. Disabling it also disables all the below authentication mechanisms.

ehlo Service Extensions (RFC 1869) added the notion of extensions to the SMTP protocol; when disabled, all other extensions are also disabled (for **auth** only the master switch is toggled, not the individual mechanisms), enabling any extension (re-)implies this.

8bitmime
8-bit MIME Transport (RFC 6152) enables usage of the `8bit` *mime-encoding[479]*. If disabled or not supported by the server, SMTP will not send a `8bit` message, and optionally *save[546]* the text in `DEAD[625]`; as this happens late the message will also have been *record[537]*ed.

pipelining
Command Pipelining (RFC 2920) helps saving packet roundtrips by allowing successive commands without waiting for respective server responses.

starttls
Secure SMTP over TLS (Transport Layer Security, RFC 3207) allows upgrading an unencrypted (SMTP not SMTPS) connection to use private, authenticated communication. To improve security and provide a safety measure against man-in-the-middle attacks this is always performed — even if the server does not announce it — unless explicitly turned off. Directly using encrypted transport channels should be preferred, as it saves network traffic.

auth Authentication (RFC 4954) allows account credentials to be passed. This word disables all authentication mechanisms, but enables only those which can be managed automatically without external help; for example GSSAPI requires an externally granted ticket to exist, and is therefore excluded by the default automatic selection, as is EXTERNAL etc.: these mechanisms have to be enabled explicitly. The default selection depends upon the (im- or explicit) presence of TLS.

If multiple **auth**entication mechanisms are available, an automatic selection of the “best” method is performed, preferring the non-automatic mechanisms. The used list can be fine-tuned, any non-empty list implies **auth**. For example, `smtp-config=-allmechs,gssapi,external,plain` will favour external over gssapi, and use plain as a last resort only. Beware, in the following example `plain` would still be used as a last ressort, sending credentials in clear (unless the transport is of an encrypted type): `smtp-config=-all,,gssapi,plain`. The following mechanisms are known:

allmechs
Special word which covers all authentication methods (where “all” means all for disabling, and all supported ones for enabling).

cram-md5
[Option] Challenge-Response authentication mechanism (CRAM; included in RFC 2195), needs *user[614]* and *password[508]*.

external
[Option] Included in the simple authentication and security layer (SASL; RFC 4422). Authentication happens through a TLS client certificate (see *tls-config-pairs[601]*, **Certificate**) on the transport layer, therefore not automatic. Needs a *user[614]*.

externanon
[Option] Likewise, but an empty user name is passed, as it is expected that the server extracts the name from the certificate. Not compliant with RFC 4422 / RFC 4954, but has been seen in the wild.

gssapi [Option] The Kerberos V5 ("GSSAPI") mechanism (RFC 4752). Needs an external ticket (to be granted by `kinit`(1)[751]), therefore not automatic. Needs a *user[614]*.

login The LOGIN mechanism (`draft-murchison-sasl-login-00.txt`). It requires three packet roundtrips, has been obsoleted by the IETF, and should only be used as a last resort. Needs *user[614]* and *password[508]*.

oauthbearer
A set of mechanisms for OAuth (RFC 7628). One packet roundtrip. Needs *user[614]* and *password[508]*. The password is a temporary bearer token, not the real password, and therefore this mechanism is not automatic. Also see the **FAQ[43]** entry **But, how about XOAUTH2 / OAUTHBEARER?[46]**

plain The PLAIN mechanism (RFC 4616). One packet roundtrip, needs *user[614]* and *password[508]*.

xoauth2
A popular slightly different variant of the later standardized **oauthbearer**.

smtp-from-USER@HOST, *smtp-from-HOST*, *smtp-from*
[Option] Until this is set the necessary `USER@HOST` information to issue a `MAIL FROM:<>` SMTP *mta[482]* command is derived from the variable *from[436]* if set, or *user[614]* and *hostname[447]* otherwise. If the [Option]al IDNA support is available (see *idna-disable[448]*) variable assignment is aborted when a necessary conversion fails.

smtp-hostname-USER@HOST, *smtp-hostname-HOST*, *smtp-hostname*
[Option][Obsolete] Use the more powerful successor *smtp-from[574]*.

smtp-use-starttls-USER@HOST, *smtp-use-starttls-HOST*, *smtp-use-starttls*
(Boolean)[Option][Obsolete] Use *smtp-config[573]*. (Forcing TLS has become a default setting.)

socket-connect-timeout
[Option] A positive number that defines the timeout to wait for establishing a socket connection before forcing *ˆERR[356]*-TIMEDOUT.

socks-proxy-USER@HOST, *socks-proxy-HOST*, *socks-proxy*
[Option] If set to the URL of a SOCKS5 server then all network activities are proxied through it, except for the single DNS name lookup necessary to resolve the proxy URL (unnecessary when given an already resolved IP address). It is automatically squared with the environment variable `SOCKS5_PROXY[644]`, changing the one will adjust the other. This example creates a local SOCKS5 proxy on port 10000 that forwards to the machine `HOST` (with identity `USER`), and from which actual network traffic happens:

```
$ ssh -D 10000 USER@HOST
$ s-nail -Ssocks-proxy=[socks5://]localhost:10000
# or =localhost:10000; no local DNS: =127.0.0.1:10000
```

spam-interface
[Option] In order to use any of the spam-related commands (like **spamrate[289]**) the desired spam interface must be defined by setting this variable. Refer to the manual section**Handling spam[20]** for the complete picture of spam handling in S-nail. All or none of the following interfaces may be available:

`spamc` Interaction with `spamc`(1)[752] from the `spamassassin`(1)[753] (*SpamAssassin*: **http://spamassassin.apache.org**) suite. Different to the generic filter interface S-nail will automatically add the correct arguments for a given command and has the necessary knowledge to parse the program's output. A default value for *spamc-command[581]* will have been compiled into the S-nail binary if `spamc`(1)[754] has been found in `PATH[641]` during compilation. Shall it be necessary to define a specific connection type (rather than using a configuration file for that), the variable *spamc-arguments[582]* can be used as in for example `-d server.example.com -p 783`. It is also possible to specify a per-user configuration via *spamc-user[583]*. Note that this interface

does not inspect the `is-spam` flag of a message for the command **`spamforget`**[287].

`filter` generic spam filter support via freely configurable hooks. This interface is meant for programs like `bogofilter`(1)[755] and requires according behaviour in respect to the hooks' exit status for at least the command **`spamrate`**[289] (‘`0`’ meaning a message is spam, ‘`1`’ for non-spam, ‘`2`’ for unsure and any other return value indicating a hard error); since the hooks can include shell code snippets diverting behaviour can be intercepted as necessary. The hooks are*spamfilter-ham* *[584]*, *spamfilter-noham*[585], *spamfilter-nospam*[586], *spamfilter-rate*[587] and *spamfilter-spam*[588]; the manual section **Handling spam**[20] contains examples for some programs. The process environment of the hooks will have the variable `MAILX_FILENAME_GENERATED`[516] set. Note that spam score support for**`spamrate`** [289] is not supported unless *spamfilter-rate-scanscore*[589] variable is set.

spam-maxsize
: [Option] Messages that exceed this size will not be passed through to the configured *spam-interface*[579]. If unset or 0, the default of 420000 bytes is used.

spamc-command
: [Option] The path to the `spamc`(1)[756] program for the `spamc` *spam-interface*[579]. Note that the path is not expanded, but used “as is”. A fallback path will have been compiled into the S-nail binary if the executable had been found during compilation.

spamc-arguments
: [Option] Even though S-nail deals with most arguments for the `spamc` *spam-interface*[579] automatically, it may at least sometimes be desirable to specify connection-related ones via this variable, for example `-d server.example.com -p 783`.

spamc-user
: [Option] Specify a username for per-user configuration files for the `spamc` *spam-interface*[579]. If this is set to the empty string then S-nail will use the name of the current *user*[614].

spamfilter-ham, *spamfilter-noham*, *spamfilter-nospam*, *spamfilter-rate*, *spamfilter-spam*
: [Option] Command and argument hooks for the `filter` *spam-interface*[579]. The manual section **Handling spam**[20] contains examples for some programs.

spamfilter-rate-scanscore
: [Option] Spam scores are not supported for the `filter` *spam-interface*[579] unless set. If the [Option]nal regular expression support is available then it will be interpreted as a number, followed by semicolon ‘`;`’ and an extended regular expression. The first output line of the *spamfilter-rate*[587] hook is then evaluated accordingly: upon success the regex group given by the number is interpreted as a floating point scan score. For example `-S spamfilter-rate-scanscore="1;^(.+)$"` simply interprets the entire output line as one.

stealthmua
: If only set without an assigned value, then this setting inhibits the generation of the `Message-ID:`, `Content-ID:` and `User-Agent:` header fields that include obvious references to S-nail. There are two pitfalls associated with this: First, the message id of outgoing messages is not known anymore. Second, an expert may still use the remaining information in the header to track down the originating mail user agent. If set to the value `noagent`, then the mentioned `Message-ID:` and `Content-ID:` suppression does not occur.

system-mailrc
: (Read-only) The compiled-in path of `s-nail.rc`[659], the system-wide of the **Resource files**[36].

termcap
: ([Option]) This specifies a comma-separated list of Terminal Information Library (libterminfo, −lterminfo) and/or Termcap Access Library (libtermcap, −ltermcap) capabilities (see **On terminal control and line editor**[15], escape commas with reverse solidus '\') to be used to overwrite or define entries. **Note:** this v ariable will only be queried once at program startup and can thus only be specified in resource files or on the command line. It will always be inspected, regardless of whether *features*[425] denotes termcap/terminfo library support via `,+termcap,`.

  String capabilities form `cap=value` pairs and are expected unless noted otherwise. Numerics have to be notated as `cap#number` where the number is expected in normal decimal notation. Finally, booleans do not have any value but indicate a true or false state simply by being defined or not; this indeed means that S-nail does not support undefining an existing boolean. String capability values will undergo some expansions before use: for one notations like `^LETTER` stand for `control-LETTER`, and for clarification purposes '`\E`' can be used to specify`escape` (the control notation '`^[`' could lead to misreadings when a left bracket follows, which it does for the standard CSI sequence); finally three letter octal sequences, as in `\061`, are supported. To specify that a terminal supports 256-colours, and to define sequences that home the cursor and produce an audible bell, one might write:

```
? set termcap='Co#256,home=\E[H,bel=^G'
```

  The following terminal capabilities are or may be meaningful for the operation of the built-in line editor or S-nail in general:

  am
  : **auto_right_margin**: boolean which indicates if the right margin needs special treatment; the **xenl** capability is related, for more see `COLUMNS`[624]. This capability is only used when backed by library support.

  clear or **cl**
  : **clear_screen**: clear the screen and home cursor. (Will be simulated via **ho** plus **cd**.)

  colors or **Co**
  : **max_colors**: numeric capability specifying the maximum number of colours. Note that S-nail does not actually care about the terminal beside that, but always emits ANSI / ISO 6429 escape sequences; also see **colour**[164].

  cr
  : **carriage_return**: move to the first column in the current row. The default built-in fallback is '`\r`'.

  cub1 or **le**
  : **cursor_left**: move the cursor left one space (non-destructively). The default built-in fallback is '`\b`'.

  cuf1 or **nd**
  : **cursor_right**: move the cursor right one space (non-destructively). The default built-in fallback is `\E[C`, which is used by most terminals. Less often occur `\EC` and `\EOC`.

  ed or **cd**
  : **clr_eos**: clear the screen.

  el or **ce**
  : **clr_eol**: clear to the end of line. (Will be simulated via **ch** plus repetitions of space characters.)

  home or **ho**
  : **cursor_home**: home cursor.

hpa or **ch**
: **column_address**: move the cursor (to the given column parameter) in the current row. (Will be simulated via **cr** plus **nd**.)

rmcup or **te** / **smcup** or **ti**
: **exit_ca_mode** and **enter_ca_mode**, respectively: exit and enter the alternative screen ca-mode, effectively turning S-nail into a fullscreen application. This must be enabled explicitly by setting *termcap-ca-mode[593]*.

smkx or **ks** / **rmkx** or **ke**
: **keypad_xmit** and **keypad_local**, respectively: enable and disable the keypad. This is always enabled if available, because it seems even keyboards without keypads generate other key codes for, e.g., cursor keys in that case, and only if enabled we see the codes that we are interested in.

xenl or **xn**
: **eat_newline_glitch**: boolean which indicates whether a newline written in the last column of an **auto_right_margin** indicating terminal is ignored. With it the full terminal width is available even on autowrap terminals. This will be inspected even without `,+termcap,` *features[425]*.

Many more capabilities which describe key-sequences are documented for **bind[153]**.

termcap-ca-mode
: [Option] Allow usage of the **exit_ca_mode** and **enter_ca_mode** *termcap[592]*abilities in order to enter an alternative exclusive screen, the so-called ca-mode; this usually requires special configuration of the `PAGER`[640], also dependent on the value of *crt[408]*. If set to a non-empty value the alternative screen is cleared before it is left, as via **mle-clear-screen[121]**. **Note:** this variable will only be queried once at program startup and can thus only be specified in resource files or on the command line.

termcap-disable
: (Boolean)[Option] Disable any interaction with a terminal control library. If set only some generic fallback built-ins and possibly the content of *termcap[592]* describe the terminal. **Note:** this v ariable will only be queried once at program startup and can thus only be specified in resource files or on the command line.

tls-ca-dir-USER@HOST, *tls-ca-dir-HOST*, *tls-ca-dir*, *tls-ca-file-USER@HOST*, *tls-ca-file-HOST*, *tls-ca-file*
: [Option] Directory and file, respectively, for pools of trusted CA certificates in PEM (Privacy Enhanced Mail) format, for the purpose of verification of TLS server certificates. Concurrent use is possible, the file is loaded once needed first, the directory lookup is performed anew as a last resort whenever necessary. The CA certificate pool built into the TLS library can be disabled via *tls-ca-no-defaults[598]*, further fine-tuning is possible via *tls-ca-flags[597]*. The directory search requires special filename conventions, see `SSL_CTX_load_verify_locations`(3)[757] and `verify`(1)[758] (or `c_rehash`(1)[759]).

tls-ca-flags-USER@HOST, *tls-ca-flags-HOST*, *tls-ca-flags*
: [Option] Can be used to fine-tune behaviour of the X509 CA certificate storage, and the certificate verification that is used (also see *tls-verify[608]*). The value is expected to consist of a comma-separated list of configuration directives, with any intervening whitespace being ignored. The directives directly map to flags that can be passed to `X509_STORE_set_flags`(3)[760], which are usually defined in a file `openssl/x509_vfy.h`, and the availability of which depends on the used TLS library version: a directive without mapping is ignored (error log subject to *debug[412]*). Directives currently understood (case-insensitively) include:

no-alt-chains
If the initial chain is not trusted, do not attempt to build an alternative chain. Setting this flag will make OpenSSL certificate verification match that of older OpenSSL versions, before automatic building and checking of alternative chains has been implemented; also see **trusted-first**.

no-check-time
Do not check certificate/CRL validity against current time.

partial-chain
By default partial, incomplete chains which cannot be verified up to the chain top, a self-signed root certificate, will not verify. With this flag set, a chain succeeds to verify if at least one signing certificate of the chain is in any of the configured trusted stores of CA certificates. The OpenSSL manual page SSL_CTX_load_verify_locations(3)[761] gives some advise how to manage your own trusted store of CA certificates.

strict Disable workarounds for broken certificates.

trusted-first
Try building a chain using issuers in the trusted store first to avoid problems with server-sent legacy intermediate certificates. Newer versions of OpenSSL support alternative chain checking and enable it by default, resulting in the same behaviour; also see **no-alt-chains**.

tls-ca-no-defaults-USER@HOST, *tls-ca-no-defaults-HOST*, *tls-ca-no-defaults*
(Boolean)[Option] Do not load the default CA locations that are built into the used to TLS library to verify TLS server certificates.

tls-config-file
[Option] If this variable is set CONF_modules_load_file(3)[762] (if announced via ,+modules-load-file, in *tls-features[604]*) is used to allow resource file based configuration of the TLS library. This happens once the library is used first, which may also be early during startup (logged with *verbose[616]*)! If a non-empty value is given then the given file, after performing **Filename transformations[28]**, will be used instead of the TLS libraries global default, and it is an error if the file cannot be loaded. The application name will always be passed as s-nail. Some TLS libraries support application-specific configuration via resource files loaded like this, see *tls-config-module[600]*.

tls-config-module-USER@HOST, *tls-config-module-HOST*, *tls-config-module*
[Option] If file based application-specific configuration via *tls-config-file[599]* is available, announced as ,+ctx-config, by *tls-features[604]*, indicating availability of SSL_CTX_config(3)[763], then, it becomes possible to use a central TLS configuration file for all programs, including s-nail, for example

```
# Register a configuration section for s-nail
s-nail = mailx_master
# The top configuration section creates a relation
# in between dynamic SSL configuration and an actual
# program specific configuration section
[mailx_master]
ssl_conf = mailx_tls_config
# And that program specific configuration section now
# can map diverse tls-config-module names to sections,
# as in: tls-config-module=account_xy
[mailx_tls_config]
account_xy = mailx_account_xy
account_yz = mailx_account_yz
```

```
[mailx_account_xy]
MinProtocol = TLSv1.2
Curves=P-521
[mailx_account_yz]
CipherString = TLSv1.2:!aNULL:!eNULL:
MinProtocol = TLSv1.1
Options = Bugs
```

tls-config-pairs-USER@HOST, *tls-config-pairs-HOST*, *tls-config-pairs*
: [Option] The value of this variable chain will be interpreted as a comma-separated list of directive/value pairs. Directives and values need to be separated by equals signs '`=`', any whitespace surrounding pair members is removed. Keys are (usually) case-insensitive. Different to when placing these pairs in a *tls-config-module[600]* section of a *tls-config-file[599]*, commas '`,`' need to be escaped with a reverse solidus '`\`' when included in pairs; also different: if the equals sign '`=`' is preceded with an asterisk '`*`' **Filename transformations[28]** will be performed on the value; it is an error if these fail. Unless proper support is announced by*tls-featur es[604]* (`,+conf-ctx,`) only the keys below are supported, otherwise the pairs will be used directly as arguments to the function `SSL_CONF_cmd`(3)[764].

  `Certificate` Filename of a TLS client certificate (chain) required by some servers. Fallback support via `SSL_CTX_use_certificate_chain_file`(3)[765]. **Filename transformations[28]** are performed. **`PrivateKey`** will be set to the same value if not initialized explicitly. Some services support so-called `external` authentication if a TLS client certificate was successfully presented during connection establishment ("connecting is authenticating").

  `CipherString` A list of ciphers for TLS connections, see `ciphers`(1)[766]. By default no list of ciphers is set, resulting in a **`Protocol`**-specific list of ciphers (the protocol standards define lists of acceptable ciphers; possibly cramped by the used TLS library). Fallback support via `SSL_CTX_set_cipher_list`(3)[767].

  `Ciphersuites` A list of ciphers used for TLSv1.3 connections, see `ciphers`(1)[768]. These will be joined onto the list of ciphers from **`CipherString`**. Available if *tls-features[604]* announces `,+ctx-set-ciphersuites,`, as necessary via `SSL_CTX_set_ciphersuites`(3)[769].

  `Curves` A list of supported elliptic curves, if applicable. By default no curves are set. Fallback support via `SSL_CTX_set1_curves_list`(3)[770], if available.

  `MaxProtocol`, **`MinProtocol`**
  : The maximum and minimum supported TLS versions, respectively. Available if *tls-features[604]* announces `,+ctx-set-maxmin-proto,`, as necessary via `SSL_CTX_set_max_proto_version`(3)[771] and `SSL_CTX_set_min_proto_version`(3)[772]; these fallbacks use an internal parser which understands the strings `SSLv3`, `TLSv1`, `TLSv1.1`, `TLSv1.2`, `TLSv1.3`, and the special value `None`, which disables the given limit.

  `Options` Various flags to set. Fallback via `SSL_CTX_set_options`(3)[773], in which case any other value but (exactly) `Bugs` results in an error.

  `PrivateKey` Filename of the private key in PEM format of a TLS client certificate. If unset, the value of **`Certificate`** is used. **Filename transformations[28]** are performed. Fallback via `SSL_CTX_use_PrivateKey_file`(3)[774].

  `Protocol` The used TLS protocol. If *tls-features[604]* announces `,+conf-ctx,` or `ctx-set-maxmin-proto` then using **`MaxProtocol`** and **`MinProtocol`** is preferable. Fallback is `SSL_CTX_set_options`(3)[775], driven via an internal parser which understands the strings `SSLv3`, `TLSv1`, `TLSv1.1`, `TLSv1.2`, `TLSv1.3`, and the special value `ALL`. Multiple protocols may be

given as a comma-separated list, any whitespace is ignored, an optional plus sign '+' prefix enables, a hyphen-minus '-' prefix disables a protocol, so that `-ALL, TLSv1.2`enables only the TLSv1.2 protocol.

tls-crl-dir, *tls-crl-file*
: [Option] Specify a directory / a file, respectively, that contains a CRL in PEM format to use when verifying TLS server certificates.

tls-features
: [Option](Read-only) This expands to a comma-separated list of the TLS library identity and optional features. To ease substring matching the string starts and ends with a comma. Currently supported identities are `libressl` (LibreSSL) , `libssl-0x30000` (OpenSSL v3.0.0 series), `libssl-0x10100` (OpenSSL v1.1.x series) and `libssl-0x10000` (elder OpenSSL series, other clones). Optional features are preceded with a plus sign '+' when available, and with a hyphen-minus '-' otherwise.

  Currently known features are `conf-ctx` (*tls-config-pairs[601]*), `ctx-config` (*tls-config-module[600]*), `ctx-set-ciphersuites` (**`Ciphersuites`** slot of *tls-config-pairs[601]*), `ctx-set-maxmin-proto` (*tls-config-pairs[601]*), `modules-load-file` (*tls-config-file[599]*), and `tls-rand-file` (*tls-rand-file[607]*).

tls-fingerprint-USER@HOST, *tls-fingerprint-HOST*, *tls-fingerprint*
: [Option] It is possible to replace the verification of the connection peer certificate against the entire local pool of CAs (for more see **Encrypted network communication[19]**) with the comparison against a precalculated certificate message digest, the so-called fingerprint, to be specified as the used *tls-fingerprint-digest[606]*. This fingerprint can for example be calculated with **`tls[292]`** fingerprint HOST.

tls-fingerprint-digest-USER@HOST, *tls-fingerprint-digest-HOST*, *tls-fingerprint-digest*
: [Option] The message digest to be used when creating TLS certificate fingerprints, the defaults, if available, in test order, being `BLAKE2s256`, `SHA256`. For the complete list of digest algorithms refer to *smime-sign-digest[570]*.

tls-rand-file
: [Option] If *tls-features[604]* announces `,+tls-rand-file,` then this will be queried to find a file with random entropy data which can be used to seed the P(seudo)R(andom)N(umber)G(enerator), see `RAND_load_file`(3)[776]. The default filename (`RAND_file_name`(3)[777], normally `~/.rnd[658]`) will be used if this variable is not set or empty, or if the **Filename transformations[28]** fail. Shall seeding the PRNG have been successful, `RAND_write_file`(3)[778] will be called to update the entropy. Remarks: libraries which do not announce this feature seed the PRNG by other means.

tls-verify-USER@HOST, *tls-verify-HOST*, *tls-verify*
: [Option] Variable chain that sets the action to be performed if an error occurs during TLS server certificate validation against the specified or default trust stores *tls-ca-dir[595]*, *tls-ca-file[596]*, or the TLS library built-in defaults (unless usage disallowed via *tls-ca-no-defaults[598]*), and as fine-tuned via *tls-ca-flags[597]*. Valid (case-insensitive) values are `strict` (fail and close connection immediately), `ask` (ask whether to continue on standard input), `warn` (show a warning and continue), `ignore` (do not perform validation). The default is `ask`.

toplines
: If defined, gives the number of lines of a message to be displayed with the command **`top[294]`**; if unset, the first five lines are printed, if set to 0 the variable *screen[547]* is inspected. If the value is negative then its absolute value will be used for unsigned right shifting (see **`vexpr[306]`**) the *screen[547]* height.

topsqueeze
: (Boolean) If set then the **top**[294] command series will strip adjacent empty lines and quotations.

ttycharset
: The character set of the terminal S-nail operates on, and the one and only supported character set that S-nail can use if no character set conversion capabilities have been compiled into it, in which case it defaults to ISO-8859-1. Otherwise it defaults to UTF-8. Sufficient locale support provided the default will be preferably deduced from the locale environment if that is set (for example LC_CTYPE[629], see there for more); runtime locale changes will be reflected by *ttycharset* except during the program startup phase and if **-S**[80] had been used to freeze the given value. Refer to the section **Character sets**[12] for the complete picture about character sets.

typescript-mode
: (Boolean) A special multiplex variable that disables all variables and settings which result in behaviour that interferes with running S-nail in script(1)[779]; it sets *colour-disable*[400], *line-editor-disable*[460] and (before startup completed only) *termcap-disable*[594]. Unsetting it does not restore the former state of the covered settings.

umask
: For a safe-by-default policy the process file mode creation mask umask(2)[780] will be set to 0077 on program startup after the resource files have been loaded, and unless this variable is set. By assigning this an empty value the active setting will not be changed, otherwise the given value will be made the new file mode creation mask. Child processes inherit the file mode creation mask of their parent.

user-HOST, user
: Variable chain that sets a global fallback user name, used in case none has been given in the protocol and account-specific URL. This variable defaults to the name of the user who runs S-nail.

v15-compat
: Enable upward compatibility with S-nail version 15.0 in respect to which configuration options are available and how they are handled. If set to a non-empty value (the default) the command modifier **wysh**[134] is implied and thus enforces **Shell-style argument quoting**[24] over **Old-style argument quoting**[23] for all commands which support both.

verbose
: Verbose mode enables logging of informational context messages. Historically a (Boolean) variable, this can either be set multiple times (what the command line option **-v**[86] uses), or be assigned a numeric value in order to increase verbosity. Assigning the value 0 disables verbosity and thus (almost) equals **unset**[272]. The maximum number is 3. Also see*deb ug*[412].

version, *version-date*, *version-hexnum*, *version-major*, *version-minor*, *version-update*
: (Read-only) S-nail version information: the first variable is a string with the complete version identification, the second the release date in ISO 8601 notation without time. The third is a 32-bit hexadecimal number with the upper 8 bits storing the major, followed by the minor and update version numbers which occupy 12 bits each. The latter three variables contain only decimal digits: the major, minor and update version numbers. The output of the command **version**[305] will include this information.

writebackedited
: If this variable is set messages modified using the **edit**[189] or **visual**[308] commands are written back to the current folder when it is quit; it is only honoured for writable folders in MBOX format, though. Note that the editor will be pointed to the raw message content in that case, i.e., neither MIME decoding nor decryption will have been performed, and proper *mbox-rfc4155*[470] From_ quoting of newly added or edited content is also left as an exercise to the user.

ENVIRONMENT

The term “environment variable” should be considered an indication that these variables are either standardized as vivid parts of process environments, or are commonly found in there. There is a strict separation in between **INTERNAL VARIABLES[31]** and the process environment that is inherited from the SHELL[643] upon program startup (and passed along to subprocesses). The separation can be resolved, meaning variables can transparently be used like **INTERNAL VARIABLES[31]**: when **set[271]** or **unset[272]** the process environment is updated automatically, for example; this includes change scope coverage via **local[130]**. (Removal requires sufficient system support: available in BSD since 1987, standardized since Y2K.) The list of resolved built-in variables follows, the command **environ[193]** can be used to resolve other variables. **varshow[303]**, without arguments and in *verbose[616]* mode, lists the resolve state.

COLUMNS
: The user’s preferred width in column positions for the terminal screen. Queried and used once on program startup in interactive or batch (**-# [90]**) mode on a (pseudo-) terminal. Actively managed (SIGWINCH) for child processes and the **MLE[93]** (**On terminal control and line editor[15]**) in interactive mode thereafter. Non-interactive mode always uses, and the fallback default is a compile-time constant, by default 80 columns. If in batch mode (on (pseudo-) terminal) COLUMNS and LINES[631] are both set but not both are usable (empty, not a number, or 0) at program startup, then the real terminal screen size will be (tried to be) determined once. (Normally the SHELL[643] manages these variables, and unsets them for pipe specifications etc.)

DEAD
: The name of the (mailbox) **folder[203]** to use for saving aborted messages if *save[546]* is set; this defaults to ~/dead.letter. If the variable *debug[412]* is set no output will be generated, otherwise the contents of the file will be replaced. Except shell globs **Filename transformations[28]** (also see **folder[203]**) will be performed.

EDITOR
: Pathname of the text editor to use for the **edit[189]** command and **~e[328]** (see **COMMAND ESCAPES[30]**); VISUAL[650] is used for a more display oriented editor.

HOME
: The user’s home directory. This variable is only used when it resides in the process environment. The calling user’s home directory will be used instead if this directory does not exist, is not accessible or cannot be read; it will always be used for the root user. (No test for being writable is performed to allow usage by non-privileged users within read-only jails, but dependent on settings this directory is a default write target for, for example, DEAD[625], MBOX[638] and more.)

LC_ALL, LC_CTYPE, LANG
: [Option] The (names in lookup order of the) locale(7)[781] (and / or see setlocale(3)[782]) which indicates the used **Character sets[12]**. Runtime changes trigger automatic updates of the entire locale system, which includes updating *ttycharset[611]* (except during startup if the variable has been frozen via **-S [80]**).

LINES
: The user’s preferred number of lines for the terminal screen. The behaviour is as described for COLUMNS[624], yet the compile-time constant used in non-interactive mode and as a fallback defaults to 24 (lines).

LISTER
: Pathname of the directory lister to use in the **folders[205]** command when operating on local mailboxes. Default is ls(1)[783] (path search through SHELL[643]).

LOGNAME
: Upon startup S-nail will actively ensure that this variable refers to the name of the user who runs S-nail, in order to be able to pass a verified name to any newly created child process.

MAIL
: Is used as the user’s **primary system mailbox[136]** unless *inbox[453]* is set. If the environmental fallback is also not set, a built-in compile-time default is used. This is assumed to be an absolute pathname.

`MAILCAPS`
: [Option] Override the default path search of **The Mailcap files[38]**: any existing file therein will be loaded in sequence, appending any content to the list of MIME type handler directives. The RFC 1524 standard imposed default value is assigned otherwise: `~/.mailcap:/etc/mailcap:/usr/etc/mailcap:/usr/local/etc/mailcap`. (The default value is a compile-time [Option].)

`MAILRC`
: Is used as a startup file instead of `~/.mailrc[660]` if set. In order to avoid side-effects from configuration files scripts should either set this variable to `/dev/null[657]` or the **`-:`** [56] command line option should be used.

`MAILX_NO_SYSTEM_RC`
: If this variable is set then reading of `s-nail.rc[659]` (aka *system-mailrc[591]*) at startup is inhibited, i.e., the same effect is achieved as if S-nail had been started up with the option **`-:`** [56] (and according argument) or **`-n`[76]**. This variable is only used when it resides in the process environment.

`MBOX`
: The name of the user's **secondary mailbox[137]** file. A logical subset of the special **Filename transformations[28]** (also see **`folder`[203]**) are supported. The default is `~/mbox[653]`. Traditionally this MBOX is used as the file to save messages from the **primary system mailbox[136]** that have been read. Also see **Message states[13]**.

`NETRC`
: [Option] This variable overrides the default location of the user's `~/.netrc[656]` file.

`PAGER`
: Pathname of the paging program backing **`more`[237]** and *crt[408]* induced pager usage. The default paginator is `more`(1)[784] (path search through `SHELL[643]`).

  The content of this variable is inspected: if it contains "less" then a non-existing environment variable `LESS` is temporarily set to the portable `RIFE` (the latter two excessively), whereas for "lv" `LV` will temporarily be set to '`-c`'. Also see**Colour ed display[16]** and *colour-disable[400]*.

`PATH`
: A colon-separated list of directories that is searched by the shell when looking for commands, for example `/bin:/usr/bin:/usr/local/bin`.

`POSIXLY_CORRECT`
: This environment entry is automatically squared with *posix[523]*.

`SHELL`
: The POSIX compatible `sh`(1)[785] to use for the commands **`!`[138]**, **`pipe`[248]**, and **`shell`[275]**, the **COMMAND ESCAPES[30]** **`~!`[314]** and **`~|`[321]**, and when starting subprocesses. A compile-time default shell is used if this variable is not defined. `$SHELL -c -- 'echo du'` is required to print '`du`' (POSIX issue #1440).

`SOCKS5_PROXY`
: This environment entry is automatically squared with *socks-proxy[578]*.

`SOURCE_DATE_EPOCH`
: Specifies a time in seconds since the Unix epoch (1970-01-01) to be used in place of the current time. This variable is looked up upon program startup, and its existence will switch S-nail to a reproducible mode (**https://reproducible-builds.org**) which uses deterministic random numbers, a special fixated pseudo `LOGNAME[633]` and more. This operation mode is used for development and by software packagers. [v15 behaviour may differ] Currently an invalid setting is only ignored, rather than causing a program abortion.

  ```
  $ SOURCE_DATE_EPOCH=‘date +%s‘ s-nail
  ```

`TERM`
: [Option] The terminal type for which output is to be prepared. For extended colour and font control refer to **Coloured display[16]**, and for terminal management in general to **On terminal control and line editor[15]**.

`TMPDIR` Except for the root user this variable defines the directory for temporary files to be used instead of `/tmp` (or the given compile-time constant) if set, existent, accessible as well as read- and writable. This variable is only used when it resides in the process environment, but S-nail will ensure at startup that this environment variable is updated to contain a usable temporary directory.

`TZ` Defines the `timezone`(3)[786] for local standard time and `date`(1)[787]. (System-dependent, often there is also a `/etc/localtime`.)

`USER` Identical to `LOGNAME`[633] (see there), but this variable is not standardized, should therefore not be used, and is only corrected if already set.

`VISUAL` Pathname of the text editor to use for the **`visual`**[308] command and **`~v`**[346] (see **COMMAND ESCAPES**[30]); `EDITOR`[626] is used for a less display oriented editor.

FILES

`~/.mailcap`, `/etc/mailcap`
: [Option] Personal and system-wide MIME type handler definition files, see **The Mailcap files**[38]. (The shown names are part of the RFC 1524 standard search path `MAILCAPS`[635].)

`~/.mailrc`[660], `s-nail.rc`[659]
: User-specific and system-wide files giving initial commands, the **Resource files**[36]. (The used filenames come from *MAILRC* and *system-mailrc*[591], respectively.)

`~/mbox` The default value for `MBOX`[638].

`~/.mime.types`, `/etc/mime.types`
: Personal and system-wide MIME types, see **The mime.types files**[37].

`~/.netrc`
: [Option] The default location of the user's `.netrc` file – the section **The .netrc file**[39] documents the file format. The used path can be set via `NETRC`[639].

`/dev/null`
: The data sink `null`(4)[788].

`~/.rnd` [Option] Possible location for persistent random entropy seed storage, see *tls-rand-file*[607].

Resource files

Upon startup several resource files are read, in order:

`s-nail.rc`
: System-wide resource file (*system-mailrc*[591]). Reading of this file can be suppressed, either by using the **`-:`** [56] (and according argument) or **`-n`** [76] command line options, or by setting the **ENVIRONMENT**[34] variable `MAILX_NO_SYSTEM_RC`[637]. A(n unmodified) version of this is also compiled-in, accessible via **`-:`** [56] '`x`'.

`~/.mailrc`
: File giving initial commands. A different file can be chosen by setting the **ENVIRONMENT**[34] variable `MAILRC`[636]. Reading of this file can be suppressed with the**`-:`** [56] command line option.

mailx-extra-rc[467]
: Defines a startup file to be read after all other resource files. It can be used to specify settings that are not understood by other `mailx`(1)[789] implementations, for example.

The content of these files is interpreted as for **COMMANDS**[21], with the exception that **`history`**[216] is never tracked. Errors while loading these files are subject to the settings of*err exit*[419] and *posix*[523]. More files with syntactically equal content can be **`source`**[284]ed. The following, saved in a file, would

be an examplary content:

```
 # This line is a comment command.  And y\
    es, it is really continued here.
set debug \
    verbose=2
    set editheaders
```

The mime.types files

As stated in **HTML mail and MIME attachments[10]** MIME (Multipurpose Internet Mail Extensions) media types needs to be registered in order to be able to classify message and attachment content. One source for them are `mime.types` files, the loading of which can be controlled by setting the variable *mimetypes-load-control[481]*. Another is the command**`mimetype` [227]**, which also offers access to the MIME type cache. `mime.types` files have the following syntax:

```
type/subtype extension [extension ...]
# For example: text/html html htm
```

where `type/subtype` denotes the MIME media type, as standardized in RFC 2046: `type` is used to declare the general type of data, while the `subtype` specifies a specific format for that type of data. One or multiple filename `extension`s, separated by whitespace, and specified without leading dot '`.`', can be bound to the media type format. Comments may be introduced anywhere on a line with a number sign '#', causing the remaining line to be discarded.

An extended (non-portable) syntax that prepends an optional `type-marker` to the above is offered by the command **`mimetype`[227]**, and is supported also in especially crafted files which can be loaded via the alternative value syntax of *mimetypes-load-control[481]*:

```
[?type-marker ]type/subtype extension [extension ...]
```

The following type-markers are supported; they are mutual exclusive unless documented otherwise:

t Treat this media type as plain text; the *t* is actually optional.

h Treat message parts with this content as HTML tagsoup. If the [Option]al HTML-tagsoup-to-text converter is not available treat the content as plain text instead.

H Likewise *h*, but instead of falling back to plain text require an explicit MIME content handler.

q If no handler can be found a text message is displayed which says so. This can be annoying, for example signatures serve a contextual purpose, their content is of no use by itself. This marker will avoid displaying the text message.

* The given MIME media type shall only be matched when looking for handlers, but not when classifying content to create messages. *t* is no longer optional when this is used. IAN A MIME registry standards do not know about "extension chains": `tar.gz` is thus a `gzip`(1)[790] compressed file. Because MIME media type handlers, like those defined in **The Mailcap files[38]**, match media types, non-standardized fictional types like `x-tar-gz` are used in the wild as MIME environments become configured. The sane solution of recursively unpacking until no more MIME media type unpacking is possible is not available. This flag may be used alongside other type-markers, and is especially useful in conjunction with *mime-counter-evidence[478]*.

When classifying all registered MIME types are searched, and the longest matching extension will be used. A filename of only an `extension` will match, for example `README` is matched by `mimetype ? text/unix-readme NEWS README`, empty filenames are not matched, so for example `.x.tar` does not match `application/x-fun x.tar` but rather `application/x-tar tar`.

Further reading: for sending messages: **`mimetype`[227]**, *mime-allow-text-controls[476]*, *mimetypes-load-control[481]*. For reading etc. messages: **HTML mail and MIME attachments[10]**, **The Mailcap files[38]**, **`mimetype`[227]**, *mime-counter-evidence[478]*, *mimetypes-load-control[481]*,

pipe-TYPE/SUBTYPE[511], *pipe-EXTENSION[510]*.

The Mailcap files

[Option] RFC 1524 defines a “User Agent Configuration Mechanism” to be used to inform mail user agent programs about the locally installed facilities for handling various data formats, i.e., about commands and how they can be used to display, edit et cetera MIME part contents, as well as a default path search that includes multiple possible locations of resource files, and the `MAILCAPS[635]` environment variable to overwrite that. Handlers found from doing the path search will be cached, the command **`mailcap[225]`** operates on that cache, and the variable *mailcap-disable[466]* will suppress automatic loading, and usage of any mailcap handlers. **HTML mail and MIME attachments [10]** gives a general overview of how MIME types are handled.

“Mailcap” files consist of a set of newline separated entries. Comment lines start with a number sign ‘#’ (in the first column!) and are ignored. Empty lines are ignored. All other lines are interpreted as mailcap entries. An entry definition may be split over multiple lines by placing the reverse solidus character ‘\’ last in all but the final line. The standard does not specify how leading whitespace of successive lines is to be treated, therefore they are retained.

“Mailcap” entries consist of a number of semicolon ‘`;`’ separated fields. The first two fields are mandatory and must occur in the specified order, the remaining fields are optional and may appear in any order. Leading and trailing whitespace of field content is ignored (removed). The reverse solidus ‘\’ character can be used to escape any following character including semicolon and itself in the content of the second field, and in value parts of any optional key/value field.

The first field defines the MIME `TYPE/SUBTYPE` the entry is about to handle (case-insensitively). If the subtype is specified as an asterisk ‘`*`’ the entry is meant to match all subtypes of the named type, e.g., `audio/*` would match any audio type. The second field is the **`view`** shell command used to display MIME parts of the given type.

Data consuming shell commands will be fed message (MIME part) data on standard input unless one or more instances of the (unquoted) string ‘`%s`’ are used: these formats will be replaced with a temporary file(name) that has been prefilled with the parts data. Data producing shell commands are expected to generata data on their standard output unless that format is used. In all cases any given ‘`%s`’ format is replaced with a properly shell quoted filename. When a command requests a temporary file via ‘`%s`’ then that will be removed again, as if the **`x-mailx-tmpfile[666]`** and **`x-mailx-tmpfile-fill[667]`** flags had been set; unless the command requests **`x-mailx-async[663]`** the **`x-mailx-tmpfile-unlink[668]`** flag is also implied; see below for more.

Optional fields define single-word flags (case-insensitive), or key / value pairs consisting of a case-insensitive keyword, an equals sign ‘`=`’, and a shell command; whitespace surrounding the equals sign is removed. Optional fields include the following:

`compose`
: A program that can be used to compose a new body or body part in the given format. (Currently unused.)

`composetyped`
: Similar to the **`compose`** field, but is to be used when the composing program needs to specify the `Content-type:` header field to be applied to the composed data. (Currently unused.)

`copiousoutput`
: A flag field which indicates that the output of the **`view`** command is integrable into S-nails normal visual display. It is mutually exclusive with **`needsterminal[662]`**.

description
A textual description that describes this type of data. The text may optionally be enclosed within double quotation marks ‘"’.

edit A program that can be used to edit a body or body part in the given format. (Currently unused.)

nametemplate
This field specifies a filename format for the ‘%s’ format used in the shell command fields, in which ‘%s’ will be replaced by a random string. (The filename is also stored in and passed to subprocesses via MAILX_FILENAME_TEMPORARY[517].) The standard says this is “only expected to be relevant in environments where filename extensions are meaningful”, and so this field is ignored unless the ‘%s’ is a prefix, optionally followed by (ASCII) alphabetic and numeric characters, the underscore and the period. For example, to specify that a JPG file is to be passed to an image viewer with a name ending in .jpg, nametemplate=%s.jpg can be used.

needsterminal
This flag field indicates that the given shell command must be run on an interactive terminal. S-nail will temporarily release the terminal to the given command in interactive mode, in non-interactive mode this entry will be entirely ignored; this flag implies **x-mailx-noquote[664]**.

print A program that can be used to print a message or body part in the given format. (Currently unused.)

test Specifies a program to be run to test some condition, for example, the machine architecture, or the window system in use, to determine whether or not this mailcap entry applies. If the test fails, a subsequent mailcap entry should be sought; also see **x-mailx-test-once[665]**. Standard I/O of the test program is redirected from and to /dev/null[657], and the format ‘%s’ is not supported (the data does not yet exist).

textualnewlines
A flag field which indicates that this type of data is line-oriented and that, if encoded in base64, all newlines should be converted to canonical form (CRLF) before encoding, and will be in that form after decoding. (Currently unused.)

x11-bitmap
Names a file, in X11 bitmap (xbm) format, which points to an appropriate icon to be used to visually denote the presence of this kind of data. This field is not used by S-nail.

x-mailx-async
Extension flag field that denotes that the given **view** command shall be executed asynchronously, without blocking S-nail. Cannot be used in conjunction with **needsterminal[662]**; the standard output of the command will go to /dev/null[657].

x-mailx-noquote
An extension flag field that indicates that even a **copiousoutput[661] view** command shall not be used when *quote[528]*ing messages, as it would by default.

x-mailx-test-once
Extension flag which denotes whether the given **test** command shall be evaluated once only with its exit status being cached. This is handy if some global unchanging condition is to be queried, like “running under the X Window System”.

x-mailx-tmpfile
Extension flag field that requests creation of a zero-sized temporary file, the name of which is to be placed in the environment variable MAILX_FILENAME_TEMPORARY[517]. It is an error to use this flag with commands that include a ‘%s’ format (because that is implemented by means of this temporary file).

x-mailx-tmpfile-fill
: Normally the MIME part content is passed to the handler via standard input; if this flag is set then the data will instead be written into the implied **x-mailx-tmpfile**[666]. In order to cause deletion of the temporary file you will have to set **x-mailx-tmpfile-unlink**[668] explicitly! It is an error to use this flag with commands that include a '%s' format.

x-mailx-tmpfile-unlink
: Extension flag field that requests that the temporary file shall be deleted automatically when the command loop is entered again at latest. It is an error to use this flag with commands that include a '%s' format, or in conjunction with**x-mailx-async** [663]. **x-mailx-tmpfile**[666] is implied.

x-mailx-last-resort
: An extension flag that indicates that this handler shall only be used as a last resort, when no other source (see **HTML mail and MIME attachments**[10]) provides a MIME handler.

x-mailx-ignore
: An extension that enforces that this handler is not used at all.

The standard includes the possibility to define any number of additional fields, prefixed by 'x-'. Flag fields apply to the entire "Mailcap" entry — in some unusual cases, this may not be desirable, but differentiation can be accomplished via separate entries, taking advantage of the fact that subsequent entries are searched if an earlier one does not provide enough information. For example, if a **view** command needs to specify the **needsterminal**[662] flag, but the **compose** command shall not, the following will help out the latter:

```
application/postscript; ps-to-terminal %s; needsterminal
application/postscript; ps-to-terminal %s; compose=idraw %s
```

In value parts of command fields any occurrence of the format string '%t' will be replaced by the TYPE/SUBTYPE specification. Any named parameter from a messages' Content-type: field may be embedded into the command line using the format '%{' followed by the parameter name and a closing brace '}' character. The entire parameter should appear as a single command line argument, regardless of embedded spaces, shell quoting will be performed by the RFC 1524 processor, thus:

```
# Message
Content-type:  multipart/mixed; boundary=42

# Mailcap file
multipart/*; /usr/local/bin/showmulti \
  %t %{boundary}  ;  composetyped  = /usr/local/bin/makemulti

# Executed shell command
/usr/local/bin/showmulti multipart/mixed 42
```

Note that S-nail does not support handlers for multipart MIME parts as shown in this example (as of today). It does not support the additional formats '%n' and '%F'. An example file, also showing how to properly deal with the expansion of '%s', which includes any quotes that are necessary to make it a valid shell argument by itself and thus will cause undesired behaviour when placed in additional user-provided quotes:

```
# Comment line
text/richtext; richtext %s; copiousoutput

text/x-perl; perl -cWT %s; nametemplate = %s.pl

# Exit EX_TEMPFAIL=75 on signal
application/pdf; \
```

```
  infile=%s\; \
    trap "rm -f ${infile}" EXIT\; \
    trap "exit 75" INT QUIT TERM\; \
    mupdf "${infile}"; \
  test = [ -n "${DISPLAY}" ]; \
  nametemplate = %s.pdf; x-mailx-async
application/pdf; pdftotext -layout - -; copiousoutput

application/*; echo "This is \\"%t\\" but \
    is 50 \% Greek to me" \; < %s head -c 512 | cat -vet; \
  copiousoutput; x-mailx-noquote; x-mailx-last-resort
```

Further reading: **HTML mail and MIME attachments[10]**, **The mime.types files[37]**, **`mimetype`[227]**, `MAILCAPS`[635], *mime-counter-evidence[478]*, *pipe-TYPE/SUBTYPE[511]*, *pipe-EXTENSION[510]*.

The .netrc file

User credentials for machine accounts (see **On URL syntax and credential lookup[18]**) can be placed in the `.netrc` file, which will be loaded and cached when requested by *netrc-lookup[489]*. The default location `~/.netrc`[656] may be overridden by the `NETRC`[639] environment variable. As long as syntax constraints are honoured the file source may be replaced with the output of the shell command set in *netrc-pipe[490]*, to load an encrypted file, for example. The cache can be managed with the command **`netrc`[239]**.

The file consists of space, tabulator or newline separated tokens. This parser implements a superset of the original BSD syntax, but users should nonetheless be aware of portability glitches, shall their `.netrc` be usable across multiple programs and platforms:

- BSD only supports double quotation marks, for example `password "pass with spaces"`.
- BSD (only?) supports escaping of single characters via a reverse solidus (a space could be escaped via ‘\ ’), in- as well as outside of a quoted string. This method is assumed to be present, and will actively be used to quote double quotation marks ‘"’ and reverse solidus ‘\’ characters inside the **`login`** and **`password`** tokens, for example for display purposes.
- BSD does not require a final quotation mark of the last user input token.
- The original BSD (Berknet) parser also supported a format which allowed tokens to be separated with commas – whereas at least Hewlett-Packard still seems to support this syntax, this parser does not!
- As a non-portable extension some widely-used programs support shell-style comments: if an input line starts, after any amount of whitespace, with a number sign ‘#’, then the rest of the line is ignored.
- Whereas other programs may require that the `.netrc` file is accessible by only the user if it contains a **`password`** token for any other **`login`** than “anonymous”, this parser will always require these strict permissions.

Of the following list of supported tokens this parser uses (and caches) **`machine`**, **`login`** and **`password`**. An existing **`default`** entry will not be used.

`machine` *`name`*

The hostname of the entries’ machine, lowercase-normalized before use. Any further file content, until either end-of-file or the occurrence of another **`machine`** or a **`default`** first-class token is bound (only related) to the machine *`name`*.

As an extension that should not be the cause of any worries this parser supports a single wildcard prefix for *`name`*:

```
machine *.example.com login USER password PASS
machine pop3.example.com login USER password PASS
machine smtp.example.com login USER password PASS
```

which would match `xy.example.com` as well as `pop3.example.com`, but neither `example.com` nor `local.smtp.example.com`. In the example neither `pop3.example.com` nor `smtp.example.com` will be matched by the wildcard, since the exact matches take precedence (it is however faster to specify it the other way around).

`default`
This is the same as **`machine`** except that it is a fallback entry that is used shall none of the specified machines match; only one default token may be specified, and it must be the last first-class token.

`login` *`name`*
The user name on the remote machine.

`password` *`string`*
The user's password on the remote machine.

`account` *`string`*
Supply an additional account password. This is merely for FTP purposes.

`macdef` *`name`*
Define a macro. A macro is defined with the specified *`name`*; it is formed from all lines beginning with the next line and continuing until a blank line is (consecutive newline characters are) encountered. (Note that**`macdef`** entries cannot be utilized by multiple machines, too, b ut must be defined following the **`machine`** they are intended to be used with.) If a macro named *`init`* exists, it is automatically run as the last step of the login process. This is merely for FTP purposes.

EXAMPLES

S/MIME step by step

[Option] The first thing that is needed for **Signed and encrypted messages with S/MIME[17]** is a personal certificate, and a private key. The certificate contains public information, in particular a name and email address(es), and the public key that can be used by others to encrypt messages for the certificate holder (the owner of the private key), and to **`verify`[304]** signed messages generated with that certificate('s private key). Whereas the certificate is included in each signed message, the private key must be kept secret. It is used to decrypt messages that were previously encrypted with the public key, and to sign messages.

For personal use it is recommended to get a S/MIME certificate from one of the major CAs on the Internet. Many CAs offer such certificates for free. Usually offered is a combined certificate and private key in PKCS#12 format which S-nail does not accept directly. To convert it to PEM format, the following shell command can be used; Read on for how to use these PEM files.

```
$ openssl pkcs12 -in cert.p12 -out certpem.pem -clcerts -nodes
$ # Alternatively
$ openssl pkcs12 -in cert.p12 -out cert.pem -clcerts -nokeys
$ openssl pkcs12 -in cert.p12 -out key.pem -nocerts -nodes
```

There is also **https://www.CAcert.org** which issues client and server certificates to members of their community for free; their root certificate (**https://www.cacert.org/certs/root.crt**) is often not in the default set of trusted CA root certificates, though, which means their root certificate has to be downloaded separately, and needs to be part of the S/MIME certificate validation chain by including it in *smime-ca-dir[559]* or as a vivid member of the *smime-ca-file[560]*. But let us take a step-by-step tour on how to setup S/MIME with a certificate from CAcert.org despite this situation!

First of all you will have to become a member of the CAcert.org community, simply by registrating yourself via the web interface. Once you are, create and verify all email addresses you want to be able to create signed and encrypted messages for/with using the corresponding entries of the web interface. Now ready to create S/MIME certificates, so let us create a new "client certificate", ensure to include all email addresses

that should be covered by the certificate in the following web form, and also to use your name as the "common name".

Create a private key and a certificate request on your local computer (see the manual pages of the used commands for more in-depth knowledge on what the used arguments etc. do):

```
$ openssl req -nodes -newkey rsa:4096 -keyout key.pem -out creq.pem
```

Afterwards copy-and-paste the content of "creq.pem" into the certificate-request (CSR) field of the web form on the CAcert.org website (you may need to unfold some "advanced options" to see the corresponding text field). This last step will ensure that your private key (which never left your box) and the certificate belong together (through the public key that will find its way into the certificate via the certificate-request). You are now ready and can create your CAcert certified certificate. Download and store or copy-and-paste it as "pub.crt".

Yay. In order to use your new S/MIME setup a combined private key/public key (certificate) file has to be created:

```
$ cat key.pem pub.crt > ME@HERE.com.paired
```

This is the file S-nail will work with. If you have created your private key with a passphrase then S-nail will ask you for it whenever a message is signed or decrypted, unless this operation has been automated as described in **Signed and encrypted messages with S/MIME[17]**. Set the following variables to henceforth use S/MIME (setting *smime-ca-file[560]* is of interest for verification only):

```
? set smime-ca-file=ALL-TRUSTED-ROOT-CERTS-HERE \
    smime-sign-cert=ME@HERE.com.paired \
    smime-sign-digest=SHA512 \
    smime-sign from=myname@my.host
```

Using CRLs with S/MIME or TLS

[Option] Certification authorities (CAs) issue certificate revocation lists (CRLs) on a regular basis. These lists contain the serial numbers of certificates that have been declared invalid after they have been issued. Such usually happens because the private key for the certificate has been compromised, because the owner of the certificate has left the organization that is mentioned in the certificate, etc. To seriously use S/MIME or TLS verification, an up-to-date CRL is required for each trusted CA. There is otherwise no method to distinguish between valid and invalidated certificates. S-nail currently offers no mechanism to fetch CRLs, nor to access them on the Internet, so they have to be retrieved by some external mechanism.

S-nail accepts CRLs in PEM format only; CRLs in DER format must be converted, like, e. g.:

```
$ openssl crl -inform DER -in crl.der -out crl.pem
```

To tell S-nail about the CRLs, a directory that contains all CRL files (and no other files) must be created. The *smime-crl-dir[564]* or *tls-crl-dir[602]* variables, respectively, must then be set to point to that directory. After that, S-nail requires a CRL to be present for each CA that is used to verify a certificate.

FAQ

In general it is a good idea to turn on *debug[412]* (**-d [64]**) and / or *verbose[616]* (**-v [86]**, twice) if something does not work well. Very often a diagnostic message can be produced that leads to the problems' solution.

S-nail shortly hangs on startup

This can have two reasons, one is the necessity to wait for a file lock and cannot be helped, the other being that S-nail calls the function `uname`(2)[791] in order to query the nodename of the box (sometimes the real one is needed instead of the one represented by the internal variable *hostname[447]*). One may have varying success by ensuring that the real hostname and `localhost` have entries in `/etc/hosts`, or, more

generally, that the name service is properly setup – and does `hostname`(1)[792] return the expected value? Does this local hostname have a domain suffix? RFC 6762 standardized the link-local top-level domain `.local`, try again after adding an (additional) entry with this extension.

I cannot login to Google mail (via OAuth)

Since 2014 some free service providers classify programs as “less secure” unless they use a special authentication method (OAuth 2.0) which was not standardized for non-HTTP protocol authentication token query until August 2015 (RFC 7628).

Different to Kerberos / GSSAPI, which is developed since the mid of the 1980s, where a user can easily create a local authentication ticket for her- and himself with the locally installed `kinit`(1)[793] program, that protocol has no such local part but instead requires a world-wide-web query to create or fetch a token; since there is no local cache this query would have to be performed whenever S-nail is invoked (in interactive sessions situation may differ).

S-nail does not directly support OAuth. It, however, supports XOAUTH2 / OAUTHBEARER, see **But, how about XOAUTH2 / OAUTHBEARER?[46]** If that is not used it is necessary to declare S-nail a “less secure app” (on the providers account web page) in order to read and send mail. However, it also seems possible to take the following steps instead:

1. give the provider the number of a mobile phone,
2. enable “2-Step Verification”,
3. create an application specific password (16 characters), and
4. use that special password instead of the real Google account password in S-nail (for more on that see the section **On URL syntax and credential lookup[18]**).

But, how about XOAUTH2 / OAUTHBEARER?

Following up **I cannot login to Google mail (via OAuth)[45]** one OAuth-based authentication method is available: the OAuth 2.0 bearer token usage as standardized in RFC 6750 (according SASL mechanism in RFC 7628), also known as XOAUTH2 and OAUTHBEARER, allows fetching a temporary access token via the web that can locally be used as a *password[508]*. The protocol is simple and extendable, token updates or even password changes via a simple TLS secured server login would be possible in theory, but today a web browser and an external support tool are prerequisites for using this authentication method. The token times out and must be refreshed periodically via web.

Some hurdles must be taken before being able to use this method. Using GMail as an example, an application (that is a name) must be registered, for which credentials, a “client ID” and a “client secret”, need to be created and saved locally (in a secure way). These initial configuration steps can be performed at **https://console.developers.google.com/apis/credentials**. Thereafter a refresh token can be requested; a python program to do this for GMail accounts is **https://github.com/google/gmail-oauth2-tools/raw/master/python/oauth2.py**:

```
$ python oauth2.py --user=EMAIL \
  --client-id=THE-ID --client-secret=THE-SECRET \
  --generate_oauth2_token
To authorize token, visit this url and follow the directions:
  https://accounts.google.com/o/oauth2/auth?client_id=...
  Enter verification code: ...
  Refresh Token: ...
  Access Token: ...
  Access Token Expiration Seconds: 3600
$ # Of which the last three are actual token responses.
$ # Thereafter access tokens can regularly be refreshed
$ # via the created refresh token (read on)
```

The generated refresh token must also be saved locally (securely). The procedure as a whole can be read at **https://github.com/google/gmail-oauth2-tools/wiki/OAuth2DotPyRunThrough**. Since periodic timers are not yet supported, keeping an access token up-to-date (from within S-nail) can only be performed via the hook *on-main-loop-tick[503]*, or (for sending only) *on-compose-enter[496]* (for more on authentication see the section **On URL syntax and credential lookup[18]**):

```
set on-main-loop-tick=o-m-l-t on-compose-enter=o-c-e
define o-m-l-t {
  xcall update_access_token
}
define o-c-e {
  xcall update_access_token
}

set access_token_=0
define update_access_token {
  local set i epoch_sec epoch_nsec
  vput vexpr i epoch
  eval set $i # set epoch_sec/_nsec of vexpr epoch
  vput vexpr i + $access_token_ 2100
  if $epoch_sec -ge $i
    vput ! password python oauth2.py --user=EMAIL \
        --client-id=THE-ID --client-secret=THE-SECRET \
        --refresh-token=THE-REFRESH-TOKEN |\
      sed '1b PASS;d; :PASS s/^.\{1,\}:\(.\{1,\}\)$/\1/'
    vput csop password trim "$password"
    if -n "$verbose"
      echo password is <$password>
    endif
    set access_token_=$epoch_sec
  endif
}
```

Not "defunctional", but the editor key does not work

Maybe a sequence is shadowed; setting *debug[412]* or maximum *verbose[616]* causes a **bind[153]** tree dump after (re-)build (upon startup or after modifying bindings).

Or the terminal library (see **On terminal control and line editor[15]**, **bind[153]**, *termcap[592]*) reports different codes than TERM[646] generates, causing dysfunctional bindings because of mismatches. (One common source of this is that the — possibly even non-existing — keypad is not turned on, and the resulting layout reports codes for the normal keyboard keys.) The expected code sequences are shown by **bind[153]** in *verbose[616]* mode, the **MLE[93]** loggs the generated ones if *debug[412]* is set in conjunction with maximum *verbose[616]*. After detecting the correct codes these can be placed in*termcap [592]*. Here a hypothetic HOME key example (with redundancy removed):

```
? set verbose; bind*
# 1B 5B=[ 31=1 3B=; 32=2 48=H
  bind base :khome mle-go-home
? set verbose=3 debug
# ..pressing HOME key
? s-nail: \x1B/?
s-nail: \x5B/[
s-nail: \x48/H
```

```
#[..]
? set noverbose nodebug termcap='khome=\E[H'; bind*
# 1B 5B=[ 48=H
  bind base :khome mle-go-home
```

Can S-nail git-send-email?

Yes. Put (at least parts of) the following in your `~/.gitconfig`:

```
[sendemail]
smtpserver = /usr/bin/s-nail
smtpserveroption = -t
#smtpserveroption = -Sexpandaddr
smtpserveroption = -Athe-account-you-need
##
suppresscc = all
suppressfrom = false
assume8bitEncoding = UTF-8
#to = /tmp/OUT
confirm = always
chainreplyto = true
multiedit = false
thread = true
quiet = true
annotate = true
```

Newer `git`(1)[794] versions (v2.33.0) added the option **`sendmailCmd`**. Patches can also be send directly, for example:

```
$ git format-patch -M --stdout HEAD^ |
  s-nail -A the-account-you-need -t RECIPIENT
```

Howto handle stale dotlock files

`folder`[203] sometimes fails to open MBOX mail databases because creation of **dotlock files**[204] is impossible due to existing but unowned lock files. S-nail does not offer an option to deal with those files, because it is considered a site policy what counts as unowned, and what not. The site policy is usually defined by administrator(s), and expressed in the configuration of a locally installed MTA (for example Postfix `stale_lock_time=500s`). Therefore the suggestion:

```
$ </dev/null s-nail -s 'MTA: be no frog, handle lock' $LOGNAME
```

By sending a mail to yourself the local MTA can use its normal queue mechanism to try the delivery multiple times, finally decide a lock file has become stale, and remove it.

IMAP CLIENT

[Option]ally there is IMAP client support available. This part of the program is obsolete and will vanish in v15 with the large MIME and I/O layer rewrite, because it uses old-style blocking I/O and makes excessive use of signal based long code jumps. Support can hopefully be readded later based on a new-style I/O, with SysV signal handling. In fact the IMAP support had already been removed from the codebase, but was reinstantiated on user demand: in effect the IMAP code is at the level of S-nail v14.8.16 (with **`imapcodec`[675]** being the sole exception), and should be treated with some care.

IMAP uses the `imap://` and `imaps://` protocol prefixes, and an IMAP-based *folder*[427] may be used. IMAP URLs (paths) undergo inspections and possible transformations before use (and the command **`imapcodec`[675]** can be used to manually apply them to any given argument). Hierarchy delimiters are normalized, a step which is configurable via the *imap-delim*[679] variable chain, but defaults to the first seen

delimiter otherwise. S-nail supports internationalised IMAP names, and en- and decodes the names from and to the *ttycharset[611]* as necessary and possible. If a mailbox name is expanded (see **Filename transformations[28]**) to an IMAP mailbox, all names that begin with '+' then refer to IMAP mailboxes below the *folder[427]* target box, while folder names prefixed by '@' refer to folders below the hierarchy base, so the following will list all folders below the current one when in an IMAP mailbox: `folders @`.

Note: some IMAP servers do not accept the creation of mailboxes in the hierarchy base, but require that they are created as subfolders of 'INBOX' – with such servers a folder name of the form

```
imaps://me@imap.myisp.example/INBOX.
```

should be used (the last character is the server's hierarchy delimiter). The following IMAP-specific commands exist:

`cache` Only applicable to cached IMAP mailboxes; takes a message list and reads the specified messages into the IMAP cache.

`connect`
: If operating in disconnected mode on an IMAP mailbox, switch to online mode and connect to the mail server while retaining the mailbox status. See the description of the *disconnected[676]* variable for more information.

`disconnect`
: If operating in online mode on an IMAP mailbox, switch to disconnected mode while retaining the mailbox status. See the description of the *disconnected[676]* variable for more. A list of messages may optionally be given as argument; the respective messages are then read into the cache before the connection is closed, thus `disco *` makes the entire mailbox available for disconnected use.

`imap` Sends command strings directly to the current IMAP server. S-nail operates always in IMAP 'selected state' on the current mailbox; commands that change this will produce undesirable results and should be avoided. Useful IMAP commands are:

create	Takes the name of an IMAP mailbox as an argument and creates it.
getquotaroot	(RFC 2087) Takes the name of an IMAP mailbox as an argument and prints the quotas that apply to the mailbox. Not all IMAP servers support this command.
namespace	(RFC 2342) Takes no arguments and prints the Personal Namespaces, the Other User's Namespaces and the Shared Namespaces. Each namespace type is printed in parentheses; if there are multiple namespaces of the same type, inner parentheses separate them. For each namespace a prefix and a hierarchy separator is listed. Not all IMAP servers support this command.

`imapcodec`
: Perform IMAP path transformations. Supports **`vput[132]`** (see **Command modifiers[22]**), and manages the error number *!*[351]. The first argument specifies the operation: `e[ncode]` normalizes hierarchy delimiters (see *imap-delim[679]*) and converts the strings from the locale *ttycharset[611]* to the internationalized variant used by IMAP, `d[ecode]` performs the reverse operation. Encoding will honour the (global) value of *imap-delim[679]*.

The following IMAP-specific internal variables exist:

disconnected
: (Boolean) When an IMAP mailbox is selected and this variable is set, no connection to the server is initiated. Instead, data is obtained from the local cache (see *imap-cache[678]*). Mailboxes that are not present in the cache and messages that have not yet entirely been fetched from the server

are not available; to fetch all messages in a mailbox at once, the command '`copy * /dev/null`' can be used while still in connected mode. Changes that are made to IMAP mailboxes in disconnected mode are queued and committed later when a connection to that server is made. This procedure is not completely reliable since it cannot be guaranteed that the IMAP unique identifiers (UIDs) on the server still match the ones in the cache at that time. Data is saved to `DEAD`[625] when this problem occurs.

disconnected-USER@HOST
The specified account is handled as described for the *disconnected*[676] variable above, but other accounts are not affected.

imap-auth-USER@HOST, *imap-auth*
Sets the IMAP authentication method. Supported are the default `login` (called `plain` by some servers), `oauthbearer` (see **FAQ**[43] entry **But, how about XOAUTH2 / OAUTHBEARER?**[46]), `external` and `externanon` (for TLS secured connections which pass a client certificate via *tls-config-pairs*[601]), as well as the [Option]al `cram-md5` and `gssapi`. All methods need a*user* [614] and a *password*[508] except `gssapi` and `external`, which only need the former. `externanon` only uses data from the client certificate.

imap-cache
Enables caching of IMAP mailboxes. The value of this variable must point to a directory that is either existent or can be created by S-nail. All contents of the cache can be deleted by S-nail at any time; it is not safe to make assumptions about them.

imap-delim-USER@HOST, *imap-delim-HOST*, *imap-delim*
The hierarchy separator used by the IMAP server. Whenever an IMAP path is specified it will undergo normalization. One of the normalization steps is the squeezing and adjustment of hierarchy separators. If this variable is set, any occurrence of any character of the given value that exists in the path will be replaced by the first member of the value; an empty value will cause the default to be used, it is '`/.`'. If not set, we will reuse the first hierarchy separator character that is discovered in a user-given mailbox name.

imap-keepalive-USER@HOST, *imap-keepalive-HOST*, *imap-keepalive*
IMAP servers may close the connection after a period of inactivity; the standard requires this to be at least 30 minutes, but practical experience may vary. Setting this variable to a numeric 'value' greater than 0 causes a 'NOOP' command to be sent each 'value' seconds if no other operation is performed.

imap-list-depth
When retrieving the list of folders on an IMAP server, the **`folders`**[205] command stops after it has reached a certain depth to avoid possible infinite loops. The value of this variable sets the maximum depth allowed. The default is 2. If the folder separator on the current IMAP server is a slash '/', this variable has no effect and the **`folders`**[205] command does not descend to subfolders.

imap-use-starttls-USER@HOST, *imap-use-starttls-HOST*, *imap-use-starttls*
Causes S-nail to issue a 'STARTTLS' command to make an unencrypted IMAP session TLS encrypted. This functionality is not supported by all servers, and is not used if the session is already encrypted by the IMAPS method. Directly using encrypted communication channels should be preferred.

SEE ALSO

`bogofilter`(1)[795], `gpg`(1)[796], `more`(1)[797], `newaliases`(1)[798], `openssl`(1)[799], `sendmail`(1)[800], `sh`(1)[801], `spamassassin`(1)[802], `iconv`(3)[803], `setlocale`(3)[804], `aliases`(5)[805], `termcap`(5)[806], `terminfo`(5)[807], `locale`(7)[808], `mailaddr`(7)[809],

`re_format`(7)[810] (or `regex`(7)[811]), `mailwrapper`(8)[812], `sendmail`(8)[813]

HISTORY

M. Douglas McIlroy writes in his article "A Research UNIX Reader: Annotated Excerpts from the Programmer's Manual, 1971-1986" that a `mail`(1)[814] command already appeared in First Edition UNIX in 1971:

> Electronic mail was there from the start. Never satisfied with its exact behavior, everybody touched it at one time or another: to assure the safety of simultaneous access, to improve privacy, to survive crashes, to exploit uucp, to screen out foreign freeloaders, or whatever. Not until v7 did the interface change (Thompson). Later, as mail became global in its reach, Dave Presotto took charge and brought order to communications with a grab-bag of external networks (v8).

BSD Mail, in large parts compatible with UNIX mail, was written in 1978 by Kurt Shoens and developed as part of the BSD UNIX distribution until 1995. This manual page is derived from "The Mail Reference Manual" that Kurt Shoens wrote for Mail 1.3, included in 3BSD in 1980. The common UNIX and BSD denominator became standardized as `mailx`(1)[815] in the X/Open Portability Guide Issue 2 (January 1987). After the rise of Open Source BSD variants Mail saw continuous development in the individual code forks, noticeably by Christos Zoulas in NetBSD. Based upon this Nail, later Heirloom Mailx, was developed by Gunnar Ritter in the years 2000 until 2008. Since 2012 S-nail is maintained by Steffen Nurpmeso.

Electronic mail exchange in general is a concept even older. The earliest well documented electronic mail system was part of the Compatible Time Sharing System (CTSS) at MIT, its MAIL command had been proposed in a staff planning memo at the end of 1964 and was implemented in mid-1965 when Tom Van Vleck and Noel Morris wrote the necessary code. Similar communication programs were built for other timesharing systems. One of the most ambitious and influential was Murray Turoff's EMISARI. Created in 1971 for the United States Office of Emergency Preparedness, EMISARI combined private electronic messages with a chat system, public postings, voting, and a user directory.

During the 1960s it was common to connect a large number of terminals to a single, central computer. Connecting two computers together was relatively unusual. This began to change with the development of the ARPANET, the ancestor of today's Internet. In 1971 Ray Tomlinson adapted the SNDMSG program, originally developed for the University of California at Berkeley timesharing system, to give it the ability to transmit a message across the network into the mailbox of a user on a different computer. For the first time it was necessary to specify the recipient's computer as well as an account name. Tomlinson decided that the underused commercial at '`@`' would work to separate the two.

Sending a message across the network was originally treated as a special instance of transmitting a file, and so a MAIL command was included in RFC 385 on file transfer in 1972. Because it was not always clear when or where a message had come from, RFC 561 in 1973 aimed to formalize electronic mail headers, including "from", "date", and "subject". In 1975 RFC 680 described fields to help with the transmission of messages to multiple users, including "to", "cc", and "bcc". In 1977 these features and others went from best practices to a binding standard in RFC 733. Queen Elizabeth II of England became the first head of state to send electronic mail on March 26 1976 while ceremonially opening a building in the British Royal Signals and Radar Establishment (RSRE) in Malvern.

AUTHORS

Kurt Shoens, Edward Wang, Keith Bostic, Christos Zoulas, Gunnar Ritter. S-nail is developed by Steffen Nurpmeso <s-mailx@lists.sdaoden.eu>.

CAVEATS

[v15 behaviour may differ] Interrupting an operation via `SIGINT` aka `control-C` from anywhere else but a command prompt is very problematic and likely to leave the program in an undefined state: many library functions cannot deal with the **`siglongjmp`**(*3*) that this software (still) performs; even though efforts have

been taken to address this, no sooner but in v15 it will have been worked out: interruptions have not been disabled in order to allow forceful breakage of hanging network connections, for example (all this is unrelated to *ignore[451]*).

The SMTP and POP3 protocol support of S-nail is very basic. Also, if it fails to contact its upstream SMTP server, it will not make further attempts to transfer the message at a later time (setting *save[546]* and *sendwait[552]* may be useful). If this is a concern, it might be better to set up a local SMTP server that is capable of message queuing.

BUGS

When a network-based mailbox is open, directly changing to another network-based mailbox of a different protocol (i.e., from POP3 to IMAP or vice versa) will cause a “deadlock”.

After deleting some message of a POP3 mailbox the header summary falsely claims that there are no messages to display, one needs to perform a scroll or dot movement to restore proper state.

Please report bugs to the *contact-mail[402]* address, for example from within s-nail: ‘`? eval[127] mail[224] $contact-mail`’. Including the*verbose [616]* output of the command **version[305]** may be helpful:

```
? set escape=! verbose; vput version xy; unset verbose;\
  eval mail $contact-mail
Bug subject
!I xy
!.
```

Information on the web at `$ s-nail -X 'echo $`*contact-web[403]*`; x'`.