Emacs SMTP Library

An Emacs package for sending mail via SMTP

Simon Josefsson, Alex Schroeder

Table of Contents

1 How Mail Works

On the internet, mail is sent from mail host to mail host using the simple mail transfer protocol (SMTP). To send and receive mail, you must get it from and send it to a mail host. Every mail host runs a mail transfer agent (MTA) such as Exim that accepts mails and passes them on. The communication between a mail host and other clients does not necessarily involve SMTP, however. Here is short overview of what is involved.

The mail program — also called a mail user agent (MUA) — usually sends outgoing mail to a mail host. When your computer is permanently connected to the internet, it might even be a mail host itself. In this case, the MUA will pipe mail to the '`/usr/lib/sendmail`' application. It will take care of your mail and pass it on to the next mail host.

When you are only connected to the internet from time to time, your internet service provider (ISP) has probably told you which mail host to use. You must configure your MUA to use that mail host. Since you are reading this manual, you probably want to configure Emacs to use SMTP to send mail to that mail host. More on that in the next section.

Things are different when reading mail. The mail host responsible for your mail keeps it in a file somewhere. The messages get into the file by way of a mail delivery agent (MDA) such as procmail. These delivery agents often allow you to filter and munge your mails before you get to see it. When your computer is that mail host, this file is called a spool, and sometimes located in the directory '`/var/spool/mail/`'. All your MUA has to do is read mail from the spool, then.

When your computer is not always connected to the internet, you must get the mail from the remote mail host using a protocol such as POP3 or IMAP. POP3 essentially downloads all your mail from the mail host to your computer. The mail is stored in some file on your computer, and again, all your MUA has to do is read mail from the spool.

When you read mail from various machines, downloading mail from the mail host to your current machine is not convenient. In that case, you will probably want to use the IMAP protocol. Your mail is kept on the mail host, and you can read it while you are connected via IMAP to the mail host.

So how does reading mail via the web work, you ask. In that case, the web interface just allows you to remote-control a MUA on the web host. Whether the web host is also a mail host, and how all the pieces interact is completely irrelevant. You usually cannot use Emacs to read mail via the web, unless you use software that parses the ever-changing HTML of the web interface.

2 Emacs Speaks SMTP

Emacs includes a package for sending your mail to a SMTP server and have it take care of delivering it to the final destination, rather than letting the MTA on your local system take care of it. This can be useful if you don't have a MTA set up on your host, or if your machine is often disconnected from the internet.

Sending mail via SMTP requires configuring your mail user agent (see ⟨undefined⟩ [Mail Methods], page ⟨undefined⟩) to use the SMTP library. How to do this should be described for each mail user agent; for the default mail user agent the variable `send-mail-function` (see ⟨undefined⟩ [Mail Sending], page ⟨undefined⟩) is used; for the Message and Gnus user agents the variable `message-send-mail-function` (see ⟨undefined⟩ [Mail Variables], page ⟨undefined⟩) is used.

```
;; If you use the default mail user agent.
(setq send-mail-function 'smtpmail-send-it)
;; If you use Message or Gnus.
(setq message-send-mail-function 'smtpmail-send-it)
```

Before using SMTP you must find out the hostname of the SMTP server to use. Your system administrator should provide you with this information, but often it is the same as the server you receive mail from.

`smtpmail-smtp-server`
: The variable `smtpmail-smtp-server` controls the hostname of the server to use. It is a string with an IP address or hostname. It defaults to the contents of the `SMTPSERVER` environment variable, or, if empty, the contents of `smtpmail-default-smtp-server`.

`smtpmail-default-smtp-server`
: The variable `smtpmail-default-smtp-server` controls the default hostname of the server to use. It is a string with an IP address or hostname. It must be set before the SMTP library is loaded. It has no effect if set after the SMTP library has been loaded, or if `smtpmail-smtp-server` is defined. It is usually set by system administrators in a site wide initialization file.

The following example illustrates what you could put in '`~/.emacs`' to set the SMTP server name.

```
;; Send mail using SMTP via mail.example.org.
(setq smtpmail-smtp-server "mail.example.org")
```

SMTP is normally used on the registered "smtp" TCP service port 25. Some environments use SMTP in "Mail Submission" mode, which uses port 587. Using other ports is not uncommon, either for security by obscurity purposes, port forwarding, or otherwise.

`smtpmail-smtp-service`
: The variable `smtpmail-smtp-service` controls the port on the server to contact. It is either a string, in which case it will be translated into an integer using system calls, or an integer.

The following example illustrates what you could put in '`~/.emacs`' to set the SMTP service port.

```
;; Send mail using SMTP on the mail submission port 587.
(setq smtpmail-smtp-service 587)
```

3 Authentication

Many environments require SMTP clients to authenticate themselves before they are allowed to route mail via a server. The two following variables contains the authentication information needed for this. The first variable, `smtpmail-auth-credentials`, instructs the SMTP library to use a SASL authentication step, currently only the CRAM-MD5 and LOGIN mechanisms are supported and will be selected in that order if the server support both.

The second variable, `smtpmail-starttls-credentials`, instructs the SMTP library to connect to the server using STARTTLS. This means the protocol exchange may be integrity protected and confidential by using TLS, and optionally also authentication of the client. This feature uses the elisp package '`starttls.el`' (see it for more information on customization), which in turn require that at least one of the following external tools are installed:

1. The GNUTLS command line tool '`gnutls-cli`', you can get it from `http://www.gnu.org/software/gnutls/`. This is the recommended tool, mainly because it can verify the server certificates.
2. The '`starttls`' external program, you can get it from '`starttls-*.tar.gz`' from `ftp://ftp.opaopa.org/pub/elisp/`.

It is not uncommon to use both these mechanisms, e.g., to use STARTTLS to achieve integrity and confidentiality and then use SASL for client authentication.

`smtpmail-auth-credentials`

The variable `smtpmail-auth-credentials` contains a list of hostname, port, username and password tuples. When the SMTP library connects to a host on a certain port, this variable is searched to find a matching entry for that hostname and port. If an entry is found, the authentication process is invoked and the credentials are used.

The hostname field follows the same format as `smtpmail-smtp-server` (i.e., a string) and the port field the same format as `smtpmail-smtp-service` (i.e., a string or an integer). The username and password fields, which either can be `nil` to indicate that the user is prompted for the value interactively, should be strings with the username and password, respectively, information that is normally provided by system administrators.

`smtpmail-starttls-credentials`

The variable `smtpmail-starttls-credentials` contains a list of tuples with hostname, port, name of file containing client key, and name of file containing client certificate. The processing is similar to the previous variable. The client key and certificate may be `nil` if you do not wish to use client authentication.

The following example illustrates what you could put in '`~/.emacs`' to enable both SASL authentication and STARTTLS. The server name (`smtpmail-smtp-server`) is *hostname*, the server port (`smtpmail-smtp-service`) is *port*, and the username and password are *username* and *password* respectively.

```
;; Authenticate using this username and password against my server.
(setq smtpmail-auth-credentials
```

```
      '(("hostname" "port" "username" "password")))

;; Note that if port is an integer, you must not quote it as a
;; string.  Normally port should be the integer 25, and the example
;; become:
(setq smtpmail-auth-credentials
      '(("hostname" 25 "username" "password")))

;; Use STARTTLS without authentication against the server.
(setq smtpmail-starttls-credentials
      '(("hostname" "port" nil nil)))
```

4 Queued delivery

If you connect to the internet via a dialup connection, or for some other reason don't have permanent internet connection, sending mail will fail when you are not connected. The SMTP library implements queued delivery, and the following variable control its behaviour.

`smtpmail-queue-mail`
: The variable `smtpmail-queue-mail` controls whether a simple off line mail sender is active. This variable is a boolean, and defaults to `nil` (disabled). If this is non-`nil`, mail is not sent immediately but rather queued in the directory `smtpmail-queue-dir` and can be later sent manually by invoking `smtpmail-send-queued-mail` (typically when you connect to the internet).

`smtpmail-queue-dir`
: The variable `smtpmail-queue-dir` specifies the name of the directory to hold queued messages. It defaults to '`~/Mail/queued-mail/`'.

The function `smtpmail-send-queued-mail` can be used to send any queued mail when `smtpmail-queue-mail` is enabled. It is typically invoked interactively with *`M-x smtpmail-send-queued-mail RET`* when you are connected to the internet.

5 Server workarounds

Some SMTP servers have special requirements. The following variables implement support for common requirements.

`smtpmail-local-domain`
: The variable `smtpmail-local-domain` controls the hostname sent in the first `EHLO` or `HELO` command sent to the server. It should only be set if the `system-name` function returns a name that isn't accepted by the server. Do not set this variable unless your server complains.

`smtpmail-sendto-domain`
: The variable `smtpmail-sendto-domain` makes the SMTP library add '`@`' and the specified value to recipients specified in the message when they are sent using the `RCPT TO` command. Some configurations of sendmail requires this behaviour. Don't bother to set this unless you have get an error like:

```
Sending failed; SMTP protocol error
```

when sending mail, and the debug buffer (see Chapter 6 [Debugging], page 7) contains an error such as:

```
RCPT TO: someone
501 someone: recipient address must contain a domain
```

6 Debugging

Sometimes delivery fails, often with the generic error message '`Sending failed; SMTP protocol error`'. Enabling one or both of the following variables and inspecting a trace buffer will often give clues to the reason for the error.

`smtpmail-debug-info`
: The variable `smtpmail-debug-info` controls whether to print the SMTP protocol exchange in the minibuffer, and retain the entire exchange in a buffer '`*trace of SMTP session to` *server*`*`', where *server* is the name of the mail server to which you send mail.

`smtpmail-debug-verb`
: The variable `smtpmail-debug-verb` controls whether to send the `VERB` token to the server. The `VERB` server instructs the server to be more verbose, and often also to attempt final delivery while your SMTP session is still running. It is usually only useful together with `smtpmail-debug-info`. Note that this may cause mail delivery to take considerable time if the final destination cannot accept mail.

7 Index

7.1 Concept Index

7.2 Function and Variable Index

Table of Contents