

msmtp

Version 1.8.12, 21 August 2020

Martin Lambers (marlam@marlam.de)

This manual was last updated 21 August 2020 for version 1.8.12 of msmtmp.

Copyright (C) 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020 Martin Lambers

Copyright (C) 2011 Scott Shumate

Copying and distribution of this file, with or without modification, are permitted in any medium without royalty provided the copyright notice and this notice are preserved. These files are offered as-is, without any warranty.

Table of Contents

1	Introduction	1
2	Configuration files	2
2.1	General commands	2
2.2	Authentication commands	3
2.3	TLS commands	4
2.4	Commands specific to sendmail mode	5
3	Invocation	8
3.1	Synopsis	8
3.2	Options	8
3.2.1	General options	8
3.2.2	Changing the mode of operation	8
3.2.3	Configuration options	9
3.2.4	Options specific to sendmail mode	10
3.3	Choosing an account	12
3.4	Exit code	12
3.5	Files	12
3.6	Environment	12
4	Transport Layer Security	14
4.1	Client Certificates	15
5	Authentication	16
6	Delivery Status Notifications	18
7	Sendmail mode	19
7.1	Envelope-from address	19
7.2	Logging	19
7.3	Header handling	19
8	Server information mode	21
9	Remote Message Queue Starting mode	22

10	Examples	23
10.1	A user configuration file	23
10.2	A system wide configuration file	24
10.3	Using msmtplib with Mutt	25
10.4	Using msmtplib with mail	26
10.5	Using msmtplib with Tor	26
10.6	Aliases file	26
11	Minimal SMTP server (msmtplib)	27

1 Introduction

msmtp is an SMTP client.

In its default mode of operation, it reads a mail from standard input and sends it to a predefined SMTP server that takes care of proper delivery. Command line options and exit codes are compatible to sendmail.

The best way to start is probably to have a look at the Examples section. See Chapter 10 [Examples], page 23.

In addition to sendmail mode, there are two other modes of operation:

- Server information mode. In this mode, msmtp prints as much information as it can get about a given SMTP server (supported features, maximum mail size, . . .).
- Remote Message Queue Starting mode. In this mode, msmtp sends a Remote Message Queue Starting request for a host, domain, or queue to a given SMTP server.

Normally, a system wide configuration file and/or a user configuration file contain information about which SMTP server to use and how to use it, but all settings can also be configured on the command line.

The information about SMTP servers is organized in accounts. Each account describes one SMTP server: host name, authentication settings, TLS settings, and so on. Each configuration file can define multiple accounts.

Supported features include:

- TLS secured connections (including server certificate verification and the possibility to send a client certificate)
- Authentication methods PLAIN, LOGIN, CRAM-MD5 and EXTERNAL (and GSS-API, SCRAM-SHA-1, DIGEST-MD5, and NTLM when compiled with GNU SASL support)
- Internationalized Domain Names (IDN)
- DSN (Delivery Status Notification) support
- PIPELINING support for increased transmission speed
- RMQS (Remote Message Queue Starting) support (ETRN keyword)

2 Configuration files

A suggestion for a suitable configuration file can be generated using the ‘`--configure`’ option; see [–configure], page 8.

msmtp supports a system wide configuration file and a user configuration file. Both are optional and need not exist.

If it exists and is readable, a system wide configuration file `SYSCONFDIR/msmtp` will be loaded. Use `--version` to find out which `SYSCONFDIR` your version of msmtp uses.

The default user configuration file is `~/.msmtp` or `$XDG_CONFIG_HOME/msmtp/config`. Accounts defined in the user configuration file override accounts from the system configuration file. Configuration file settings can be changed by command line options.

A configuration file is a simple text file. Empty lines and comment lines (first non-blank character is ‘`#`’) are ignored. Every other line must contain a command and may contain an argument to that command. The argument may be enclosed in double quotes (“).

If a file name starts with the tilde (~), this tilde will be replaced by `$HOME`.

If a command accepts the argument ‘`on`’, it also accepts an empty argument and treats that as if it was ‘`on`’.

Commands are organized in accounts. Each account starts with the ‘`account`’ command and defines the settings for one SMTP account.

See Chapter 10 [Examples], page 23.

2.1 General commands

‘`defaults`’

Set defaults. The following commands will set default values for all following account definitions in the current configuration file.

‘`account name [: account[,...]]`’

Start a new account definition with the given name. The current default values are filled in (see [defaults], page 2).

If a colon and a list of previously defined accounts is given after the account name, the new account, with the filled in default values, will inherit all settings from the accounts in the list.

‘`host hostname`’

The SMTP server to send the mail to. The argument may be a host name or a network address. Every account definition must contain this command.

‘`port number`’

The port that the SMTP server listens on. The default is 25 (“smtp”), unless TLS without STARTTLS is used, in which case it is 465 (“smtps”).

‘`source_ip [IP]`’

Set a source IP address to bind the outgoing connection to. Useful only in special cases on multi-home systems. An empty argument disables this.

‘`proxy_host [IP|hostname]`’

Use a SOCKS proxy. All network traffic will go through this proxy host, including DNS queries, except for a DNS query that might be necessary to resolve

the proxy host name itself (this can be avoided by using an IP address as proxy host name). An empty argument disables proxy usage. The supported SOCKS protocol version is 5. If you plan to use this with Tor, see also Section 10.5 [Using msmtplib with Tor], page 26.

`'proxy_port [number]'`

Set the port number for the proxy host. An empty `'number'` argument resets this to the default port, which is 1080 ("socks").

`'socket [socketname]'`

Set the file name of a unix domain socket to connect to. This overrides both `'host'/'port'` and `'proxy_host'/'proxy_port'`.

`'timeout (off|seconds)'`

Set or unset a network timeout, in seconds. The argument `'off'` means that no timeout will be set, which means that the operating system default will be used.

`'protocol (smtp|lmtp)'`

Set the protocol to use. Currently only SMTP and LMTP are supported. SMTP is the default. See [port], page 2, for default ports.

`'domain argument'`

This command sets the argument of the SMTP EHLO (or LMTP LHLO) command. The default is `'localhost'`, which is stupid but usually works. Try to change the default if mails get rejected due to anti-SPAM measures. Possible choices are the domain part of your mail address (`provider.example` for `joe@provider.example`) or the fully qualified domain name of your host (if available).

2.2 Authentication commands

See Chapter 5 [Authentication], page 16.

`'auth [(on|off|method)]'`

Enable or disable authentication and optionally choose a method to use. The argument `'on'` chooses a method automatically. Accepted methods are `'plain'`, `'scram-sha-1'`, `'oauthbearer'`, `'cram-md5'`, `'gssapi'`, `'external'`, `'digest-md5'`, `'login'`, and `'ntlm'`. See Chapter 5 [Authentication], page 16.

`'user [username]'`

Set the user name for authentication. An empty argument unsets the user name. Authentication must be activated with the `'auth'` command.

`'password [secret]'`

Set the password for authentication. An empty argument unsets the password. Consider using the `'passwordeval'` command or a key ring instead of this command, to avoid storing cleartext passwords in the configuration file. See Chapter 5 [Authentication], page 16.

`'passwordeval [eval]'`

Set the password for authentication to the output (stdout) of the command *eval*. This can be used e.g. to decrypt password files on the fly or to query key rings, and thus to avoid storing cleartext passwords.

Note that the *eval* command must not mess with standard input (stdin) because that is where *msmtp* reads the mail from. If in doubt, append `< /dev/null` to *eval*.

See Chapter 5 [Authentication], page 16.

`'ntlm domain [ntlm domain]'`

Set a domain for the 'ntlm' authentication method. This is obsolete.

2.3 TLS commands

See Chapter 4 [Transport Layer Security], page 14.

`'tls [(on|off)]'`

Enable or disable TLS (also known as SSL) for secured connections.

`'tls_starttls [(on|off)]'`

Choose the TLS variant: start TLS from within the session ('on', default), or tunnel the session through TLS ('off').

`'tls_trust_file [file]'`

Activate server certificate verification using a list of trusted Certification Authorities (CAs). The default is the special value 'system', which selects the system default. An empty argument disables trust in CAs. If you select a file, it must be in PEM format, and you should also use 'tls_crl_file'.

`'tls_crl_file [file]'`

Set a certificate revocation list (CRL) file for TLS, to check for revoked certificates. An empty argument disables this.

`'tls_fingerprint [fingerprint]'`

Set the fingerprint of a single certificate to accept for TLS. This certificate will be trusted regardless of its contents (this overrides 'tls_trust_file'). The fingerprint should be of type SHA256, but can for backwards compatibility also be of type SHA1 or MD5 (please avoid this). The format should be 01:23:45:67:.... Use '`--serverinfo --tls --tls-certcheck=off --tls-fingerprint=`' to get the server certificate fingerprint.

`'tls_key_file [file]'`

Send a client certificate to the server (use this together with 'tls_cert_file'). The file must contain the private key of a certificate in PEM format. An empty argument disables this feature.

`'tls_cert_file [file]'`

Send a client certificate to the server (use this together with 'tls_key_file'). The file must contain a certificate in PEM format. An empty argument disables this feature.

`'tls_certcheck [(on|off)]'`

Enable or disable checks of the server certificate. They are enabled by default. Disabling them will override `'tls_trust_file'` and `'tls_fingerprint'`. WARNING: When the checks are disabled, TLS sessions will not be secure!

`'tls_min_dh_prime_bits [bits]'`

Set or unset the minimum number of Diffie-Hellman (DH) prime bits accepted for TLS sessions. The default is set by the TLS library and can be selected by using an empty argument to this command. Only lower the default (for example to 512 bits) if there is no other way to make TLS work with the remote server.

`'tls_priorities [priorities]'`

Set the priorities for TLS sessions. The default is set by the TLS library and can be selected by using an empty argument to this command. See the GnuTLS documentation of the `'gnutls_priority_init'` function for a description of the *priorities* string.

`'tls_host_override [host]'`

By default, TLS host verification uses the host name given by the `'host'` command. This command allows one to use a different host name for verification. This is only useful in special cases.

2.4 Commands specific to sendmail mode

See Chapter 7 [Sendmail mode], page 19.

`'from [address]'`

Set the envelope-from address. The following substitution patterns are supported:

- `%U` will be replaced by `$USER`, or if that fails by `$LOGNAME`, or if that fails by the login name of the user running `msmtp`.
- `%H` will be replaced by `$HOSTNAME`, or if that fails by the host name of the system.
- `%C` will be replaced by the canonical name of `%H`.
- `%M` will be replaced by the contents of `/etc/mailname` (potentially a different directory is used depending on the build configuration; see the output of `'msmtp --version'` and look for the location of the system configuration file).

Note that the obsolete `'auto_from'` command replaces this envelope-from address. See Section 7.1 [Envelope-from address], page 19.

`'dsn_notify (off|condition)'`

Set the condition(s) under which the mail system should send DSN (Delivery Status Notification) messages. The argument `'off'` disables explicit DSN requests, which means the mail system decides when to send DSN messages. This is the default. The *condition* must be `'never'`, to never request notification, or a comma separated list (no spaces!) of one or more of the following: `'failure'`, to request notification on transmission failure, `'delay'`, to be notified of message

delays, **‘success’**, to be notified of successful transmission. The SMTP server must support the DSN extension. See Chapter 6 [Delivery Status Notifications], page 18.

‘dsn_return (off|amount)’

This command controls how much of a mail should be returned in DSN (Delivery Status Notification) messages. The argument **‘off’** disables explicit DSN requests, which means the mail system decides how much of a mail it returns in DSN messages. This is the default. The *amount* must be **‘headers’**, to just return the message headers, or **‘full’**, to return the full mail. The SMTP server must support the DSN extension. See Chapter 6 [Delivery Status Notifications], page 18.

‘set_from_header [(auto|on|off)]’

When to set a From header: **‘auto’** adds a From header if the mail does not have one (this is the default), **‘on’** always sets a From header and overrides any existing one, and **‘off’** never sets a From header.

If the mail server rejects the mail because its From header does not match the envelope from address (a common anti-spam measure), then you might want to set this option to **‘on’**.

For compatibility with older versions, **‘add_missing_from_header [(on|off)]’** is still supported and corresponds to the **‘auto’** and **‘off’** settings. See Section 7.3 [Header handling], page 19.

‘set_date_header [(auto|off)]’

When to set a Date header: **‘auto’** adds a Date header if the mail does not have one (this is the default), and **‘off’** never sets a Date header.

For compatibility with older versions, **‘add_missing_date_header [(on|off)]’** is still supported and corresponds to the **‘auto’** and **‘off’** settings. See Section 7.3 [Header handling], page 19.

‘remove_bcc_headers [(on|off)]’

This command controls whether to remove Bcc headers. The default is to remove them.

‘undisclosed_recipients [(on|off)]’

When set, the original To, Cc, and Bcc headers of the mail are removed and a single new header line "To: undisclosed-recipients:;" is added. The default setting is off. See Section 7.3 [Header handling], page 19.

‘logfile [file]’

Enable logging to the specified file. An empty argument disables logging. The file name **‘-’** directs the log information to standard output. See Section 7.2 [Logging], page 19.

‘logfile_time_format [fmt]’

Set or unset the log file time format. This will be used as the format string for the `strftime()` function. An empty argument chooses the default (**‘"%b %d %H:%M:%S"'**). See Section 7.2 [Logging], page 19.

`'syslog [(on|off|facility)]'`

Enable or disable syslog logging. The *facility* can be one of `'LOG_USER'`, `'LOG_MAIL'`, `'LOG_LOCAL0'`, ..., `'LOG_LOCAL7'`. The default is `'LOG_USER'`. Syslog logging is disabled by default. See Section 7.2 [Logging], page 19.

`'aliases [file]'`

Replace local recipients with addresses in the aliases file. The aliases file is a cleartext file containing mappings between a local address and a list of replacement addresses. The mappings are of the form:

`local: someone@example.com, person@domain.example`

Multiple replacement addresses are separated with commas. Comments start with `'#'` and continue to the end of the line.

The local address `'default'` has special significance and is matched if the local address is not found in the aliases file. If no `'default'` alias is found, then the local address is left as is.

An empty argument to the aliases command disables the replacement of local addresses. This is the default.

`'auto_from [(on|off)]'`

Obsolete; you can achieve the same and more using the substitution patterns of the `'from'` command.

Enable or disable automatic envelope-from addresses. The default is `'off'`. When enabled, an envelope-from address of the form `user@domain` will be generated. The local part will be set to `USER` or, if that fails, to `LOGNAME` or, if that fails, to the login name of the current user. The domain part can be set with the `'maildomain'` command; if that is empty, the address not have a domain part. See Section 7.1 [Envelope-from address], page 19.

`'maildomain [domain]'`

Obsolete; you can achieve the same and more using the substitution patterns of the `'from'` command.

Set a domain part for the generation of an envelope-from address. See [auto_from], page 7.

3 Invocation

3.1 Synopsis

- Sendmail mode (default):
`msmtp [option...] [--] recipient...`
`msmtp [option...] -t [--] [recipient...]`
- Configuration mode:
`msmtp --configure mailaddress`
- Server information mode:
`msmtp [option...] --serverinfo`
- Remote Message Queue Starting mode:
`msmtp [option...] --rmqs=(host|@domain|#queue)`

3.2 Options

Options override configuration file settings. They are compatible with sendmail where appropriate.

3.2.1 General options

- `--version`
 Print version information, including information about the libraries used.
- `--help` Print help.
- `-p`
- `--pretend`
 Print the configuration settings that would be used, but do not take further action. An asterisk (*) will be printed instead of the password.
- `-v`
- `-d`
- `--debug` Print lots of debugging information, including the whole conversation with the server. Be careful with this option: the (potentially dangerous) output will not be sanitized, and your password may get printed in an easily decodable format!

3.2.2 Changing the mode of operation

- `--configure=mailaddress`
 Generate a configuration for the given mail address and print it. This can be modified or copied unchanged to the configuration file. Note that this only works for mail domains that publish appropriate SRV records; see RFC 8314.
- `-S`
- `--serverinfo`
 Print information about the SMTP server and exit. This includes information about supported features (mail size limit, authentication, TLS, DSN, ...) and

about the TLS certificate (if TLS is active). See Chapter 8 [Server information mode], page 21.

`--rmqs=(host|@domain|#queue)`

Send a Remote Message Queue Starting request for the given host, domain, or queue to the SMTP server and exit. See Chapter 9 [Remote Message Queue Starting mode], page 22.

3.2.3 Configuration options

Most options in this category correspond to a configuration file command. Please refer to Chapter 2 [Configuration files], page 2, for detailed information.

`-C filename`

`--file=filename`

Use the given file instead of `~/.msmtp` or `XDG_CONFIG_HOME/msmtp/config` as the user configuration file.

`-a account`

`--account=account`

Use the given account instead of the account named `'default'`. This option cannot be used together with the `--host` option. See [Choosing an account], page 12.

`--host=hostname`

Use this server with settings from the command line; do not use any configuration file data. This option cannot be used together with the `--account` option. It disables loading of configuration files. See [Choosing an account], page 12.

`--port=number`

Set the port number to connect to. See [port], page 2.

`--source-ip=[IP]`

Set or unset an IP address to bind the socket to. See [source_ip], page 2.

`--proxy-host=[IP|hostname]`

Set or unset a SOCKS proxy to use. See [proxy_host], page 2.

`--proxy-port=[number]`

Set or unset a port number for the proxy host. See [proxy_port], page 3.

`--socket=[socketname]`

Set or unset a local unix domain socket name to connect to. See [socket], page 3.

`--timeout=(off|seconds)`

Set or unset a network timeout, in seconds. See [timeout], page 3.

`--protocol=(smtp|lmtp)`

Set the protocol. See [protocol], page 3.

`--domain=[argument]`

Set the argument of the SMTP EHLO (or LMTP LHLO) command. See [domain], page 3.

- '--auth[=(on|off|*method*)]'
- Enable or disable authentication and optionally choose the method. See [auth], page 3.
- '--user=[*username*]
- Set or unset the user name for authentication. See [user], page 3.
- '--passwordeval=[*eval*]
- Evaluate password for authentication. See [passwordeval], page 3.
- '--tls[=(on|off)]'
- Enable or disable TLS/SSL. See [tls], page 4.
- '--tls-starttls[=(on|off)]'
- Enable or disable STARTTLS for TLS. See [tls-starttls], page 4.
- '--tls-trust-file=[*file*]
- Set or unset a trust file for TLS. See [tls-trust-file], page 4.
- '--tls-crl-file=[*file*]
- Set or unset a certificate revocation list (CRL) file for TLS. See [tls-crl-file], page 4.
- '--tls-fingerprint=[*fingerprint*]
- Set or unset the fingerprint of a trusted TLS certificate. See [tls-fingerprint], page 4.
- '--tls-key-file=[*file*]
- Set or unset a key file for TLS. See [tls-key-file], page 4.
- '--tls-cert-file=[*file*]
- Set or unset a cert file for TLS. See [tls-cert-file], page 4.
- '--tls-certcheck[=(on|off)]'
- Enable or disable server certificate checks for TLS. See [tls-certcheck], page 4.
- '--tls-min-dh-prime-bits=[*bits*]
- Set or unset minimum bit size of the Diffie-Hellman (DH) prime. See [tls-min-dh-prime-bits], page 5.
- '--tls-priorities=[*priorities*]
- Set or unset TLS priorities. See [tls-priorities], page 5.
- '--tls-host-override=[*host*]
- Set or unset override for TLS host verification. See [tls-host-override], page 5.

3.2.4 Options specific to sendmail mode

- '-f *address*'
- '--from=*address*'
- Set the envelope-from address. See [from], page 5.
- If no account was chosen yet (with --account or --host), this option will choose the first account that has the given envelope-from address (set with the 'from' command). If no such account is found, "default" is used. See [Choosing an account], page 12.
- See the 'from' command for substitution patterns supported in *address*.

‘-N (off|condition)’
 ‘--dsn-notify=(off|condition)’
 Set or unset DSN notification conditions. See [dsn_notify], page 5.

‘-R (off|amount)’
 ‘--dsn-return=(off|amount)’
 Set or unset the DSN notification amount. See [dsn_return], page 6. Note that ‘hdrs’ is accepted as an alias for ‘headers’ to be compatible with sendmail.

‘--set-from-header[=(auto|on|off)]’
 Set From header handling. See [set_from_header], page 6.

‘--set-date-header[=(auto|off)]’
 Set Date header handling. See [set_date_header], page 6.

‘--remove-bcc-headers[=(on|off)]’
 Enable or disable the removal of Bcc headers. See [remove_bcc_headers], page 6.

‘--undisclosed-recipients[=(on|off)]’
 Enable or disable the replacement of To/Cc/Bcc with "To: undisclosed-recipients;". See [undisclosed_recipients], page 6.

‘-X [file]’
 ‘--logfile=[file]’
 Set or unset the log file. See [logfile], page 6.

‘--logfile-time-format=[fmt]’
 Set or unset the log file time format. See [logfile_time_format], page 6.

‘--syslog[=(on|off|facility)]’
 Enable or disable syslog logging. See [syslog], page 6.

‘-t’
 ‘--read-recipients’
 Send the mail to the recipients given in the To, Cc, and Bcc headers of the mail in addition to the recipients given on the command line.
 If any Resent- headers are present, then the addresses from any Resent-To, Resent-Cc, and Resent-Bcc headers in the first block of Resent- headers are used instead.

‘--read-envelope-from’
 Read the envelope from address from the From header of the mail.

‘--aliases=[file]’
 Set or unset an aliases file. See [aliases], page 7.

‘-Fname’ Msmtmp adds a From header to mails that lack it, using the envelope from address. This option allows to set a full name to be used in that header.

‘--auto-from[=(on|off)]’
 Obsolete. See [auto_from], page 7.

‘--maildomain=[domain]’
 Obsolete. See [maildomain], page 7.

`--` This marks the end of options. All following arguments will be treated as recipient addresses, even if they start with a `'`.

The following options are accepted but ignored for sendmail compatibility: `-Btype`, `-bm`, `-G`, `-hN`, `-i`, `-L tag`, `-m`, `-n`, `-O option=value`, `-ox value`

3.3 Choosing an account

There are three ways to choose the account to use.

1. `--account=account`
Use the given account. Command line settings override configuration file settings.
2. `--host=hostname`
Use only the settings from the command line; do not use any configuration file data.
3. `--from=address` or `--read-envelope-from`
Choose the first account from the system or user configuration file that has a matching envelope-from address as specified by a [from], page 5, command. This works only when neither `--account` nor `--host` is used.
Subaddresses are supported. For example, the envelope from address `user+detail@example.com` will match the account for `user@example.com`.

If none of the above options is used (or if no account has a matching [from], page 5, command), then the account "default" is used.

3.4 Exit code

The standard exit codes from `sysexits.h` are used.

3.5 Files

`SYSCONFDIR/msmtprc`

The system configuration file. Use the `--version` option to find out what `SYSCONFDIR` is on your platform.

`~/.msmtprc` or `$XDG_CONFIG_HOME/msmtp/config.`

The default user configuration file.

`~/.netrc` and `SYSCONFDIR/netrc`

The `netrc` file contains login information. Before prompting for a password, `msmtp` will search it in `~/.netrc` and `SYSCONFDIR/netrc`.

3.6 Environment

`USER`, `LOGNAME`

These variables override the user's login name when constructing an envelope-from address. `LOGNAME` is only used if `USER` is unset.

`TMPDIR` Directory to create temporary files in. If this is unset, a system specific default directory is used. A temporary file is used to buffer the headers of the mail but not the mail body, so the file will not get very large.

‘EMAIL, SMTPSERVER’

These environment variables are used only if neither **--host** nor **--account** is used and there is no default account defined in the configuration files. In this case, the host name is taken from **SMTPSERVER**, and the envelope from address is taken from **EMAIL**, unless overridden by **--from** or **--read-envelope-from**. Currently **SMTPSERVER** must contain a plain host name (no URL), and **EMAIL** must contain a plain address (no names or additional information).

4 Transport Layer Security

Transport Layer Security (TLS) "... provides communications privacy over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery" (quote from RFC2246).

A server can use TLS in one of two modes:

- Via a STARTTLS command
The session starts with the normal protocol initialization, and TLS is then started using the protocol's STARTTLS command.
- Immediately
TLS is initialized before the normal protocol initialization. This requires a separate port.

The first mode is the default, but you can switch to the second mode by disabling [tls_starttls], page 4.

When TLS is started, the server sends a certificate to identify itself. To verify the server identity, a client program is expected to check that the certificate is formally correct and that it was issued by a Certificate Authority (CA) that the user trusts. (There can also be certificate chains with intermediate CAs.)

The list of trusted CAs is specified using the [tls_trust_file], page 4, command. The default value is `'system'` and chooses the system-wide default, but you can also choose the trusted CAs yourself.

One practical problem with this approach is that the client program should also check if the server certificate has been revoked for some reason, using a Certificate Revocation List (CRL). A CRL file can be specified using the [tls_crl_file], page 4, command, but getting the relevant CRL files and keeping them up to date is not straightforward. You are basically on your own.

A much more serious and fundamental problem is that you need to trust CAs. Like any other organization, a CA can be incompetent, malicious, subverted by bad people, or forced by government agencies to compromise end users without telling them. All of these things happened and continue to happen worldwide. The idea to have central organizations that have to be trusted for your communication to be secure is fundamentally broken.

Instead of putting trust in a CA, you can choose to trust only a single certificate for the server you want to connect to. For that purpose, specify the certificate fingerprint with [tls_fingerprint], page 4. This makes sure that no man-in-the-middle can fake the identity of the server by presenting you a fraudulent certificate issued by some CA that happens to be in your trust list. However, you have to update the fingerprint whenever the server certificate changes, and you have to make sure that the change is legitimate each time, e.g. when the old certificate expired. This is inconvenient, but it's the price to pay.

Information about a server certificate can be obtained with `'--serverinfo --tls --tls-certcheck=off'`. This includes the issuer CA of the certificate (so you can trust that CA via `'tls_trust_file'`), and the fingerprint of the certificate (so you can trust that particular certificate via `'tls_fingerprint'`). See Chapter 8 [Server information mode], page 21.

If you need to fine tune TLS parameters, have a look at the [tls_priorities], page 5, and [tls_min_dh_prime_bits], page 5, commands.

4.1 Client Certificates

TLS also allows the server to verify the identity of the client. For this purpose, the client has to present a certificate issued by a CA that the server trusts. To present that certificate, the client also needs the matching key file. You can set the certificate and key files using `[tls_cert_file]`, page 4, and `[tls_key_file]`, page 4. This mechanism can also be used to authenticate users, so that traditional user / password authentication is not necessary anymore. See the EXTERNAL mechanism in Chapter 5 [Authentication], page 16.

```
# Enable TLS
tls on
# Enable TLS client certificates
tls_cert_file /path/to/client_cert
tls_key_file /path/to/client_key
# Enable authentication via the EXTERNAL mechanism (optional; depends on server)
# The user name is empty because the server should get it from the client cert
auth external
user ""
```

You can also use client certificates stored on some external authentication device by specifying GnuTLS device URIs in `[tls_cert_file]`, page 4, and `[tls_key_file]`, page 4. You can find the correct URIs using `p11tool --list-privkeys --login` (`p11tool` is bundled with GnuTLS). If your device requires a PIN to access the data, you can specify that using one of the password mechanisms (e.g. `[passwordeval]`, page 3, `[password]`, page 3).

```
tls_cert_file pkcs11:model=PKCS%2315%20emulated;manufacturer=piv_II;serial=00000000;to
tls_key_file pkcs11:model=PKCS%2315%20emulated;manufacturer=piv_II;serial=00000000;to
passwordeval gpg2 --no-tty -q -d ~/.smart-card-pin.gpg
```

5 Authentication

Many SMTP servers require a client to authenticate before sending mail.

Usually a user name and a password are used for authentication. The user name specified in the configuration file with the [user], page 3, command. There are five different methods to specify the password:

1. Add the password to the system key ring.

Currently supported key rings are the Gnome key ring and the Mac OS X Keychain. For the Gnome key ring, use the command ‘**secret-tool**’ (part of Gnome’s libsecret) to store passwords:

```
$ secret-tool store --label=msmtp \
  host mail.freemail.example \
  service smtp \
  user joe.smith
```

On Mac OS X, use the following command:

```
security add-internet-password -s mail.freemail.example -r smtp -a joe.smith -w
```

In both examples, replace mail.freemail.example with the SMTP server name, and joe.smith with your user name.

2. Store the password in an encrypted files, and use [passwordeval], page 3, to specify a command to decrypt that file, e.g. using GnuPG. See Chapter 10 [Examples], page 23.
3. Store the password in the configuration file using the [password], page 3, command. (Usually it is not considered a good idea to store passwords in cleartext files. If you do it anyway, you must make sure that the file can only be read by yourself.)
4. Store the password in ~/.netrc. This method is probably obsolete.
5. Type the password into the terminal when it is required.

It is recommended to use method 1 or 2.

Multiple authentication methods exist. Most servers support only some of them. Historically, sophisticated methods were developed to protect passwords from being sent unencrypted to the server, but nowadays everybody needs Chapter 4 [Transport Layer Security], page 14, anyway, so the simple methods suffice since the whole session is protected. A suitable authentication method is chosen automatically, and when TLS is disabled for some reason, only methods that avoid sending cleartext passwords are considered.

The following user / password methods are supported:

- ‘PLAIN’
A simple cleartext method supported by almost all servers.
- ‘SCRAM-SHA-1’
A method that avoids cleartext passwords.
- ‘CRAM-MD5’
An obsolete method that avoids cleartext passwords, but is not considered secure anymore.
- ‘DIGEST-MD5’
An overcomplicated obsolete method that avoids cleartext passwords, but is not considered secure anymore.

- ‘LOGIN’
A non-standard cleartext method similar to (but worse than) PLAIN.
- ‘NTLM’
An obscure non-standard method that is now considered broken. It sometimes requires a special domain parameter passed via [ntlm_domain], page 4. Do not use it.

There are currently three authentication methods that are not based on user / password information and have to be chosen manually:

- ‘OAUTHBEARER’
An OAuth2 token from the mail provider is used as the password. See the documentation of your mail provider for details on how to get this token. The ‘passwordeval’ command can be used to pass the regularly changing tokens into msmtplib from a script or an environment variable.
- ‘EXTERNAL’
The authentication happens outside of the protocol, typically by sending a TLS client certificate (see [Client Certificates], page 14).
The EXTERNAL method merely confirms that this authentication succeeded; it does not perform the authentication. Thus it may not be necessary to use it for authentication to succeed, and if the server does not support the EXTERNAL method, this does not mean that it does not support authentication with TLS client certificates.
- ‘GSSAPI’
With this method, the Kerberos framework takes care of secure authentication. Only a user name is required.

It depends on the underlying authentication library and its version whether a particular method is supported or not. Use `--version` to find out which methods are supported by your version.

6 Delivery Status Notifications

In situations such as delivery failure or delay, the mail system usually generates a message for the sender of the mail, informing him about the difficulties.

Delivery Status Notification (DSN) requests, defined in RFC 3461, try to give the sender of the mail control about how and when these DSN messages are sent. The SMTP server must support the DSN extension. See Chapter 8 [Server information mode], page 21.

A first parameter controls when such messages should be generated: never, on delivery failure, on delivery delay, and/or on success. This can be set with [dsn_notify], page 5, and [-dsn_notify], page 10.

A second parameter controls how much of the original mail should be contained in a DSN message: only the headers, or the full mail. This can be set with [dsn_return], page 6, and [-dsn_return], page 11. Note that this parameter only applies to DSNs that indicate delivery failure for at least one recipient. If a DSN contains no indications of delivery failure, only the headers of the message are returned.

7 Sendmail mode

7.1 Envelope-from address

The SMTP server expects a sender mail address for each mail. This is the envelope-from address. It is independent of the From header (because it is part of the mail *envelope*, not of the mail itself), but in most cases both addresses are the same.

Envelope-from addresses are set with the [from], page 5, command, which allows automatism using some substitution patterns.

7.2 Logging

When logging is enabled, msmtplib will generate one log line for each mail it tries to send.

The line will include the following information:

- Host name of the SMTP server: **host=hostname**
- Whether TLS was used: **tls=(on|off)**
- Whether authentication was used: **auth=(on|off)**
- The user name used for authentication (only if authentication is used): **user=name**
- The envelope-from address: **from=address**
- The recipient addresses: **recipients=addr1,addr2,...**
- The size of the mail as transferred to the server, in bytes (only if the delivery succeeded): **mailsize=number**
- The SMTP status code and SMTP error message (only in case of failure and only if available): **smtpstatus=number, smtpmsg='message'**. Multiline SMTP messages will be concatenated into one line.
- The msmtplib error message (only in case of failure and only if available): **errmsg='message'**
- The msmtplib exit code (from `sysexit.h`; 'EX_OK' indicates success): **exitcode=EX_...**

If a logfile is given with the [logfile], page 6, command, this log line will be prepended with the current date and time (formatted according to [logfile_time_format], page 6) and appended to the specified file.

If syslog logging is enabled with the [syslog], page 6, command, the log line is passed to the syslog service with the specified facility.

7.3 Header handling

Msmtplib transmits mails unaltered to the SMTP server, with the following exceptions:

- The Bcc header(s) will be removed. This behavior can be changed with the [remove_bcc_headers], page 6, command,
- A From header will be added if the mail does not have one. This can be changed with the [set_from_header], page 6, command. The header will use the envelope from address and optionally a full name set with the '-F' option.
- A Date header will be added if the mail does not have one. This can be changed with the [set_date_header], page 6, command.

- When `[undisclosed_recipients]`, page 6, is set, the original To, Cc, and Bcc headers are removed and replaced with "To: undisclosed-recipients;".

8 Server information mode

In server information mode, `msmtp` prints as much information about the SMTP server as it can get and then exits.

The SMTP features that can be detected are:

- **SIZE**
The maximum message size that the SMTP server accepts.
- **PIPELINING**
Whether certain SMTP commands may be send in groups rather than one by one. This can speed up mail transmission if the recipient list is long. This feature is used automatically.
- **STARTTLS**
See Chapter 4 [Transport Layer Security], page 14.
- **AUTH**
See Chapter 5 [Authentication], page 16.
- **DSN**
See Chapter 6 [Delivery Status Notifications], page 18.
- **ETRN**
See Chapter 9 [Remote Message Queue Starting mode], page 22.

If TLS is activated for server information mode, the following information will be printed about the SMTP server's TLS certificate (if available):

- Owner information
 - Common Name
 - Organization
 - Organizational unit
 - Locality
 - State or Province
 - Country
- Issuer information
 - Common Name
 - Organization
 - Organizational unit
 - Locality
 - State or Province
 - Country
- General
 - Activation time
 - Expiration time
 - SHA256 fingerprint
 - SHA1 fingerprint (deprecated)

9 Remote Message Queue Starting mode

Remote Message Queue Starting (RMQS) is defined in RFC 1985. It is a way for a client to request that a server start the processing of its mail queues for messages that are waiting at the server for the client machine. If any messages are at the server for the client, then the server creates a new SMTP session and sends the messages at that time.

msmtp can only send the request (using the ETRN SMTP command); a mail server on the client side should then accept the connection of the remote SMTP server to receive the mail.

RMQS requests can be sent with the `[-rmqs]`, page 9, option. Destinations defined in RFC 1985 are:

- *host*
Request the messages for the given host.
- *@domain*
Request the messages for the given domain.
- *#queue*
Request the delivery of the messages in the given queue.

10 Examples

10.1 A user configuration file

```
# Example for a user configuration file ~/.msmtprc
#
# This file focusses on TLS and authentication. Features not used here include
# logging, timeouts, SOCKS proxies, TLS parameters, Delivery Status Notification
# (DSN) settings, and more.

# Set default values for all following accounts.
defaults

# Use the mail submission port 587 instead of the SMTP port 25.
port 587

# Always use TLS.
tls on

# Set a list of trusted CAs for TLS. The default is to use system settings, but
# you can select your own file.
#tls_trust_file /etc/ssl/certs/ca-certificates.crt
# If you select your own file, you should also use the tls_crl_file command to
# check for revoked certificates, but unfortunately getting revocation lists and
# keeping them up to date is not straightforward.
#tls_crl_file ~/.tls-crls

# A freemail service
account freemail

# Host name of the SMTP server
host smtp.freemail.example

# As an alternative to tls_trust_file/tls_crl_file, you can use tls_fingerprint
# to pin a single certificate. You have to update the fingerprint when the
# server certificate changes, but an attacker cannot trick you into accepting
# a fraudulent certificate. Get the fingerprint with
# $ msmtprc --serverinfo --tls --tls-certcheck=off --host=smtp.freemail.example
#tls_fingerprint 00:11:22:33:44:55:66:77:88:99:AA:BB:CC:DD:EE:FF:00:11:22:33

# Envelope-from address
from joe_smith@freemail.example

# Authentication. The password is given using one of five methods, see below.
```

```

auth on
user joe.smith

# Password method 1: Add the password to the system keyring, and let msmtplib get
# it automatically. To set the keyring password using Gnome's libsecret:
# $ secret-tool store --label=msmtplib \
#   host smtp.freemail.example \
#   service smtp \
#   user joe.smith

# Password method 2: Store the password in an encrypted file, and tell msmtplib
# which command to use to decrypt it. This is usually used with GnuPG, as in
# this example. Usually gpg-agent will ask once for the decryption password.
passwordeval gpg2 --no-tty -q -d ~/.msmtplib-password.gpg

# Password method 3: Store the password directly in this file. Usually it is not
# a good idea to store passwords in cleartext files. If you do it anyway, at
# least make sure that this file can only be read by yourself.
#password secret123

# Password method 4: Store the password in ~/.netrc. This method is probably not
# relevant anymore.

# Password method 5: Do not specify a password. Msmtplib will then prompt you for
# it. This means you need to be able to type into a terminal when msmtplib runs.

# A second mail address at the same freemail service
account freemail2 : freemail
from joey@freemail.example

# The SMTP server of your ISP
account isp
host mail.isp.example
from smithjoe@isp.example
auth on
user 12345

# Set a default account
account default : freemail

```

10.2 A system wide configuration file

```

# A system wide configuration file is optional.
# If it exists, it usually defines a default account.

```

```
# This allows msmtplib to be used like /usr/sbin/sendmail.
account default

# The SMTP smarthost
host mail.oursite.example

# Use TLS on port 465
port 465
tls on
tls_starttls off

# Construct envelope-from addresses of the form "user@oursite.example"
from %U@oursite.example

# Syslog logging with facility LOG_MAIL instead of the default LOG_USER
syslog LOG_MAIL
```

10.3 Using msmtplib with Mutt

Create a configuration file for msmtplib and add the following lines to your Mutt configuration file:

```
set sendmail="/path/to/msmtplib"
set use_from=yes
set realname="Your Name"
set from=you@example.com
set envelope_from=yes
```

The ‘envelope_from=yes’ option lets Mutt use the `-f` option of msmtplib. Therefore msmtplib chooses the first account that matches the from address you@example.com. Alternatively, you can use the `-a` option:

```
set sendmail="/path/to/msmtplib -a my_account"
```

Or set everything from the command line:

```
set sendmail="/path/to/msmtplib --host=mailhub -f me@example.com --tls"
```

See [Choosing an account], page 12.

If you have multiple mail accounts in your msmtplib configuration file and let Mutt use the `-f` option to choose one, you can easily switch accounts in Mutt with the following Mutt configuration lines:

```
macro generic "<esc>1" ":set from=you@example.com"
macro generic "<esc>2" ":set from=you@your-employer.example"
macro generic "<esc>3" ":set from=you@some-other-provider.example"
```

Now you can use `<esc>1`, `<esc>2`, and `<esc>3` to switch accounts.

The following example uses a different approach: it maps the single key `<tab>` in Compose context for switching between the various account in a handy visual way. In the same Compose context, `=` is mapped in order to show the current msmtplib account. This example was contributed by Thomas Baruchel.

```
# Define <tab> and = in order to switch or see the current msmtplib account
```

```
# Don't forget to put the right path for msmtplib binary
macro compose \Cx_ ":set sendmail"
macro compose \Cx| "\Cx_ = \"/usr/local/bin/msmtplib"
macro compose \Cx& ":macro compose \\t \\\Cx"
macro compose <tab> "\Cx0"
macro compose = "\Cx_\n"
# Put the account in the following lines (here three accounts)
# Don't forget to put the number of the account at the beginning
# of the line, and the number of the next account after the '&'
macro compose \Cx0 "\Cx|\"\\n\Cx&1\\n\Cx_\n" # default and switch to 1
macro compose \Cx1 "\Cx| -a example_account\"\\n\Cx&2\\n\Cx_\n" # switch to 2
macro compose \Cx2 "\Cx| -a gmail\"\\n\Cx&0\\n\Cx_\n" # switch to 0
# End of the accounts
```

10.4 Using msmtplib with mail

Define a default account, and put the following into `~/.mailrc`:

```
set sendmail="/path/to/msmtplib"
```

You need to define a default account, because mail does not allow extra options to the msmtplib command line.

10.5 Using msmtplib with Tor

Use the following settings:

```
proxy_host 127.0.0.1
proxy_port 9050
tls on
```

Use an IP address as proxy host name, so that msmtplib does not leak a DNS query when resolving it.

TLS is required to prevent exit hosts from reading your SMTP session. You also need [tls_trust_file], page 4, or [tls_fingerprint], page 4, to check the server identity. Do not set 'domain' to something that you do not want to reveal (do not set it at all if possible).

10.6 Aliases file

```
# Example aliases file

# Send root to Joe and Jane
root: joe_smith@example.com, jane_chang@example.com

# Send cron to Mark
cron: mark_jones@example.com

# Send everything else to admin
default: admin@domain.example
```

11 Minimal SMTP server (msmtpd)

Msmtpd is a minimal SMTP server that pipes mails to msmtp (or some other program) for delivery. It is intended to be used with system services that expect an SMTP server on the local host and cannot be configured to use the sendmail interface that msmtp provides.

Msmtpd handles local SMTP clients. It listens on 127.0.0.1 port 25 by default, but can also run without its own network sockets in inetd mode, where it handles a single SMTP session on standard input / output.

In the string that defines the command that msmtpd pipes each mail to, the first occurrence of ‘%F’ will be replaced with the envelope from address. Furthermore, all recipients of the mail will be appended as arguments. The command must not write to standard output, as that would mess up the SMTP session.

Only run msmtpd if you know you need it. Only use a local interface to listen on. Take care to run it with correct user rights and permissions (e.g. use ‘CAP_NET_BIND_SERVICE’ to bind to port 25 instead of running as root, or use systemd with inetd service capabilities). Be aware that the pipe command will be run as the same user that msmtpd runs as.

Msmtpd handles the following options:

```

'--version'
    Print version information

'--help'    Print help.

'--inetd'   Start single SMTP session on stdin/stdout

'--interface=ip'
    Listen on the given IPv6 or IPv4 address instead of 127.0.0.1

'--port=number'
    Listen on the given port number instead of 25

'--command=cmd'
    Pipe mails to cmd instead of msmtp

```

Example for managing msmtpd with ‘start-stop-daemon’:

```

# start msmtpd
start-stop-daemon --start --pidfile /var/run/msmtpd.pid --make-pidfile --chuid msmtpd
# stop msmtpd
start-stop-daemon --stop --pidfile /var/run/msmtpd.pid --remove-pidfile --quiet --sig

```