

Simple Mail Transfer Protocol

“SMTP” redirects here. For the email delivery company, see [SMTP \(company\)](#).

Simple Mail Transfer Protocol (SMTP) is an [Internet standard](#) for [electronic mail](#) (email) transmission. First defined by [RFC 821](#) in 1982, it was last updated in 2008 with the [Extended SMTP](#) additions by [RFC 5321](#) - which is the protocol in widespread use today.

SMTP by default uses [TCP port 25](#). The protocol for mail submission is the same, but uses port 587. SMTP connections secured by [SSL](#), known as [SMTPS](#), default to port 465 (nonstandard, but sometimes used for legacy reasons).

Although [electronic mail servers](#) and [other mail transfer agents](#) use SMTP to send and receive mail messages, user-level client mail applications typically use SMTP only for sending messages to a mail server for [relaying](#). For receiving messages, client applications usually use either [POP3](#) or [IMAP](#).

Although proprietary systems (such as [Microsoft Exchange](#) and [IBM Notes](#)) and [webmail](#) systems (such as [Outlook.com](#), [Gmail](#) and [Yahoo! Mail](#)) use their own non-standard protocols to access mail box accounts on their own mail servers, all use SMTP when sending or receiving email from outside their own systems.

1 History

Various forms of one-to-one [electronic messaging](#) were used in the 1960s. People communicated with one another using systems developed for specific [mainframe](#) computers. As more computers were interconnected, especially in the US Government's [ARPANET](#), standards were developed to allow users of different systems to email one another. SMTP grew out of these standards developed during the 1970s.

SMTP can trace its roots to two implementations described in 1971: the Mail Box Protocol, whose implementation has been disputed,^[1] but is discussed in [RFC 196](#) and other RFCs, and the SNDMSG program, which, according to [RFC 2235](#), [Ray Tomlinson](#) of [BBN](#) invented for [TENEX](#) computers to send mail messages across the [ARPANET](#).^{[2][3][4]} Fewer than 50 hosts were connected to the ARPANET at this time.^[5]

Further implementations include [FTP Mail](#)^[6] and [Mail Protocol](#), both from 1973.^[7] Development work contin-

ued throughout the 1970s, until the ARPANET transitioned into the modern Internet around 1980. [Jon Postel](#) then proposed a [Mail Transfer Protocol](#) in 1980 that began to remove the mail's reliance on [FTP](#).^[8] SMTP was published as [RFC 788](#) in November 1981, also by Postel.

The SMTP standard was developed around the same time as [Usenet](#), a one-to-many communication network with some similarities.

SMTP became widely used in the early 1980s. At the time, it was a complement to [Unix to Unix Copy Program](#) (UUCP) mail, which was better suited for handling email transfers between machines that were intermittently connected. SMTP, on the other hand, works best when both the sending and receiving machines are connected to the network all the time. Both use a [store and forward](#) mechanism and are examples of [push technology](#). Though [Usenet's newsgroups](#) are still propagated with UUCP between servers,^[9] UUCP as a mail transport has virtually disappeared^[10] along with the "bang paths" it used as message routing headers.^[11]

[Sendmail](#), released with [4.1cBSD](#), right after [RFC 788](#), was one of the first mail transfer agents to implement SMTP.^[12] Over time, as BSD Unix became the most popular operating system on the Internet, sendmail became the most common MTA (mail transfer agent).^[13] Some other popular SMTP server programs include [Postfix](#), [qmail](#), [Novell GroupWise](#), [Exim](#), [Novell NetMail](#), [Microsoft Exchange Server](#) and [Oracle Communications Messaging Server](#).

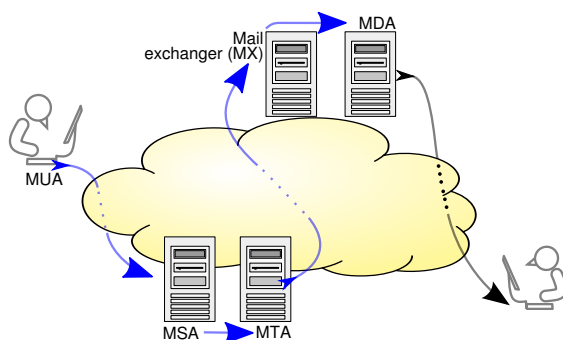
Message submission ([RFC 2476](#)) and [SMTP-AUTH](#) ([RFC 2554](#)) were introduced in 1998 and 1999, both describing new trends in email delivery. Originally, SMTP servers were typically internal to an organization, receiving mail for the organization *from the outside*, and relaying messages from the organization *to the outside*. But as time went on, SMTP servers (mail transfer agents), in practice, were expanding their roles to become [message submission agents](#) for [Mail user agents](#), some of which were now relaying mail *from the outside* of an organization. (e.g. a company executive wishes to send email while on a trip using the corporate SMTP server.) This issue, a consequence of the rapid expansion and popularity of the [World Wide Web](#), meant that SMTP had to include specific rules and methods for relaying mail and authenticating users to prevent abuses such as relaying of unsolicited email ([spam](#)). Work on message submission ([RFC 2476](#)) was originally started because popular mail servers would often rewrite mail in an attempt to fix prob-

lems in it, for example, adding a domain name to an unqualified address. This behavior is helpful when the message being fixed is an initial submission, but dangerous and harmful when the message originated elsewhere and is being relayed. Cleanly separating mail into submission and relay was seen as a way to permit and encourage rewriting submissions while prohibiting rewriting relay. As spam became more prevalent, it was also seen as a way to provide authorization for mail being sent out from an organization, as well as traceability. This separation of relay and submission quickly became a foundation for modern email security practices.

As this protocol started out purely **ASCII** text-based, it did not deal well with binary files, or characters in many non-English languages. Standards such as Multipurpose Internet Mail Extensions (**MIME**) were developed to encode binary files for transfer through SMTP. Mail transfer agents (MTAs) developed after **Sendmail** also tended to be implemented **8-bit-clean**, so that the alternate “just send eight” strategy could be used to transmit arbitrary text data (in any 8-bit ASCII-like character encoding) via SMTP. **Mojibake** was still a problem due to differing character set mappings between vendors, although the email addresses themselves still allowed only **ASCII**. 8-bit-clean MTAs today tend to support the **8BITMIME** extension, permitting binary files to be transmitted almost as easily as plain text. Recently the **SMTPUTF8** extension was created to support **UTF-8** text, allowing international content and addresses in non-Latin scripts like **Cyrillic** or **Chinese**.

Many people contributed to the core SMTP specifications, among them **Jon Postel**, **Eric Allman**, **Dave Crocker**, **Ned Freed**, **Randall Gellens**, **John Klensin**, and **Keith Moore**.

2 Mail processing model



Blue arrows can be implemented using SMTP variations.

Email is submitted by a mail client (MUA, mail user agent) to a mail server (MSA, mail submission agent) using SMTP on TCP port 587. Most mailbox providers still allow submission on traditional port 25. From there, the MSA delivers the mail to its mail transfer agent (MTA,

mail transfer agent). Often, these two agents are just different instances of the same software launched with different options on the same machine. Local processing can be done either on a single machine, or split among various appliances; in the former case, involved processes can share files; in the latter case, SMTP is used to transfer the message internally, with each host configured to use the next appliance as a **smart host**. Each process is an MTA in its own right; that is, an SMTP server.

The boundary MTA has to locate the target host. It uses the **Domain name system** (DNS) to look up the mail exchanger record (MX record) for the recipient's domain (the part of the **email address** on the right of @). The returned MX record contains the name of the target host. The MTA next connects to the exchange server as an SMTP client. (The article on **MX record** discusses many factors in determining which server the sending MTA connects to.)

Once the MX target accepts the incoming message, it hands it to a **mail delivery agent** (MDA) for local mail delivery. An MDA is able to save messages in the relevant **mailbox** format. Again, mail reception can be done using many computers or just one —the picture displays two nearby boxes in either case. An MDA may deliver messages directly to storage, or forward them over a network using SMTP, or any other means, including the **Local Mail Transfer Protocol** (LMTP), a derivative of SMTP designed for this purpose.

Once delivered to the local mail server, the mail is stored for batch retrieval by authenticated mail clients (MUAs). Mail is retrieved by end-user applications, called email clients, using **Internet Message Access Protocol** (IMAP), a protocol that both facilitates access to mail and manages stored mail, or the **Post Office Protocol** (POP) which typically uses the traditional **mbox** mail file format or a proprietary system such as Microsoft Exchange/Outlook or **Lotus Notes/Domino**. **Webmail** clients may use either method, but the retrieval protocol is often not a formal standard.

SMTP defines message *transport*, not the message *content*. Thus, it defines the mail *envelope* and its parameters, such as the **envelope sender**, but not the header (except *trace information*) nor the body of the message itself. STD 10 and RFC 5321 define SMTP (the envelope), while STD 11 and RFC 5322 define the message (header and body), formally referred to as the **Internet Message Format**.

3 Protocol overview

SMTP is a **connection-oriented**, **text-based** protocol in which a mail sender communicates with a mail receiver by issuing command strings and supplying necessary data over a reliable ordered data stream channel, typically a **Transmission Control Protocol** (TCP) connection. An

SMTP session consists of commands originated by an SMTP **client** (the initiating **agent**, sender, or transmitter) and corresponding responses from the SMTP **server** (the listening agent, or receiver) so that the session is opened, and session parameters are exchanged. A session may include zero or more SMTP transactions. An *SMTP transaction* consists of three command/reply sequences (see example below.) They are:

1. **MAIL** command, to establish the return address, a.k.a. Return-Path,^[14] reverse-path,^[15] **bounce address**, mfrom, or envelope sender. This is the address to which **bounce messages** should be sent.
2. **RCPT** command, to establish a recipient of this message. This command can be issued multiple times, one for each recipient. These addresses are also part of the envelope.
3. **DATA** to signal the beginning of the *message text*; the content of the message, as opposed to its envelope. It consists of a *message header* and a *message body* separated by an empty line. DATA is actually a group of commands, and the server replies twice: once to the *DATA command* proper, to acknowledge that it is ready to receive the text, and the second time after the end-of-data sequence, to either accept or reject the entire message.

Besides the intermediate reply for DATA, each server's reply can be either positive (2xx reply codes) or negative. Negative replies can be permanent (5xx codes) or transient (4xx codes). A **reject** is a permanent failure by an SMTP server; in this case the SMTP client should send a bounce message. A **drop** is a positive response followed by message discard rather than delivery.

The initiating host, the SMTP client, can be either an end-user's **email client**, functionally identified as a **mail user agent** (MUA), or a relay server's **mail transfer agent** (MTA), that is an SMTP server acting as an SMTP client, in the relevant session, in order to relay mail. Fully capable SMTP servers maintain queues of messages for retrying message transmissions that resulted in transient failures.

A MUA knows the *outgoing mail* SMTP server from its configuration. An SMTP server acting as client, i.e. *relaying*, typically determines which SMTP server to connect to by looking up the **MX** (Mail eXchange) **DNS** resource record for each recipient's **domain name**. Conformant MTAs (not all) fall back to a simple **A record** in case no MX record can be found. Relaying servers can also be configured to use a **smart host**.

An SMTP server acting as client initiates a **TCP** connection to the server on the "well-known port" designated for SMTP: **port 25**. MUAs should use port 587 to connect to an MSA. The main difference between an MTA and an MSA is that **SMTP Authentication** is mandatory for the latter only.

3.1 SMTP vs mail retrieval

SMTP is a delivery protocol only. In normal use, mail is "pushed" to a destination mail server (or next-hop mail server) as it arrives. Mail is routed based on the destination server, not the individual user(s) to which it is addressed. Other protocols, such as the **Post Office Protocol** (POP) and the **Internet Message Access Protocol** (IMAP) are specifically designed for use by individual users retrieving messages and managing **mail boxes**. To permit an intermittently-connected mail server to *pull* messages from a remote server on demand, SMTP has a feature to initiate mail queue processing on a remote server (see **Remote Message Queue Starting** below). POP and IMAP are unsuitable protocols for relaying mail by intermittently-connected machines; they are designed to operate after final delivery, when information critical to the correct operation of mail relay (the "mail envelope") has been removed.

3.2 Remote Message Queue Starting

Remote Message Queue Starting is a feature of SMTP that permits a remote host to start processing of the mail queue on a server so it may receive messages destined to it by sending the TURN command. This feature however was deemed insecure^[16] and was extended in **RFC 1985** with the **ETRN** command which operates more securely using an **authentication** method based on Domain Name System information.

3.3 On-Demand Mail Relay

Main article: **On-Demand Mail Relay**

On-Demand Mail Relay (ODMR) is an **SMTP extension** standardized in **RFC 2645** that allows an intermittently-connected SMTP server to receive email queued for it when it is connected.

3.4 Internationalization

Main article: **International email**

Users whose native script is not Latin based, or who use **diacritic** not in the **ASCII** character set have had difficulty with the Latin email address requirement. **RFC 6531** was created to solve that problem, providing internationalization features for SMTP, the SMTPUTF8 extension and support for multi-byte and non-ASCII characters in email addresses, such as Pelé@live.com (simple diacritic), δοκιμή@παράδειγμα.δοκιμή, and ㄖㄞˊㄝˊㄝˊㄝˊㄝˊ.

Current support is limited, but there is strong interest in broad adoption of **RFC 6531** and the related RFCs in

countries like China that have a large user base where Latin (ASCII) is a foreign script.

4 Outgoing mail SMTP server

An **email client** needs to know the IP address of its initial SMTP server and this has to be given as part of its configuration (usually given as a **DNS** name). This server will deliver outgoing messages on behalf of the user.

4.1 Outgoing mail server access restrictions

Server administrators need to impose some control on which clients can use the server. This enables them to deal with abuse, for example **spam**. Two solutions have been in common use:

- In the past, many systems imposed usage restrictions by the *location* of the client, only permitting usage by clients whose IP address is one that the server administrators control. Usage from any other client IP address is disallowed.
- Modern SMTP servers typically offer an alternative system that requires **authentication** of clients by credentials before allowing access.

4.1.1 Restricting access by location

Under this system, an **ISP's** SMTP server will not allow access by users who are outside the ISP's network. More precisely, the server may only allow access to users with an IP address provided by the ISP, which is equivalent to requiring that they are connected to the Internet using that same ISP. A mobile user may often be on a network other than that of their normal ISP, and will then find that sending email fails because the configured SMTP server choice is no longer accessible.

This system has several variations. For example, an organisation's SMTP server may only provide service to users on the same network, enforcing this by firewalling to block access by users on the wider Internet. Or the server may perform range checks on the client's IP address. These methods were typically used by corporations and institutions such as universities which provided an SMTP server for outbound mail only for use internally within the organisation. However, most of these bodies now use client authentication methods, as described below.

By restricting access to certain IP addresses, server administrators can readily recognise the IP address of any abuser. As it will be a meaningful address to them, the administrators can deal with the rogue machine or user.

Where a user is mobile, and may use different ISPs to connect to the internet, this kind of usage restriction is onerous, and altering the configured outbound email SMTP server address is impractical. It is highly desirable to be able to use email client configuration information that does not need to change.

4.1.2 Client authentication

Modern SMTP servers typically require **authentication** of clients by credentials before allowing access, rather than restricting access by location as described earlier. This more flexible system is friendly to mobile users and allows them to have a fixed choice of configured outbound SMTP server.

4.1.3 Open relay

A server that is accessible on the wider Internet and does not enforce these kinds of access restrictions is known as an **open relay**. This is now generally considered a bad practice worthy of **blacklisting**.

4.2 Ports

Communication between mail servers generally always uses the standard **TCP** port 25 designated for SMTP.

Mail *clients* however generally don't use this, instead using specific "submission" ports. Mail services generally accept email submission from clients on one of:

- 587 (Submission), as formalized in **RFC 6409** (previously **RFC 2476**)
- 465 This port has been deprecated since **RFC 2487**, after being briefly assigned for *secure SMTP* in the 1990s. Despite this, it is commonly used by mail providers^{[17][18]}

Port 2525 and others may be used by some individual providers, but have never been officially supported.

Most **Internet service providers** now block all outgoing port 25 traffic from their customers as an anti-spam measure.^{[19][20]} For the same reason, businesses will typically configure their firewall to only allow outgoing port 25 traffic from their designated mail servers.

5 SMTP transport example

A typical example of sending a message via SMTP to two mailboxes (*alice* and *theboss*) located in the same mail domain (*example.com* or *localhost.com*) is reproduced in the following session exchange. (In this example, the conversation parts are prefixed with *S:* and *C:*, for *server* and

client, respectively; these labels are not part of the exchange.)

After the message sender (SMTP client) establishes a reliable communications channel to the message receiver (SMTP server), the session is opened with a greeting by the server, usually containing its **fully qualified domain name** (FQDN), in this case *smtp.example.com*. The client initiates its dialog by responding with a HELO command identifying itself in the command's parameter with its FQDN (or an address literal if none is available).^[21]

```
S: 220 smtp.example.com ESMTP Postfix C: HELO relay.example.org
S: 250 Hello relay.example.org, I am glad to meet you
C: MAIL FROM:<bob@example.org>
S: 250 Ok C: RCPT TO:<alice@example.com>
S: 250 Ok C: RCPT TO:<theboss@example.com>
S: 250 Ok C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: From: "Bob Example" <bob@example.org>
C: To: "Alice Example" <alice@example.com>
C: Cc: theboss@example.com
C: Date: Tue, 15 January 2008 16:02:43 -0500
C: Subject: Test message
C: C: Hello Alice.
C: This is a test message with 5 header fields and 4 lines in the message body.
C: Your friend, C: Bob C: .
S: 250 Ok: queued as 12345
C: QUIT
S: 221 Bye {The server closes the connection}
```

The client notifies the receiver of the originating email address of the message in a MAIL FROM command. In this example, the email message is sent to two mailboxes on the same SMTP server: one for each recipient listed in the To and Cc header fields. The corresponding SMTP command is RCPT TO. Each successful reception and execution of a command is acknowledged by the server with a result code and response message (e.g., 250 Ok).

The transmission of the body of the mail message is initiated with a DATA command after which it is transmitted verbatim line by line and is terminated with an end-of-data sequence. This sequence consists of a new-line (<CR><LF>), a single **full stop** (period), followed by another new-line. Since a message body can contain a line with just a period as part of the text, the client sends *two* periods every time a line starts with a period; correspondingly, the server replaces every sequence of two periods at the beginning of a line with a single one. Such escaping method is called *dot-stuffing*.

The server's positive reply to the end-of-data, as exemplified, implies that the server has taken the responsibility of delivering the message. A message can be doubled if there is a communication failure at this time, e.g. due to a power shortage: Until the sender has received that 250 reply, it must assume the message was not delivered. On the other hand, after the receiver has decided to accept the message, it must assume the message has been delivered to it. Thus, during this time span, both agents have active copies of the message that they will try to deliver.^[22] The probability that a communication failure occurs exactly at this step is directly proportional to the amount of filtering that the server performs on the message body, most often

for anti-spam purposes. The limiting timeout is specified to be 10 minutes.^[23]

The QUIT command ends the session. If the email has other recipients located elsewhere, the client would QUIT and connect to an appropriate SMTP server for subsequent recipients after the current destination(s) had been queued. The information that the client sends in the HELO and MAIL FROM commands are added (not seen in example code) as additional header fields to the message by the receiving server. It adds a Received and Return-Path header field, respectively.

Some clients are implemented to close the connection after the message is accepted (250 Ok: queued as 12345), so the last two lines may actually be omitted. This causes an error on the server when trying to send the 221 reply.

6 Optional extensions

Although optional and not shown in this example, many clients ask the server for the SMTP extensions that the server supports, by using the EHLO greeting of the Extended SMTP specification (RFC 1870). Clients fall back to HELO only if the server does not respond to EHLO.

Modern clients may use the ESMTP extension keyword SIZE to query the server for the maximum message size that will be accepted. Older clients and servers may try to transfer excessively sized messages that will be rejected after consuming network resources, including connect time to network links that is paid by the minute.

Users can manually determine in advance the maximum size accepted by ESMTP servers. The client replaces the HELO command with the EHLO command.

```
S: 220 smtp2.example.com ESMTP Postfix C: EHLO bob.example.org
S: 250-smtp2.example.com Hello bob.example.org [192.0.2.201]
S: 250-SIZE 14680064
S: 250-PIPELINING
S: 250 HELP
```

Thus *smtp2.example.com* declares that it will accept a fixed maximum message size no larger than 14,680,064 **octets** (8-bit bytes). Depending on the server's actual resource usage, it may be currently unable to accept a message this large.

In the simplest case, an ESMTP server will declare a maximum SIZE immediately after receiving an EHLO. According to RFC 1870, however, the numeric parameter to the SIZE extension in the EHLO response is optional. Clients may instead, when issuing a MAIL FROM command, include a numeric estimate of the size of the message they are transferring, so that the server can refuse receipt of overly-large messages.

7 Spoofing and spamming

Main article: [Anti-spam techniques](#)

The original design of SMTP had no facility to authenticate senders, or check that servers were authorized to send on their behalf, with the result that email spoofing is possible, and commonly used in email spam and phishing.

Occasional proposals are made to modify SMTP extensively or replace it completely. One example of this is [Internet Mail 2000](#), but neither it, nor any other has made much headway in the face of the [network effect](#) of the huge installed base of classic SMTP. Instead, mail servers now use a range of techniques, including [DomainKeys](#), [DomainKeys Identified Mail](#), [Sender Policy Framework](#) and [DMARC](#), [DNSBLs](#) and [greylisting](#) to reject or quarantine suspicious emails.

8 Implementations

Main articles: [List of mail servers](#) and [Comparison of mail servers](#)

9 Related requests for comments

- [RFC 1123](#) – Requirements for Internet Hosts—Application and Support (STD 3)
- [RFC 1870](#) – SMTP Service Extension for Message Size Declaration (obsoletes: [RFC 1653](#))
- [RFC 2505](#) – Anti-Spam Recommendations for SMTP MTAs (BCP 30)
- [RFC 2920](#) – SMTP Service Extension for Command Pipelining (STD 60)
- [RFC 3030](#) – SMTP Service Extensions for Transmission of Large and Binary MIME Messages
- [RFC 3207](#) – SMTP Service Extension for Secure SMTP over Transport Layer Security (obsoletes [RFC 2487](#))
- [RFC 3461](#) – SMTP Service Extension for Delivery Status Notifications (obsoletes [RFC 1891](#))
- [RFC 3463](#) – Enhanced Status Codes for SMTP (obsoletes [RFC 1893](#), updated by [RFC 5248](#))
- [RFC 3464](#) – An Extensible Message Format for Delivery Status Notifications (obsoletes [RFC 1894](#))
- [RFC 3798](#) – Message Disposition Notification (updates [RFC 3461](#))
- [RFC 3834](#) – Recommendations for Automatic Responses to Electronic Mail
- [RFC 4952](#) – Overview and Framework for Internationalized Email (updated by [RFC 5336](#))
- [RFC 4954](#) – SMTP Service Extension for Authentication (obsoletes [RFC 2554](#), updates [RFC 3463](#), updated by [RFC 5248](#))
- [RFC 5068](#) – Email Submission Operations: Access and Accountability Requirements (BCP 134)
- [RFC 5248](#) – A Registry for SMTP Enhanced Mail System Status Codes (BCP 138) (updates [RFC 3463](#))
- [RFC 5321](#) – The Simple Mail Transfer Protocol (obsoletes [RFC 821](#) aka STD 10, [RFC 974](#), [RFC 1869](#), [RFC 2821](#), updates [RFC 1123](#))
- [RFC 5322](#) – Internet Message Format (obsoletes [RFC 822](#) aka STD 11, and [RFC 2822](#))
- [RFC 5504](#) – Downgrading Mechanism for Email Address Internationalization
- [RFC 6409](#) – Message Submission for Mail (STD 72) (obsoletes [RFC 4409](#), [RFC 2476](#))
- [RFC 6522](#) – The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages (obsoletes [RFC 3462](#), and in turn [RFC 1892](#))
- [RFC 6531](#) – SMTP Extension for Internationalized Email Addresses (updates [RFC 2821](#), [RFC 2822](#), [RFC 4952](#), and [RFC 5336](#))

10 See also

- [Bounce address](#)
- [Email encryption](#)
- [Ident](#)
- [POP before SMTP / SMTP after POP](#)
- [Sender Policy Framework \(SPF\)](#)
- [Variable envelope return path](#)
- [Dkim](#)

11 Notes

- [1] [The History of Electronic Mail](#), *Tom Van Vleck*: "It is not clear this protocol was ever implemented"
- [2] [The First Network Email](#), Ray Tomlinson, BBN

- [3] Picture of "The First Email Computer" by Dan Murphy, a PDP-10
- [4] Dan Murphy's TENEX and TOPS-20 Papers
- [5] RFC 2235
- [6] RFC 469 – Network Mail Meeting Summary
- [7] RFC 524 – A Proposed Mail Protocol
- [8] RFC 772 – Mail Transfer Protocol
- [9] Tldp.org
- [10] draft-barber-uucp-project-conclusion-05 – The Conclusion of the UUCP Mapping Project
- [11] The article about sender rewriting contains technical background info about the early SMTP history and source routing before RFC 1123.
- [12] Eric Allman (1983), *Sendmail – An Internetwork Mail Router* (PDF), BSD UNIX documentation set, Berkeley: University of California, retrieved June 29, 2012
- [13] Craig Partridge (2008), *The Technical Development of Internet Email* (PDF), IEEE Annals of the History of Computing, IEEE Computer Society, doi:10.1109/MAHC.2008.32
- [14] "The MAIL, RCPT, and DATA verbs", [D. J. Bernstein]
- [15] RFC5321 Section-7.2
- [16] RFC 1985, *SMTP Service Extension for Remote Message Queue Starting*, J. De Winter, The Internet Society (August 1996)
- [17]
- [18]
- [19]
- [20] They will generally make exceptions for businesses which have legitimate mail servers, and may allow access to their own mail servers
- [21] RFC 5321, *Simple Mail Transfer Protocol*, J. Klensin, The Internet Society (October 2008)
- [22] RFC 1047
- [23] rfc5321#section-4.5.3.2.6

12 References

- Hughes, L (1998). *Internet E-mail: Protocols, Standards and Implementation*. Artech House Publishers. ISBN 0-89006-939-5.
- Hunt, C (2003). *sendmail Cookbook*. O'Reilly Media. ISBN 0-596-00471-0.
- Johnson, K (2000). *Internet Email Protocols: A Developer's Guide*. Addison-Wesley Professional. ISBN 0-201-43288-9.

- Loshin, P (1999). *Essential Email Standards: RFCs and Protocols Made Practical*. John Wiley & Sons. ISBN 0-471-34597-0.
- Rhoton, J (1999). *Programmer's Guide to Internet Mail: SMTP, POP, IMAP, and LDAP*. Elsevier. ISBN 1-55558-212-5.
- Wood, D (1999). *Programming Internet Mail*. O'Reilly. ISBN 1-56592-479-7.

13 External links

- Email Address Internationalization IETF Working Group

14 Text and image sources, contributors, and licenses

14.1 Text

- **Simple Mail Transfer Protocol** *Source:* https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol?oldid=674525360 *Contributors:* The Anome, Arvindn, Christian List, Aldie, Gianfranco, Hirzel, Xoder, Sciplus, Nixdorf, Lousyd, Pagingmrherman, Nanshu, Yaronf, Pratyeka, Grin, Macar-enwiki, Jonik, Samuel-enwiki, Darkonc, Htaccess, Omegatron, Joy, Stormie, Mordomo, Wilinckx-enwiki, Robbot, Chealer, Fredrik, Babbage, ElBenevolente, Jleedev, Ianeiloart, Laudaka, Fredhoysted, Fleminra, Rick Block, Erdal Ronahi, Sdfisher, Ans, AlistairMcMillan, Elmindreda, Beland, Bumm13, Gscshoyru, Macquigg, Abdull, Zondor, Corti, Mike Rosoft, Seffer, JTN, Rich Farmbrough, Edgarde, Shlomif, ZeroOne, Plugwash, Nabber00, Cedders, RuslanBo, Leif, Ebradsha, Cwolvesheep, Chbarts, Wrs1864, Krellis, HasharBot-enwiki, Alansohn, Gargaj, Free Bear, Benefros, Osias, Stephan Leeds, Suruena, Tedp, Versageek, Kinema, HenryLi, Richwales, Adrian.benko, Jjatria-enwiki, Kelly Martin, Woohookitty, Mindmatrix, IRbaboon, Pol098, Elvey, Coneslayer, JVz, Jake Wartenberg, Aapo Laitinen, FlaBot, Southpaw018, Mathiastck, Intgr, Milomedes, Lukegb, Chobot, TkGy, YurikBot, Sceptre, Hairy Dude, RussBot, Hydrargyrum, Barefootguru, C777, Rsrikanth05, Yahya Abdal-Aziz, CecilWard, Tony1, Htonl, Ms2ger, Rwxrwxrwx, 21655, Lt-wiki-bot, Cedar101, Milliped, Frnkstn, Ivolutien, SmackBot, Incnis Mersi, Jhawk2k, Speight, Ohnoitsjamie, GBL, Lapsus Linguae, Anwar saadat, Amatulic, Chris the speller, AhmedHan, DStoykov, Thumperward, Snori, DelinK, LaKing, Nbarth, Omniplex, DHN-bot-enwiki, Parastical, TheKMan, Napalm Llama, WintersChild, Dave-ros, Mwtoews, LeoNomis, AnK, Zac67, Kuru, Aelfgifu, Twocs, Breno, Brownb2, NongBot-enwiki, Kvng, Rory O'Kane, Zophelia, JohnJSmith, Beno1000, Cryptic C62, CmdrObot, PorthosBot, Augurr, SJFriedl, MeekMark, Chrisahn, Equendil, Phantom87, Cydebot, UncleBubba, TAB, Vanished User jdksfajlasd, Anabolism, YiannisI, Thijs!bot, Eprb123, Hai Dang Quang-enwiki, A3RO, Uruiamme, Dawnseeker2000, Escarbot, Mentifisto, AntiVandalBot, TGDA, Julie Deanna, Deadbeef, MER-C, Enjoi4586, Magioladitis, Squire55, DirkTheDaring, Faizhaider, Quadro, Usien6, Jmbox80, DerHexer, Bakulev, Riccardobot, R'n'B, Lalala2005, Kentucho, Mange01, Trusilver, Scurless, Andareed, Liagent, LordAnubisBOT, Robert Illes, Cinedictum, 97198, MKoltnow, Bigdumbdinosaur, STBotD, Bonadea, Ale2006, Funandtrvl, VolkovBot, Fences and windows, Mdmkolbe, Hqb, Cedric dlb, Dirkbb, ErikWarmelink, Archwyrn, Kbrose, SieBot, Nubiatech, Xbt, Hac13, Fstanchina, Diego Grez-Cañete, Andrij Kursetsky, StaticGull, Hariva, Nacarlson, ClueBot, HujiBot, Ldorfman, Wikievil666, The Thing That Should Not Be, Simsvc, Astrps, Excirial, Alexbot, Sjmsteffann, PixelBot, Jotterbot, Sdrtirs, Aitias, Johnuniq, DumZiBoT, Fwayne, Varantes, Pgallert, Nepenthes, Addbot, GhettoBlaster, Mabdul, Jasper Deng, Graylorde, Тиверополник, Neillucas, Lightbot, Zorrobot, Luckas-bot, Bunnyhop11, Ptbotgourou, TaBOT-zerem, Washburnmav, Maxí, Brianhf, AnomieBOT, Iexel, Hogarth45, Message From Xenu, Jim1138, Andrew.cudzilo, Blueraspberry, Xqbot, UBJ 43X, FIL123456789, Omnipaedista, Mark Schierbecker, Hymek, Shep68, Recognizance, I dream of horses, HRoestBot, Cafeduke, Nrawat, Devrishighosh, Lotje, Michael9422, Dinamik-bot, Reach Out to the Truth, EmausBot, Rolf the wolf, Mjdtjm, BabsiSnoecks, GoingBatty, EQuail, Richard asr, Shoo, MRLC94, Rajeev1486, Felixkasza, L Kensington, Michael Leeman, NTox, VictorianMutant, Kerberos976, Will Beback Auto, ClueBot NG, Pokemonblackds, Vlhsp, Bshurtz, Animedude5555, Theopolisme, Helpful Pixie Bot, Corbijr, Planetary Chaos Redux, Samitalways4u, Pratyaa Ghosh, Zhaofeng Li, Mdann52, ChrisGualtieri, ChristopherTStone, Cerabot-enwiki, TwoTwoHello, Epicgenius, Dukandricka, Tara Zieminek, Sgen233, KoriganStone, Mario Castelán Castro, Teowey and Anonymous: 375

14.2 Images

- **File:Crystal_Clear_app_browser.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/f/fe/Crystal_Clear_app_browser.png *License:* LGPL *Contributors:* All Crystal icons were posted by the author as LGPL on kde-look *Original artist:* Everaldo Coelho and YellowIcon
- **File:Internet_map_1024_-_transparent,_inverted.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/3/3f/Internet_map_1024_-_transparent%2C_inverted.png *License:* CC BY 2.5 *Contributors:* Originally from the English Wikipedia; description page is/was here. *Original artist:* The Opte Project
- **File:SMTP-transfer-model.svg** *Source:* <https://upload.wikimedia.org/wikipedia/commons/6/69/SMTP-transfer-model.svg> *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Ale2006-from-en
- **File:Symbol_book_class2.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/8/89/Symbol_book_class2.svg *License:* CC BY-SA 2.5 *Contributors:* Mad by Lokal_Profil by combining: *Original artist:* Lokal_Profil
- **File:Symbol_list_class.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/d/db/Symbol_list_class.svg *License:* Public domain *Contributors:* ? *Original artist:* ?

14.3 Content license

- Creative Commons Attribution-Share Alike 3.0