Simple Mail Transfer Protocol

From Wikipedia, the free encyclopedia

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 Extended SMTP additions by RFC 5321, which is the protocol in widespread use today.

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 retrieving messages, client applications usually use either IMAP or POP3.

SMTP communication between mail servers uses TCP port 25. Mail clients on the other hand, often submit the outgoing emails to a mail server on port 587. Despite being deprecated, mail providers sometimes still permit the use of nonstandard port 465 for this purpose.

SMTP connections secured by TLS, known as SMTPS, can be made using STARTTLS.[1]

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.

Contents

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,[2] 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.[3][4][5] Fewer than 50 hosts were connected to the ARPANET at this time.[6]

Further implementations include FTP Mail[7] and Mail Protocol, both from 1973.[8] Development work continued 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.[9] 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,[10] UUCP as a mail transport has virtually disappeared[11] along with the "bang paths" it used as message routing headers.[12]

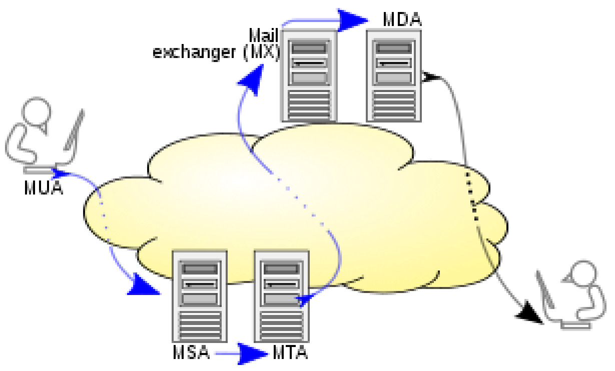
Sendmail, released with 4.1cBSD, right after RFC 788, was one of the first mail transfer agents to implement SMTP.[13] Over time, as BSD Unix became the most popular operating system on the Internet, sendmail became the most common MTA (mail transfer agent).[14] Some other popular SMTP server programs 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 problems 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.

Mail processing model



Blue arrows depict implementation of SMTP variations.

Email is submitted by a mail client (mail user agent, MUA) to a mail server (mail submission agent, MSA) using SMTP on TCP port 587. Most mailbox providers still allow submission on traditional port 25. The MSA delivers the mail to its mail transfer agent (mail transfer agent, MTA). Often, these two agents are 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 multiple machines; mail agent processes on one machine can share files, but if processing is on multiple machines, they transfer messages between each other using SMTP, where each machine is configured to use the next machine as a smart host. Each process is an MTA (an SMTP server) in its own right.

The boundary MTA 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 MX record contains the name of the target host. Based on the target host and other factors, the MTA selects an exchange server: see the article MX record. The MTA connects to the exchange server as an SMTP client.

Message transfer can occur in a single connection between two MTAs, or in a series of hops through intermediary systems. A receiving SMTP server may be the ultimate destination, an intermediate "relay" (that is, it stores and forwards the message) or a "gateway" (that is, it may forward the message using some protocol other than SMTP). Each hop is a formal handoff of responsibility for the message, whereby the receiving server must either deliver the message or properly report the failure to do so.[15]

Once the final hop accepts the incoming message, it hands it to a mail delivery agent (MDA) for local delivery. An MDA saves messages in the relevant mailbox format. As with sending, this reception can be done using one or multiple computers, but in the diagram above the MDA is depicted as one box near the mail exchanger box. An MDA may deliver messages directly to storage, or forward them over a network using SMTP or other protocol such as 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.

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:

1. **MAIL** command, to establish the return address, also called return-path,[16] reverse-path,[17] bounce address, mfrom, or envelope sender.
2. **RCPT** command, to establish a recipient of the 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* itself, 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 and the client should send a bounce message to the server it received it from. 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. A relay server typically determines which server to connect to by looking up the MX (Mail eXchange) DNS resource record for each recipient's domain name. If no MX record is found, a conformant relaying server (not all are) instead looks up the A record. Relay servers can also be configured to use a smart host. A relay server initiates a TCP connection to the server on the "well-known port" for SMTP: port 25, or for connecting to an MSA, port 587. The main difference between an MTA and an MSA is that connecting to an MSA requires SMTP Authentication.

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.

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[18] and was extended in RFC 1985 with the `ETRN` command which operates more securely using an authentication method based on Domain Name System information.

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.

Internationalization

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 those with diacritics and other language characters such as Greek and Chinese. [19]

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.

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.

Outgoing mail server access r estrictions

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.

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.

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.

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. SMTP Authentication, often abbreviated SMTP AUTH, is an extension of the SMTP in order to log in using an authentication mechanism.

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.

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[20][21]

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.[22] For the same reason, businesses will typically configure their firewall to only allow outgoing port 25 traffic from their designated mail servers.

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).[23]

```
S: 220 smtp.example.com ESMTP Postfix
C: HELO relay.example.com
S: 250 smtp.example.com, I am glad to meet you
C: MAIL FROM:<bob@example.com>
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.com>
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. This is also the return or bounce address in case the message cannot be delivered. 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.[24] 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.[25]

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.

Optional extensions

Clients learn what options a server supports, by using the `EHLO` greeting, as exemplified below, instead of the original `HELO` (example above). Clients fall back to `HELO` only if the server does not support SMTP extensions.[26]

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.[27]

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.com
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.

Spoofing and spamming

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.

Implementations

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)

See also

- Bounce address
- Email encryption
- Ident
- POP before SMTP / SMTP after POP

- Sender Policy Framework (SPF)
- Variable envelope return path
- DKIM

Notes

1. "SMTP Service Extension for Secure SMTP over Transport Layer Security" (https://tools.ietf.org/html/rfc3207).
2. The History of Electronic Mail (http://www.multicians.org/thvv/mail-history.html), *Tom Van Vleck*: "It is not clear this protocol was ever implemented"
3. *The First Network Email* (https://openmap.bbn.com/~tomlinso/ray/firstemailframe.html), Ray Tomlinson, BBN
4. Picture of "The First Email Computer (https://openmap.bbn.com/~tomlinso/ray/ka10.html)" by Dan Murphy, a PDP-10
5. Dan Murphy's TENEX and TOPS-20 Papers (http://www.opost.com/dlm/tenex/) Archived (https://web.archive.org/web/20071118204016/http://www.opost.com/dlm/tenex/) November 18, 2007, at the Wayback Machine.
6. RFC 2235 (https://tools.ietf.org/html/rfc2235)
7. RFC 469 (https://tools.ietf.org/html/rfc469) – Network Mail Meeting Summary
8. RFC 524 (https://tools.ietf.org/html/rfc524) – A Proposed Mail Protocol
9. RFC 772 (https://tools.ietf.org/html/rfc772) – Mail Transfer Protocol
10. Tldp.org (http://tldp.org/HOWTO/Usenet-News-HOWTO/x64.html)
11. draft-barber-uucp-project-conclusion-05 – The Conclusion of the UUCP Mapping Project (https://tools.ietf.org/html/draft-barber-uucp-project-conclusion-05)
12. The article about sender rewriting contains technical background info about the early SMTP history and source routing before RFC 1123 (https://tools.ietf.org/html/rfc1123).
13. Eric Allman (1983), *Sendmail – An Internetwork Mail Router* (http://docs.freebsd.org/44doc/smm/09.sendmail/paper.pdf) (PDF), BSD UNIX documentation set, Berkeley: University of California, retrieved June 29, 2012
14. Craig Partridge (2008), *The Technical Development of Internet Email* (http://www.ir.bbn.com/~craig/email.pdf) (PDF), IEEE Annals of the History of Computing, IEEE Computer Society, doi:10.1109/MAHC.2008.32 (https://doi.org/10.1109%2FMAHC.2008.32)
15. John Klensin (October 2008). "Basic Structure" (https://tools.ietf.org/html/rfc5321#section-2.1). *Simple Mail Transfer Protocol* (https://tools.ietf.org/html/rfc5321). IETF. sec. 2.1. RFC 5321. https://tools.ietf.org/html/rfc5321#section-2.1. Retrieved 16 January 2016.
16. "The MAIL, RCPT, and DATA verbs" (http://cr.yp.to/smtp/mail.html), [D. J. Bernstein]
17. RFC 5321 (https://tools.ietf.org/html/rfc5321) Section-7.2
18. RFC 1985 (https://tools.ietf.org/html/rfc1985), *SMTP Service Extension for Remote Message Queue Starting*, J. De Winter, The Internet Society (August 1996)
19. Jiankang Yao (19 December 2014). "Chinese email address" (http://www.ietf.org/mail-archive/web/ima/current/msg05395.html). *EAI* (Mailing list). IETF. Retrieved 24 May 2016.
20. "POP server settings for Yahoo Mail" (https://help.yahoo.com/kb/SLN4724.html). *Yahoo!*. Retrieved 18 January 2016. "Outgoing Mail (SMTP) Server: Server - smtp.mail.yahoo.com; Port - 465 or 587."
21. "Problems sending mail with POP or IMAP" (https://support.google.com/mail/answer/78775?hl=en). *Google*. Retrieved 18 January 2016. "If you tried configuring your SMTP server on port 465."
22. Cara Garretson (2005). "ISPs Pitch In to Stop Spam" (http://www.pcworld.com/article/116843/article.html). PC World. Retrieved 18 January 2016. "Last month, the Anti-Spam Technical Alliance, formed last year by Yahoo, America Online, EarthLink, and Microsoft, issued a list of antispam recommendations that includes filtering Port 25."
23. RFC 5321 (https://tools.ietf.org/html/rfc5321), *Simple Mail Transfer Protocol*, J. Klensin, The Internet Society (October 2008)
24. RFC 1047 (https://tools.ietf.org/html/rfc1047)
25. rfc5321#section-4.5.3.2.6 (https://tools.ietf.org/html/rfc5321#section-4.5.3.2.6)
26. John Klensin; Ned Freed; Marshall T. Rose; Einar A. Stefferud; Dave Crocker (November 1995). *SMTP Service Extensions* (https://tools.ietf.org/html/rfc1869). IETF. RFC 1869. https://tools.ietf.org/html/rfc1869.

27. "MAIL Parameters" (http://www.iana.org/assignments/mail-parameters/mail-parameters.txt). IANA. Retrieved 3 April 2016.

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.