EMAILCORE
Internet-Draft
Intended status: Standards Track
Expires: 13 May 2025

J.C. Klensin, Ed.

K. Murchison, Ed.
Fastmail

9 November 2024

Applicability Statement for IETF Core Email Protocols
draft-ietf-emailcore-as-13

Abstract

Electronic mail is one of the oldest Internet applications that is still in very active use. While the basic protocols and formats for mail transport and message formats have evolved slowly over the years, events and thinking in more recent years have supplemented those core protocols with additional features and suggestions for their use. This Applicability Statement describes the relationship among many of those protocols and provides guidance and makes recommendations for the use of features of the core protocols.

Open Issues

* #92 - CNAME handling in "5.1. Locating the Target Host" (https://github.com/ietf-wg-emailcore/emailcore/issues/92): Per IETF 120, Klensin to propose text.

* #93 - "7.3. VRFY, EXPN, and Security" should point to SMTP AUTH RFC (https://github.com/ietf-wg-emailcore/emailcore/issues/93): Per IETF 120, Alexey to propose text or close issue.

Status of This Memo

Table of Contents

1. Introduction

This document is an Applicability Statement [RFC2026], Section 3.2 that provides guidance in the use of the Internet's core email specifications, the Simple Mail Transfer Protocol (SMTP) [I-D.ietf-emailcore-rfc5321bis] and the Internet Message Format (IMF) [I-D.ietf-emailcore-rfc5322bis], and some extensions that have been built on them. In order to promote interoperability amongst senders, receivers, and intermediaries, it includes discussions and recommendations about selected features of SMTP, IMF, and certain extensions to them that are required, recommended, or to be avoided except in special cases. Furthermore, this document discusses some common mechanisms for confidentiality and authentication in electronic mail.

1.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Applicability of Some SMTP Provisions

Over the years since [RFC5321] was published in October 2008, usage of SMTP has evolved, machines and network speeds have increased, and the frequency with which SMTP senders and receivers have to be prepared to deal with systems that are disconnected from the Internet for long periods or that require many hops to reach has decreased. During the same period, the IETF has become much more sensitive to privacy and security issues and the need to be more resistant or robust against spam and other attacks. In addition SMTP (and Message Format) extensions have been introduced that are expected to evolve the Internet's mail system to better accommodate environments in which Basic Latin Script is not the norm.

This section describes adjustments that may be appropriate for SMTP under various circumstances and discusses the applicability of other protocols that represent newer work or that are intended to deal with relatively newer issues.

2.1. Handling of the Domain Argument to the EHLO Command

If the Domain argument to the EHLO command does not have an address record in the DNS that matches the IP address of the client, the SMTP server may refuse any mail from the client as part of established anti-abuse practice. Operational experience has demonstrated that the lack of a matching address record for the the domain name argument is at best an indication of a poorly-configured MTA, and at worst that of an abusive host.

2.2. Use of Address Literals

The address-literal ABNF non-terminal is used in various places in [I-D.ietf-emailcore-rfc5321bis] grammar however, for SMTP connections over the public internet, an address-literal as the argument to EHLO command or the Domain part of the Mailbox argument to the MAIL FROM command is quite likely to result in the message being rejected as a matter of policy at many sites, since they are deemed to be signs of at best a misconfigured server, and at worst either a compromised host or a server that's intentionally configured to hide its identity.

2.3. Use of Addresses in Top-Level Domains

While addresses in top-level domains (TLDs) are syntactically valid, mail to these addresses has never worked reliably. A handful of country code TLDs have top level MX records but they have never been widely used nor well supported. In 2013 [RFC7085] found 18 TLDs with MX records, which dropped to 17 in 2021 despite many new TLDs having been added.

Mail sent to addresses with single label domains has typically expected the address to be an abbreviation to be completed by a search list, so mail to bob@sales would be completed to bob@sales.example.com. This shortcut has led to unfortunate consequnces; in one famous case, in 1991 when the .CS domain was added to the root, mail in computer science departments started to fail as mail to bob@cs was now treated as mail to Czechoslovakia. Hence, for reliable service, mail SHOULD NOT use addreses that contain single label domains.

2.4. Use of SMTP Extensions

As SMTP has evolved over the years, several extensions have become ubiquitous. As a result, the following extensions MUST be supported by SMTP senders and receivers:

* 8-bit MIME [RFC6152]

Similarly, the following extensions SHOULD be supported by SMTP senders and receivers:

* Command Pipelining [RFC2920]

* Internationalized Email ([RFC6530], [RFC6531], [RFC6532])

Delivery Status Notifications [RFC3461] requests, while recommended and useful if supported, have not been widely implemented and deployed. Mail systems that send such requests should be prepared for systems that receive them to not recognize or support them. Note that this extension for notification requests is distinct from the format of notifications defined in [RFC3464] and [RFC6533] and, the special media type defined in [RFC6522]. All of those SHOULD be supported.

Furthermore, while Enhanced Mail System Status Codes ([RFC3463], [RFC5248]) are widely supported, they are not ubiquitous. Nevertheless, they have been found to be useful to SMTP senders in determining the exact reason for a transmission failure in a machine-readable, language-independent manner, thus allowing them to present

more detailed and language-specific error messages to users. Given the usefulness of these enhanced codes, SMTP receivers are RECOMMENDED to implement the SMTP Service Extension for Returning Enhanced Error Codes [RFC2034] utilizing the codes registered in [RFC5248].

3. Applicability of Message Format Provisions

This section describes adjustments to the Internet Message Format that may be appropriate under various circumstances.

3.1. Use of Empty Quoted Strings

The quoted-string ABNF non-terminal is used in various places in [I-D.ietf-emailcore-rfc5322bis] grammar. While it allows for empty quoted string, such construct is going to cause interoperability issues when used in certain header fields. In particular, use of empty quoted strings is discouraged in "received-token" (a component of a Received header field). For example, the following email header field is non-interoperable:

```
Received: from node.example by x.y.test "" foo; 21 Nov 1997
10:01:22 -0600
```

Use of empty quoted strings is fine in "display-name". For example, the following email header field is interoperable:

```
To: "" <test@example.com>
```

3.2. Use of Received Header Fields

3.2.1. Generation

Email addresses are commonly classified as Personally Identifiable Information (PII). Improper application of the FOR clause in Received header fields can result in disclosure of PII. As such, the FOR clause MUST NOT be generated if the message copy is associated with multiple recipients from mutliple SMTP RCPT commands. Otherwise, the value of the FOR clause MUST contain the RCPT address that caused the message to be routed to the recipient of the given copy of the message.

Note however, that if a mail system generates a FOR clause when there is only a single recipient, and doesn’t generate one when there are multiple recipients, the absence of the field is an indication that there is another recipient, which may allow someone to infer that a "blind" copy is involved.

3.2.2. Consumption

Received header fields support analysis of handling and delivery problems, as well as aiding evaluation of a message with suspicious content or attributes. The fields are easily created and have no direct security or privacy protections, and the fields can contain personally sensitive information.

Therefore, the fields do not warrant automatic trust and do warrant careful consideration before disclosing to others. They should be used with care, for whatever information is deemed valuable, and especially when syntax or values occur that are not defined by the specifications [I-D.ietf-emailcore-rfc5321bis] [I-D.ietf-emailcore-rfc5322bis].

3.3. Reuse of Existing Messages

Many mail user agents (MUAs) have functions which use an existing email message as a template for editing a new message. For example, an MUA may take an existing message, allow the user to replace the originator and destinations, edit parts of the body, and send it on to the new recipients. When performing such functions, the MUA SHOULD:

* Remove all header fields unknown to the MUA

* Remove any header fields that are only pertinent to the transport of the original message, such as trace header fields (see Section 3.6.7 of [I-D.ietf-emailcore-rfc5322bis])

4. Use of Email Addresses

4.1. Case-Sensitivity, Delimiters, and Mailbox Equivalency

SMTP specifies that the local-part of an email address is case-sensitive (see Section 2.4 of [I-D.ietf-emailcore-rfc5321bis]):

> The local-part of a mailbox MUST BE treated as case sensitive. Therefore, SMTP implementations MUST take care to preserve the case of mailbox local-parts. In particular, for some hosts, the user "smith" is different from the user "Smith". However, exploiting the case sensitivity of mailbox local-parts impedes interoperability and is discouraged.

While case-sensitivity is specified as an absolute requirement, it is important to stress that most implementations do not make case distinctions in local parts (most treat "smith", "Smith", and "SMITH"

as the same), and most implementations do preserve the case that is received (from SMTP or HTTP, from address books, or from user input). Maximum interoperability will be achieved by keeping local-parts unchanged (and especially making no attempt to change their case in any way) and by assuming that local-parts that differ only in their case probably refer to the same mailbox. This is particularly important for software that validates user-input fields, where case changes are tempting, but must be avoided.

It is also important to note, as we encounter non-ASCII local-parts over time, that case changes are both character-set dependent and language dependent, and attempts to change case without having the full context necessary are likely to be wrong often enough to matter.

Additionally, final delivery systems vary in how they interpret the use of delimiters such as '+' and '.' in local-parts. Some systems make distinctions between local-parts such as "smith" and "smith+foo", or "jane.doe" and "janedoe", while others treat them as referring to the same mailboxes respectively. Since only the final delivery system can properly interpret the local-part of an address, originating and transit/relay mail systems are discouraged from making any assumptions as to address equivalency or from making any changes to local-parts containing such delimiters.

4.2. Use of non-ASCII Characters

Proper generation and transmission of email addresses containing non-ASCII characters is discussed in [RFC6530]. Section 9 of [RFC6530] says: "a downgrade mechanism that transforms the local part of an email address cannot be utilized in transit." This is actually just a special case of a principle, discussed in Section 2.3.11 of [I-D.ietf-emailcore-rfc5321bis] and elsewhere, that nothing other than the final delivery system should attempt to interpret or alter the local-part of an address. In particular, they MUST NOT:

* use web URI percent encoding (see Section 2.1 of [RFC3986]) in either the local-part or the domain-part of an address

* perform Internationalized Domain Names for Applications (IDNA) Punycode Conversion (see Section 4.4 of [RFC5891]) on the local-part of an address

since none of these encodings will produce an address that is guaranteed to be treated as equivalent to the original one.

In some cases, servers or clients may be able to use local knowledge to substitute ASCII addresses for specific non-ASCII addresses, but that is beyond the scope of this memo. See Section 8 of [RFC6530] for further discussion.

4.3. Use and Validation in HTML and Other Contexts

Email addresses are frequently used as input in HyperText Markup Language (HTML) forms but the allowed grammar of these email addresses is more restrictive than the grammar for a 'Mailbox' in Section 4.1.2 of [I-D.ietf-emailcore-rfc5321bis] (the lack of quoted strings and limited characters allowed in domains). Implementations that intend to accept email addresses in HTML forms are encouraged to consult the valid email address grammar in Section 4.10.5.1.5 of [HTML].

Additionally, the following general guidance is provided:

* Few mail systems allow leading, trailing, or consecutive unquoted dots ('.') in the local-part of email addresses even though the HTML grammar referenced above currently allows them. Consequently, implementations are discouraged from accepting such addresses.

* Some mail systems allow a trailing dot ('.') in the domain part of email addresses (as allowed by Domain Names [RFC1035]), but this is not interoperable with all systems. Consequently, implementations are encouraged to strip trailing dots from the domain part of email addresses.

5. Use of Multipurpose Internet Mail Extensions (MIME)

Although the Multipurpose Internet Mail Extensions (MIME) [RFC2045] specification and its predecessors have remained separate from the Internet Message Format (IMF) [I-D.ietf-emailcore-rfc5322bis] specification and its predecessors, MIME features such as non-textual message bodies, multi-part message bodies, and the use of character sets other than US-ASCII in message bodies and header fields have become nearly ubiquitous in contemporary email. As a result, IMF generators and parsers are expected to support MIME.

6. Confidentiality and Authentication with SMTP

SMTP is specified without embedded mechanisms for authentication or confidentiality; its traffic is therefore "in the clear". Years of operational experience have shown that such transmission exposes the message to easy compromise, including wiretapping and spoofing. To mitigate these risks, operation of SMTP has evolved over the years so that it is used with the benefit of Transport Layer Security (TLS) [RFC8446] to provide both confidentiality and authentication in the transmission of messages. This section discusses those topics and their most common uses.

It is important that the reader understand what is meant by the terms "Authentication" and "Confidentiality", and for that we will borrow directly from RFC8446.

* Authentication is the process of establishing the identity of one or more of the endpoints of a communication channel. TLS only requires authentication of the server side of the communication channel; authentication of the client side is optional.

* The term "confidentiality" describes a state where the data (i.e., the message) is transmitted in a way that it is only visible to the endpoints of a communication channel.

It is not uncommon for implementers to use the term "encryption" to mean "confidentiality", but this is not quite correct. Rather, encryption using TLS is the current method by which confidentiality is achieved with SMTP, but that does not mean that future methods might not be developed.

Note: With typical email use of TLS, authentication only is performed for the target receiving server and is not done for the sending client. That is, it serves to validate that the connection has been made to the intended server, but does not validate who initiated it.

6.1. Optional Confidentiality

The most common implementation of message confidentiality is what’s known as "opportunistic TLS", which is frequently referred to as "opportunistic encryption". With this method, a receiving server announces in its greeting that it is capable of supporting TLS encryption through the presence of the "STARTTLS" keyword. The sending client then attempts to negotiate an encrypted connection, and if successful, transmits the message in encrypted form; if negotiation fails, the client falls back to sending the message in clear text.

Opportunistic TLS is confidentiality without authentication, because no effort is made to authenticate the receiving server, and it is optional confidentiality due to the ability to fall back to transmission in the clear if a secure connection cannot be established. That said, most modern implementations of SMTP support this method, especially at the largest mailbox providers, and so the vast majority of email traffic is encrypted during its time transiting from the client to the server.

Note: While TLS provides protection while the message is in transit, there is no guarantee that the message will be stored in encrypted fashion at its destination. In fact, storage in plain text should be expected!

6.2. Required Confidentiality, with Receiving Server Authentication

Two protocols exist that move message confidentiality from optional to required (with conditions as noted below) - MTA-STS [RFC8461] and DANE for SMTP [RFC7672]. While they differ in their implementation details, receiving servers relying on either protocol are stating that they only accept mail if the transmission can be encrypted with TLS, and a failure to negotiate a secure connection MUST result in the sending client refusing to transmit the message. Support for both protocols is increasing, but is not yet mandatory.

These two protocols differ from Opportunistic TLS in that they require receiving server authentication and there is no fallback to sending in the clear if negotiation of an encrypted connection fails.

Note: Both protocols mentioned in this section rely not only on the receiving server but also the sending client supporting the protocol intended to be used. If the sending client does not implement or understand the protocol requested by the receiving server, the sending client will use Opportunistic TLS or clear-text to transmit the message.

6.3. Message-Level Authentication

Protocols exist to allow for authentication of different identities associated with an email message - SPF [RFC7208] and DKIM [RFC6376]. A third protocol, DMARC [RFC7489], relies on SPF and DKIM to allow for validation of the domain in the visible From header, and a fourth, ARC [RFC8617], provides a way for each hop to record results of authentication checks performed at that hop.

All of these are outside the scope of this document, as they are outside the scope of SMTP. They deal with validating the authorized usage of one or more domains in an email message, and not with establishing the identity of the receiving server.

6.4. SMTP Authentication

SMTP Authentication [RFC4954], which is often abbreviated as SMTP AUTH, is an extension to SMTP. While its name might suggest that it would be within scope for this section of the Applicability Statement, nothing could be further from the truth.

SMTP AUTH defines a method for a client to identify itself to a Message Submission Agent (MSA) when presenting a message for transmission, usually using ports 465 or 587 rather than the traditional port 25. The most common implementation of SMTP AUTH is for a person to present a username and password to their mailbox provider's outbound SMTP server when configuring their MUA for sending mail.

6.5. Message-Level Confidentiality

Protocols such as S/MIME [RFC8551] and OpenPGP [RFC4880] exist to allow for message confidentiality outside of the operation of SMTP. That is to say, using these protocols results in encryption of the message prior to its being submitted to the SMTP communications channel, and decryption of the message is the responsibility of the message recipient. There are numerous implementations of these protocols, too many to list here. As they operate fully independent of SMTP, they are out of scope for this document.

7. Acknowledgments

The Emailcore group arose out of discussions on the ietf-smtp group over changes and additions that should be made to the core email protocols. It was agreed upon that it was time to create a working group that would fix many potential errors and opportunities for misunderstandings within the RFCs.

Special thanks to the following for providing significant portions of text for this document: Dave Crocker, Todd Herr, Barry Leiba, John Levine, Alexey Melnikov, Pete Resnick, and E. Sam.

8. IANA Considerations

This memo includes no requests to or actions for IANA. The IANA registries associated with the protocol specifications it references are specified in their respective documents.

9. Security Considerations

Security and privacy considerations are discussed throughout this document as they pertain to the referenced specifications.

10. References

10.1. Normative References

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, DOI 10.17487/RFC2026, October 1996, <https://www.rfc-editor.org/info/rfc2026>.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <https://www.rfc-editor.org/info/rfc2045>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

10.2. Informative References

[HTML] Web Hypertext Application Technology Working Group, "HTML Living Standard", 4 October 2022, <https://html.spec.whatwg.org/>.

[I-D.ietf-emailcore-rfc5321bis] Klensin, J. C., "Simple Mail Transfer Protocol", Work in Progress, Internet-Draft, draft-ietf-emailcore-rfc5321bis-34, 7 November 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-emailcore-rfc5321bis-34>.

[I-D.ietf-emailcore-rfc5322bis] Resnick, P., "Internet Message Format", Work in Progress, Internet-Draft, draft-ietf-emailcore-rfc5322bis-12, 13 June 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-emailcore-rfc5322bis-12>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC2034] Freed, N., "SMTP Service Extension for Returning Enhanced Error Codes", RFC 2034, DOI 10.17487/RFC2034, October 1996, <https://www.rfc-editor.org/info/rfc2034>.

[RFC2920] Freed, N., "SMTP Service Extension for Command Pipelining", STD 60, RFC 2920, DOI 10.17487/RFC2920, September 2000, <https://www.rfc-editor.org/info/rfc2920>.

[RFC3461] Moore, K., "Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notifications (DSNs)", RFC 3461, DOI 10.17487/RFC3461, January 2003, <https://www.rfc-editor.org/info/rfc3461>.

[RFC3463] Vaudreuil, G., "Enhanced Mail System Status Codes", RFC 3463, DOI 10.17487/RFC3463, January 2003, <https://www.rfc-editor.org/info/rfc3463>.

[RFC3464] Moore, K. and G. Vaudreuil, "An Extensible Message Format for Delivery Status Notifications", RFC 3464, DOI 10.17487/RFC3464, January 2003, <https://www.rfc-editor.org/info/rfc3464>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <https://www.rfc-editor.org/info/rfc4880>.

[RFC4954] Siemborski, R., Ed. and A. Melnikov, Ed., "SMTP Service Extension for Authentication", RFC 4954, DOI 10.17487/RFC4954, July 2007, <https://www.rfc-editor.org/info/rfc4954>.

[RFC5248] Hansen, T. and J. Klensin, "A Registry for SMTP Enhanced Mail System Status Codes", BCP 138, RFC 5248, DOI 10.17487/RFC5248, June 2008, <https://www.rfc-editor.org/info/rfc5248>.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <https://www.rfc-editor.org/info/rfc5321>.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <https://www.rfc-editor.org/info/rfc5891>.

[RFC6152] Klensin, J., Freed, N., Rose, M., and D. Crocker, Ed., "SMTP Service Extension for 8-bit MIME Transport", STD 71, RFC 6152, DOI 10.17487/RFC6152, March 2011, <https://www.rfc-editor.org/info/rfc6152>.

[RFC6376] Crocker, D., Ed., Hansen, T., Ed., and M. Kucherawy, Ed., "DomainKeys Identified Mail (DKIM) Signatures", STD 76, RFC 6376, DOI 10.17487/RFC6376, September 2011, <https://www.rfc-editor.org/info/rfc6376>.

[RFC6522] Kucherawy, M., Ed., "The Multipart/Report Media Type for the Reporting of Mail System Administrative Messages", STD 73, RFC 6522, DOI 10.17487/RFC6522, January 2012, <https://www.rfc-editor.org/info/rfc6522>.

[RFC6530] Klensin, J. and Y. Ko, "Overview and Framework for Internationalized Email", RFC 6530, DOI 10.17487/RFC6530, February 2012, <https://www.rfc-editor.org/info/rfc6530>.

[RFC6531] Yao, J. and W. Mao, "SMTP Extension for Internationalized Email", RFC 6531, DOI 10.17487/RFC6531, February 2012, <https://www.rfc-editor.org/info/rfc6531>.

[RFC6532] Yang, A., Steele, S., and N. Freed, "Internationalized Email Headers", RFC 6532, DOI 10.17487/RFC6532, February 2012, <https://www.rfc-editor.org/info/rfc6532>.

[RFC6533] Hansen, T., Ed., Newman, C., and A. Melnikov, "Internationalized Delivery Status and Disposition Notifications", RFC 6533, DOI 10.17487/RFC6533, February 2012, <https://www.rfc-editor.org/info/rfc6533>.

[RFC7085] Levine, J. and P. Hoffman, "Top-Level Domains That Are Already Dotless", RFC 7085, DOI 10.17487/RFC7085, December 2013, <https://www.rfc-editor.org/info/rfc7085>.

[RFC7208] Kitterman, S., "Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1", RFC 7208, DOI 10.17487/RFC7208, April 2014, <https://www.rfc-editor.org/info/rfc7208>.

[RFC7489] Kucherawy, M., Ed. and E. Zwicky, Ed., "Domain-based Message Authentication, Reporting, and Conformance (DMARC)", RFC 7489, DOI 10.17487/RFC7489, March 2015, <https://www.rfc-editor.org/info/rfc7489>.

[RFC7672] Dukhovni, V. and W. Hardaker, "SMTP Security via Opportunistic DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS)", RFC 7672, DOI 10.17487/RFC7672, October 2015, <https://www.rfc-editor.org/info/rfc7672>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8461] Margolis, D., Risher, M., Ramakrishnan, B., Brotman, A., and J. Jones, "SMTP MTA Strict Transport Security (MTA-STS)", RFC 8461, DOI 10.17487/RFC8461, September 2018, <https://www.rfc-editor.org/info/rfc8461>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/info/rfc8551>.

[RFC8617] Andersen, K., Long, B., Ed., Blank, S., Ed., and M. Kucherawy, Ed., "The Authenticated Received Chain (ARC) Protocol", RFC 8617, DOI 10.17487/RFC8617, July 2019, <https://www.rfc-editor.org/info/rfc8617>.

Appendix A. Change Log

RFC Editor: Please remove this appendix before publication.

A.1. Changes from draft-klensin-email-core-as-00 (2020-03-30) to draft-ietf-emailcore-as-00

* Change of filename, metadata, and date to reflect transition to WG document for new emailcore WG. No other substantive changes

A.2. Changes from draft-ietf-emailcore-as-00 (2020-10-06) to -01

* Added co-authors (list is in alphabetical order for the present).
* Updated references to 5321bis and 5322bis.
* Added note at top, "This version is provided as a document management convenience to update the author list and make an un-expired version available to the WG. There are no substantive changes from the prior version", which should be removed for version -02.

A.3. Changes from draft-ietf-emailcore-as-01 (2021-04-09) to -02

* Added new editors and also added some issues the emailcore group will be dealing with.
* Added reference to RFC 6648.

A.4. Changes from draft-ietf-emailcore-as-02 (2021-08-06) to -03

* Moved discussion of address-literals (issue #1) and domain names in EHLO (issue #19) under SMTP Provisions section
* Moved discussion of empty quoted-strings under Message Format Provisions section
* Added text on use of addresses in TLDs (issue #50)
* Marked all authors as editors.
* Miscellaneous editorial changes.

A.5. Changes from draft-ietf-emailcore-as-03 (2022-01-31) to -04

* Added requirements for SMTP extensions (issue #40).

A.6. Changes from draft-ietf-emailcore-as-04 (2022-05-21) to -05

* Added text addressing use ofx enhanced status codes.
* Added text addressing confidentiality and authentication (issue #54).

A.7. Changes from draft-ietf-emailcore-as-05 (2022-10-24) to -06

* Converted source to xml2rfc v3.

* Replaced placeholder Introduction with new text.

* Updated keywords boilerplate.

* Added text on interoperability of email addresses in general and use in HTML forms (issue #51).

* Added text stating that implementations are expected to support MIME (issue #65).

* Added placeholders for issues #38 and #55.

* Add list of contributors in Acknowledgments.

* Added minimal Security Considerations section.

A.8. Changes from draft-ietf-emailcore-as-06 (2022-11-07) to -07

* Added text addressing use of FOR clause in Received header fields (issue #55).

* Miscellaneous editorial changes.

A.9. Changes from draft-ietf-emailcore-as-07 (2023-03-13) to -08

* Added text addressing use of Received header fields by MUAs (issue #85).

* Added advice against use of Percent-Encoding non-ASCII characters in email addresses (issue #78).

* Miscellaneous editorial changes.

A.10. Changes from draft-ietf-emailcore-as-08 (2023-12-18) to -09

* Acknowledge the existence of port 465 for submission (issue #80).

* Remove "Use of Time Zones in Date and Received Header Fields" placeholder (issue #66).

* Miscellaneous editorial changes.

A.11. Changes from draft-ietf-emailcore-as-09 (2024-07-02) to -10

* Added Open Issues Section

* Removed placeholder for issue #38 - Clarify 78 octet limit versus 998 line length limit (https://github.com/ietf-wg-emailcore/emailcore/issues/38)

* Applied "final" proposed text for issue #78 - Advice against using URL %-encoding on non-ASCII email addresses (https://github.com/ietf-wg-emailcore/emailcore/issues/78)

* Applied proposed text for issue #84 - Handling of Trace Header Fields by MUAs (https://github.com/ietf-wg-emailcore/emailcore/issues/84)

A.12. Changes from draft-ietf-emailcore-as-10 (2024-07-03) to -11

* Added Open Issue #94 - Use of Quoted Strings (https://github.com/ietf-wg-emailcore/emailcore/issues/94)

A.13. Changes from draft-ietf-emailcore-as-11 (2024-10-21) to -12

* Applied new proposed text to Section 3.1

* Applied new proposed text for issue #40 - Recommended SMTP Extensions (https://github.com/ietf-wg-emailcore/emailcore/issues/40)

* Applied new proposed text for issue #78 - Advice against using URL %-encoding on non-ASCII email addresses (https://github.com/ietf-wg-emailcore/emailcore/issues/78)

* Applied new proposed text for issue #84 - Handling of Trace Header Fields by MUAs (https://github.com/ietf-wg-emailcore/emailcore/issues/84)

* Applied new proposed text for issue #85 - What mail agents should do/not do with Received header fields (https://github.com/ietf-wg-emailcore/emailcore/issues/85)

A.14. Changes from draft-ietf-emailcore-as-12 (2024-11-09) to -13

* Fixed discussion of Punycode (domain-part -> local-part) in Section 4.2

* Removed Keywords from discussion in Section 3.1

* Added example of empty display-name in Section 3.1

Authors' Addresses

John C Klensin (editor)
1770 Massachusetts Ave, Ste 322
Cambridge, MA 02140
United States of America
Phone: +1 617 245 1457
Email: john-ietf@jck.com

Kenneth Murchison (editor)
Fastmail US LLC
1429 Walnut Street - Suite 1201
Philadelphia, PA 19102
United States of America
Email: murch@fastmailteam.com

EMAILCORE
Internet-Draft
Obsoletes: 5321, 1846, 7504, 7505 (if approved)
Intended status: Standards Track
Expires: 19 July 2025

J. Klensin
15 January 2025

Simple Mail Transfer Protocol
draft-ietf-emailcore-rfc5321bis-39

Abstract

This document is a specification of the basic protocol for Internet electronic mail transport. It (including text carried forward from RFC 5321) consolidates, updates, and clarifies several previous documents, making all or parts of most of them obsolete. It covers the SMTP extension mechanisms and best practices for the contemporary Internet, but does not provide details about particular extensions. The document also provides information about use of SMTP for other than strict mail transport and delivery. This document replaces RFC 5321, the earlier version with the same title, and supersedes RFCs 1846, 7504, and 7505, incorporating all the relevant information in them.

Notes on Reading This Draft

This note is to be removed before publishing as an RFC.

Version -39 of this draft is posted 2025-01-15 for the convenience of the WG in preparation for the online interim meeting scheduled for 2025-01-17. While all earlier comments other than those addressed to IANA or the RFC Editor have been removed, some of the remaining one have been edited and a few new ones have been added to facilitate WG review. The draft contains additional changes tentatively made in response to comments from the IESG review and an IANA follow-up to the 2024-12-17 review. Despite several requests, neither of those IANA reviews has made it into the datatracker. A more detailed summary of changes to this I-D since the last posted draft is in Appendix F.10.

Early versions of drafts for this document, going back to the first versions considered by the EMAILCORE WG, were extensively annotated with information, primarily about changes made over the decade since RFC 5321 appeared, especially when those changes might be controversial or should get careful review. Those notes and annotations appeared as Appendices G, H, and a detailed change log as part of Appendix I in versions of this Internet-Draft through draft-ietf-emailcore-rfc5321bis-28, dated 2024-05-14. They, and the former

Appendix C, have been removed from this version in order to make the document considerably shorter and less complex for IETF Last Call. Those who are interested in or question the history of changes between the Draft Standard RFC 5321 [55] that are not explained in Section 1.2 or Appendix E will probably be able to save themselves and the community some time by consulting those appendices and annotations in <draft-ietf-emailcore-rfc5321bis-28>.

This evolving draft was developed and discussed on the emailcore@ietf.org list.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. Transport of Electronic Mail

The objective of the Simple Mail Transfer Protocol (SMTP) is to transfer mail reliably and efficiently.

SMTP is independent of the particular transmission subsystem and requires only a reliable ordered data stream channel. While this document specifically discusses transport over TCP, other transports are possible. Appendices to RFC 821 [6] describe some of them.

An important feature of SMTP is its capability to transport mail across multiple networks, usually referred to as "SMTP mail relaying" (see Section 3.6). A network consists of the mutually-TCP-accessible hosts on the public Internet, the mutually-TCP-accessible hosts on a firewall-isolated TCP/IP Intranet [37] or hosts in some other LAN or WAN environment utilizing a non-TCP transport-level protocol. Using SMTP, a process can transfer mail to another process on the same network or to some other network via a relay or gateway process accessible to both networks.

In this way, a mail message may pass through a number of intermediate relay or gateway hosts on its path from sender to ultimate recipient. The Mail eXchanger mechanisms of the domain name system (RFC 1035 [7], RFC 974 [20], and Section 5 of this document) are used to identify the appropriate next-hop destination for a message being transported.

1.2. History and Context for This Document

This Internet Standard specification contains material, in many cases including copied exact text, from several documents including some dating back to RFC 821 [6], published over forty years ago. While most of the early features are unchanged, others have been updated or enhanced. This section summarizes the relationship of the present specification to earlier ones leading up to the very similar RFC 5321 [55] of October 2008. Changes between RFC 5321 and <<This Document>> appear in Appendix E.

This document provides the specification of the basic protocol for Internet electronic mail transport. It consolidates, updates and clarifies, but does not add new or change existing functionality of the following:

* the original SMTP (Simple Mail Transfer Protocol) specification of RFC 821 [6],

* domain name system requirements and implications for mail transport from RFC 1035 [7] and RFC 974 [20],

* the clarifications and applicability statements in RFC 1123 [10],

* the new error codes added by RFC 1846 [25] and later by RFC 7504 [53], obsoleting both of those documents, and

* material drawn from the SMTP Extension mechanisms in RFC 1869 [27].

It also includes editorial, clarification, and correction changes that were made to RFC 2821 [35] to bring that specification to Draft Standard and similar changes to RFC 5321 [55] to bring the current document to Internet Standard as well as changes to the description and specification of IANA registries to align with contemporary practice and thinking.

It may help the reader to understand that, to reduce the risk of introducing errors, large parts of the document essentially merge the earlier specifications listed in the bullet points above rather than providing a completely rewritten, reorganized, and integrated description of SMTP. That strategy, and the consequent document organization, had IETF consensus at the time RFC 2821 was written. An index and additional cross-references are provided to assist in the quest for information.

This document obsoletes RFCs 5321 [55] (the earlier version of this specification), 1846 [25] and incorporates the substance of 7504 [53] (specification of reply codes), and 7505 [54] (the "Null MX" specification). Although SMTP was designed as a mail transport and delivery protocol, this specification also contains information that is relevant to its optional use for submission of mail by users and to some aspects of the Post Office Protocol (POP) (RFC 937 [18], RFC 1939 [28]) and IMAP (RFC 9051 [42]) protocols. In general, the separate mail submission protocol specified in RFC 6409 [49] is now preferred to direct use of SMTP for that function; more discussion of that subject appears in that document.

Section 2.3 provides definitions of terms specific to this document. Except when the historical terminology is necessary for clarity, this document uses the current 'client' and 'server' terminology to identify the sending and receiving SMTP processes, respectively. In general, "sender-SMTP" and "SMTP client" are equivalent as are "receiver-SMTP" and "SMTP server".

1.3. Related Documents

A companion document, rfc5322bis [16], discusses message header sections and bodies and specifies formats and structures for them.

Partially because of its origins as discussed earlier in this section, this document is narrowly focused on the SMTP protocol and does not discuss the many extensions to SMTP (an IANA registry provides the current list of such extensions [58]) or associated protocols expected to be used with it. In particular, SMTP was designed to transmit messages in clear text --specifically without consideration of the contents of messages other that the specific mail headers it adds (see Section 4.4) even though part of all of the message body may be encrypted before the message enters the SMTP system -- with the consequence that, in general, protection of messages from surveillance in transit requires extensions or supplemental protocols. Particularly important extensions and supplemental protocols, including ones intended to address those security issues, are discussed in a forthcoming companion document, referred to as the email Applicability Statement [56]. That document also provides additional discussion of security considerations surrounding the use of SMTP as well as discussion of other relevant documents.

1.4. Document Conventions

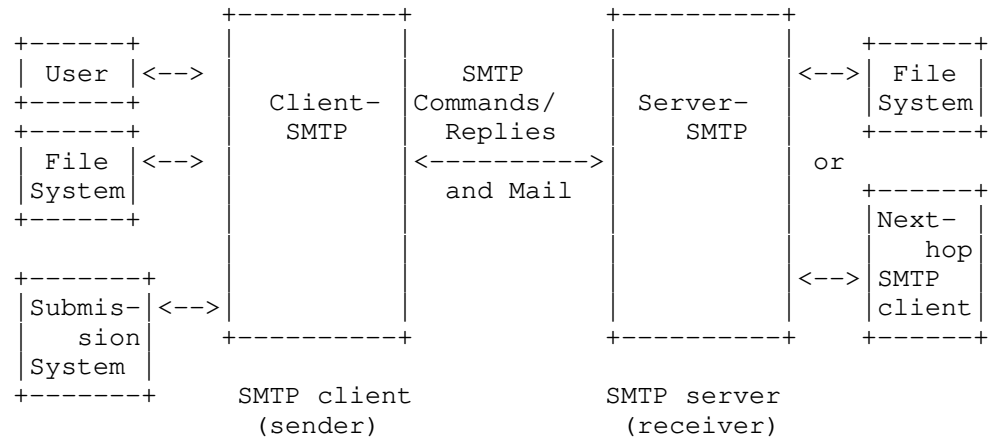
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 RFC 2119 [1] and RFC 8174 [2] when, and only when, they appear in all capitals, as shown here. As each of these terms was intentionally and carefully chosen to improve the interoperability of email, each use of these terms is to be treated as a conformance requirement.

This document has a long history. To avoid the risk of various errors and of confusing readers and documents that point to this one, most examples and the domain names they contain are preserved from RFC 2821. Readers are cautioned that these are illustrative examples that should not actually be used in either code or configuration files. In particular, some of those examples use variations on "foo", a convention explained in RFC 3092 [39].

2. The SMTP Model

2.1. Basic Structure

The SMTP design can be pictured as:



When an SMTP client has a message to transmit, it establishes a two-way transmission channel to an SMTP server. The responsibility of an SMTP client is to transfer mail messages to one or more SMTP servers, or report its failure to do so.

The means by which a mail message is presented to an SMTP client (i.e., between the first two columns above), and how that client determines the identifier(s) ("names") of the domain(s) to which mail messages are to be transferred, are local matters. They are not addressed by this document. In some cases, the designated domain(s), or those determined by an SMTP client, will identify the final destination(s) of the mail message. In other cases, common with SMTP clients associated with implementations of the POP (RFC 937 [18], RFC 1939 [28]) or IMAP (RFC 9051 [42]) protocols, or when the SMTP client is inside an isolated transport service environment, the domain determined will identify an intermediate destination through which all mail messages are to be relayed. SMTP clients that transfer all traffic regardless of the target domains associated with the individual messages, or that do not maintain queues for retrying message transmissions that initially cannot be completed, may otherwise conform to this specification but are not considered fully-capable. Fully-capable SMTP implementations, including the relays used by these less capable ones, and their destinations, are expected to support all of the queuing, retrying, and alternate address functions discussed in this specification. In many situations and configurations, the less-capable clients discussed above SHOULD be using the message submission protocol (RFC 6409 [49]) rather than SMTP.

The means by which an SMTP client, once it has determined a target domain, determines the identity of an SMTP server to which a copy of a message is to be transferred, and then performs that transfer, are covered by this document. To effect a mail transfer to an SMTP server, an SMTP client establishes a two-way transmission channel to that SMTP server. An SMTP client determines the address of an appropriate host running an SMTP server by resolving a destination domain name to either an intermediate Mail eXchanger host or a final target host.

An SMTP server may be either the ultimate destination or an intermediate "relay" (that is, it may assume the role of an SMTP client after receiving the message) or "gateway" (that is, it may transport the message further using some protocol other than SMTP). SMTP commands are generated by the SMTP client and sent to the SMTP server. SMTP replies are sent from the SMTP server to the SMTP client in response to the commands.

In other words, message transfer can occur in a single connection between the original SMTP-sender and the final SMTP-recipient, or can occur in a series of hops through intermediary systems. In either case, once the server has issued a success response at the end of the mail data, a formal handoff of responsibility for the message occurs: the protocol requires that a server MUST accept responsibility for either delivering the message or properly reporting the failure to do so (see Sections 6.1, 6.2, and 7.8, below).

Once the transmission channel is established and initial handshaking is completed, the SMTP client normally initiates a mail transaction. Such a transaction consists of a series of commands to specify the originator and destination of the mail and transmission of the message content (including any lines in the header section or other structure) itself. When the same message is sent to multiple recipients, this protocol encourages the transmission of only one copy of the data for all recipients at the same destination (or intermediate relay) host.

The server responds to each command with a reply; replies may indicate that the command was accepted, that additional commands are expected, or that a temporary or permanent error condition exists. Commands specifying the sender or recipients may include server-permitted SMTP service extension requests, as discussed in Section 2.2. The dialog is purposely lock-step, one-at-a-time, although this can be modified by mutually agreed upon extension requests such as command pipelining (RFC 2920 [36]).

Once a given mail message has been transmitted, the client may either request that the connection be shut down or may initiate other mail transactions. In addition, an SMTP client may use a connection to an SMTP server for ancillary services such as verification of email addresses or retrieval of mailing list subscriber addresses.

As discussed above, this protocol provides mechanisms for the transmission of mail. Historically, this transmission normally occurred directly from the sending user’s host to the receiving user’s host when the two hosts are connected to the same transport service. When they are not connected to the same transport service or other circumstances dictate, transmission occurs via one or more relay SMTP servers. A very common case in the Internet today involves submission of the original message to an intermediate, "message submission" server, which is similar to a relay but has some additional properties; such servers are discussed in Section 2.3.10 and at some length in RFC 6409 [49]. An intermediate host that acts as either an SMTP relay or as a gateway into some other transmission environment is usually selected through the use of the domain name service (DNS) Mail eXchanger mechanism.

2.2. The Extension Model

2.2.1. Background

In an effort that started in 1990, approximately a decade after RFC 821 was completed, the protocol was modified with a "service extensions" model that permits the client and server to agree to utilize shared functionality beyond the original SMTP requirements [24]. The SMTP extension mechanism defines a means whereby an extended SMTP client and server may recognize each other, and the server can inform the client as to the service extensions that it supports.

Contemporary SMTP implementations MUST support the basic extension mechanisms. For instance, servers MUST support the EHLO command even if they do not implement any specific extensions and clients SHOULD preferentially utilize EHLO rather than HELO. (However, for compatibility with older conforming implementations, SMTP clients and servers MUST support the original HELO mechanisms as a fallback.) Unless the different characteristics of HELO must be identified for interoperability purposes, this document discusses only EHLO.

SMTP is widely deployed and high-quality implementations have proven to be very robust. However, the Internet community now considers some services to be important that were not anticipated when the protocol was first designed. If support for those services is to be added, it must be done in a way that permits older implementations to continue working acceptably. The extension framework consists of:

* The SMTP command EHLO, superseding the earlier HELO,

* a registry of SMTP service extensions,

* additional parameters to the SMTP MAIL and RCPT commands, and

* optional replacements for commands defined in this protocol, such as for DATA in non-ASCII transmissions (RFC 3030 [38]).

SMTP's strength comes primarily from its simplicity. Experience with many protocols has shown that protocols with few options tend towards ubiquity, whereas protocols with many options tend towards obscurity.

Each SMTP implementation, as part of deciding whether to implement and support an extension and regardless of the extension's apparent benefits, must carefully scrutinize it with respect to its implementation, deployment, and interoperability costs. In many cases, the cost of extending the SMTP service will likely outweigh the benefit.

2.2.2. Definition and Registration of Extensions

Especially for extensions intended for general use and expected to interoperate well with multiple implementations, a readily-available, stable, and adequate definition is essential for those evaluating, implementing, or configuring the extension. The information below describes important characteristics of that documentation. In order to make it accessible and to prevent naming conflicts, the IANA maintains a registry of SMTP service extensions [64] and each service extension must be recorded in that registry as specified in Section 8.1.1 and below.

Experience has shown that obtaining thoughtful review and input from the broader community produces much better results than narrower discussions, e.g., only among the designers. While it is usually best to obtain that input prior to registration and to do so formally as part of an IETF Standards Track specification, there is no requirement to do so. An alternate, simplified, registration procedure (see Section 8.1.1.1, Paragraph 4, Item 2) allows extensions to be written and registered that permit modifications after registration, perhaps even after deployment experience. Even when that simplified procedure is used, and although it is not required, it will often be useful for the submitter or IANA to notify a relevant IETF mailing list of the extension request. Registrants may also reach out to selected individuals for advice on the specification and how to best obtain additional useful input. Details of the registration process itself and two available registration models appear in Section 8.1.1 below.

For standards track registrations, the definition and related description of the extension will include, not only the keyword name and syntax for the service extension and other information required for registration, but a detailed description of the purpose of the extension, what it is expected to accomplish, how its use changes the behavior of client and server SMTP implementations that use it, and how it interacts with other relevant extensions and elements of this specification. To be of maximum use, the alternative procedure should still make that information available.

Any keyword value presented in the EHLO response MUST correspond to an SMTP service extension registered with IANA as described in Section 8.1. A conforming server MUST NOT offer keyword values that are not described in a registered extension.

SMTP Clients MUST ignore any announced extension they do not recognize and, if the announcement involves parsing or other problems that prevent reliable interpretation of the response to the EHLO command, send QUIT and terminate the SMTP session.

2.2.3. Special Issues with Extensions

Extensions that change fairly basic properties of SMTP operation are permitted. The text in other sections of this document must be understood in that context. In particular, extensions can change the minimum limits specified in Section 4.5.3, can change the ASCII character set requirement as mentioned above, or can introduce some optional modes of message handling.

In particular, if an extension implies that the delivery path normally supports special features of that extension, and an intermediate SMTP system finds a next hop that does not support the required extension, it MAY choose, based on the specific extension and circumstances, to requeue the message and try later and/or try an alternate MX host. If this strategy is employed, the timeout to fall back to an unextended format (if one is available) SHOULD be less than the normal timeout for bouncing as undeliverable (e.g., if normal timeout is three days, the requeue timeout before attempting to transmit the mail without the extension might be one day).

2.3. SMTP Terminology

2.3.1. Mail Objects

SMTP transports a mail object. A mail object contains an envelope and content.

The SMTP envelope is sent as a series of SMTP protocol units (described in Section 3). It consists of an originator address (to which error reports should be directed), one or more recipient addresses, and optional protocol extension material. Historically, variations on the reverse-path (originator) address specification command (MAIL) could be used to specify alternate delivery modes, such as immediate display; those variations have now been deprecated (see Appendix D, particularly Appendix D.6).

The SMTP content is sent in the SMTP DATA protocol unit and, except as described below, is restricted only to lines of characters from the US-ASCII repertoire [4] described in Section 2.3.8. Uses of SMTP to transmit mail messages on the public Internet generally follow a more specific model, with the content consisting of two parts: the header section and the body. If the content conforms to the message format specification (RFC5322bis [16]), the header section consists of a collection of header fields, each consisting of a header name, a colon, and data. The body, if structured, is defined according to other contemporary standards such as MIME (RFC 2045 [30]). SMTP extensions (such as "8BITMIME", RFC 6152 [52]) or the replacement for the DATA command specified in RFC 3030 [38]) may relax these

restrictions either completely or in a way that is specific to the body content. Two MIME extensions (RFC 2047 [31] and RFC 2231 [34]) define an algorithm for representing header values outside the US-ASCII repertoire, while still encoding them using that repertoire.

2.3.2. Senders and Receivers

In RFC 821, the two hosts participating in an SMTP transaction were described as the "SMTP-sender" and "SMTP-receiver". This document has been changed to reflect current industry terminology and hence refers to them as the "SMTP client" (or sometimes just "the client") and "SMTP server" (or just "the server"), respectively. Since a given host may act both as server and client in a relay situation, "receiver" and "sender" terminology is still used where needed for clarity.

2.3.3. Mail Agents and Message Stores

Additional mail system terminology became common after RFC 821 was published and, where convenient, is used in this specification. In particular, SMTP servers and clients provide a mail transport service and therefore act as "Mail Transfer Agents" (MTAs). "Mail User Agents" (MUAs or UAs) are normally thought of as the sources and targets of mail. At the source, an MUA might collect mail to be transmitted from a user and hand it off to an MTA or, more commonly in recent years, a specialized variation of an MTA called a "Submission Server" [49]. At the other end of the process, the final ("delivery") MTA would be thought of as handing the mail off to an MUA (or at least transferring responsibility to it, e.g., by depositing the message in a "message store"). However, while these terms are used with at least the appearance of great precision in other environments, the implied boundaries between MUAs and MTAs often do not accurately match common, and conforming, practices with Internet mail. Hence, the reader should be cautious about inferring the strong relationships and responsibilities that might be implied if these terms were used elsewhere.

2.3.4. Host

For the purposes of this specification, a host is a computer system attached to the Internet (or, in some cases, to a private TCP/IP network) and supporting the SMTP protocol. Hosts are known by names (see the next section); they SHOULD NOT be identified by numerical addresses, i.e., by address literals as described in Section 4.1.2.

2.3.5. Domain Names

A domain name (or often just a "domain") consists of one or more components, separated by dots if more than one appears. In the case of a top-level domain used by itself in an email address, a single string is used without any dots. This makes the requirement, described in more detail below, that only fully-qualified domain names (often referred to as "FQDN"s) appear in SMTP transactions on the public Internet, particularly important where top-level domain names are involved. These components ("labels" in the DNS terminology of RFC 1035 [7]) are restricted for purposes of SMTP as defined here to consist of a sequence of letters, digits, and hyphens drawn from the ASCII character set [4] and conforming to what RFC 1035 calls the "preferred name syntax", with the exception that leading digits in labels are permitted [8]. Domain names are used as names of hosts and, except where additionally restricted in this document, of other entities in the domain name hierarchy. For example, a domain may refer to a host alias (label of a CNAME RR) or the label of Mail eXchanger records to be used to deliver mail instead of representing a host name. See RFC 1035 and Section 5 of this specification.

The domain name, as described in this document and in RFC 1035 [7], MUST be the entire, fully-qualified domain name. Other than an address literal (see Section 4.1.3) where those are permitted, any string that is not a domain name in FQDN form is no more than a reference to be interpreted locally. Such local references for domain names MUST NOT appear in any SMTP transaction (Cf. Section 5). Mechanisms for inferring FQDNs from local references (including partial names or local aliases) are out of scope for this specification and normally the province of message submission. Due to a history of problems, SMTP servers SHOULD NOT make such inferences (Message Submission Servers [49] have somewhat more flexibility) and intermediate (relay) SMTP servers MUST NOT make them.

Unless further restricted in this document, domain names used in SMTP are names that can be resolved to MX or address (A or AAAA) RRs (see also Section 5), or to CNAME RRs that can be resolved to an MX or address RR. There are two exceptions to the rule requiring FQDNs:

* The domain name given in the EHLO command MUST be either a primary host name (a domain name that resolves to an address RR) or, if the host has no name, an address literal, as described in Section 4.1.3 and discussed further in the EHLO discussion of Section 4.1.4.

* The reserved mailbox name "postmaster" MAY be used in a RCPT command without domain qualification (see Section 4.1.1.3) and MUST be accepted if so used.

2.3.6. Buffer and State Table

SMTP sessions are stateful, with both parties carefully maintaining a common view of the current state. In this document, we model this state by a virtual "buffer" and a "state table" on the server that may be used by the client to, for example, "clear the buffer" or "reset the state table", causing the information in the buffer to be discarded and the state to be returned to some previous state.

2.3.7. Commands and Replies

SMTP commands and, unless altered by a service extension, message data, are transmitted from the sender to the receiver via the transmission channel in "lines" (defined in Section 2.3.8 below).

An SMTP reply is an acknowledgment (positive or negative) sent in "lines" from receiver to sender via the transmission channel in response to a command. The general form of a reply is a numeric completion code (indicating failure or success) usually followed by a text string. The codes are for use by programs and the text is usually intended for human users. RFC 3463 [12], specifies further structuring of the reply strings, including the use of supplemental and more specific completion codes (see also RFC 5248 [51]).

2.3.8. Lines

Lines consist of zero or more data characters terminated by the sequence ASCII character "CR" (hex value 0D) followed immediately by ASCII character "LF" (hex value 0A). This termination sequence is denoted as <CRLF> in this document. Conforming implementations MUST NOT recognize or generate any other character or character sequence as a line terminator. Limits MAY be imposed on line lengths by servers (see Section 4).

In addition, the appearance of "bare" "CR" or "LF" characters in text (i.e., either without the other) has a long history of causing problems in mail implementations and applications that use the mail system as a tool. Unless negotiated otherwise using an SMTP extension, SMTP client implementations MUST NOT transmit these characters except when they are intended as line terminators and then MUST, as indicated above, transmit them only as a <CRLF> sequence.

2.3.9. Message Content and Mail Data

The terms "message content" and "mail data" are used interchangeably in this document to describe the material transmitted after the DATA command is accepted and before the end of data indication is transmitted. Message content includes the message header section and the possibly structured message body. In the absence of extensions, both are required to be ASCII (see Section 2.3.1). The MIME specification (RFC 2045 [30]) provides the standard mechanisms for structured message bodies.

2.3.10. Originator, Delivery, Relay, and Gateway Systems

This specification makes a distinction among four types of SMTP systems, based on the role those systems play in transmitting electronic mail. An "originating" system (sometimes called an SMTP originator) introduces mail into the Internet or, more generally, into a transport service environment. A "delivery" SMTP system is one that receives mail from a transport service environment and passes it to a mail user agent or deposits it in a message store that a mail user agent is expected to subsequently access. A "relay" SMTP system (usually referred to just as a "relay") receives mail from an SMTP client and transmits it, without modification to the message data other than adding trace information (see Section 4.4), to another SMTP server for further relaying or for delivery.

A "gateway" SMTP system (usually referred to just as a "gateway") receives mail from a client system in one transport environment and transmits it to a server system in another transport environment. Differences in protocols or message semantics between the transport environments on either side of a gateway may require that the gateway system perform transformations to the message that are not permitted to SMTP relay systems. For the purposes of this specification, firewalls that rewrite addresses should be considered as gateways, even if SMTP is used on both sides of them (see RFC 2979 [37]).

2.3.11. Mailbox and Address

As used in this specification, an "address" is a character string that identifies a user to whom mail will be sent or a location into which mail will be deposited. The term "mailbox" refers to that depository. The two terms are typically used interchangeably unless the distinction between the location in which mail is placed (the mailbox) and a reference to it (the address) is important. An address normally consists of user and domain specifications. The standard mailbox naming convention is defined to be "local-part@domain"; contemporary usage permits a much broader set of applications than simple "user names". Consequently, and due to a

long history of problems when intermediate hosts have attempted to optimize transport by modifying them, the local-part MUST be interpreted and assigned semantics only by the host specified in the domain part of the address.

2.3.12. Sessions and Transactions

This document distinguishes between an "SMTP session" and a "mail transaction". An SMTP session, often called a "mail session", starts when a connection is made between client and server, ending when that connection is terminated. A "mail transaction" is started and terminated by particular commands, most often MAIL and RSET or QUIT (the latter also instructs the server to close the SMTP session). For more information and details, see Section 3.1 and Section 3.3.

2.4. General Syntax Principles and Transaction Model

SMTP commands and replies have a rigid syntax. All commands begin with a command verb. All replies begin with a three digit numeric code. In some commands and replies, arguments are required following the verb or reply code. Some commands do not accept arguments (after the verb), and some reply codes are followed, sometimes optionally, by free form text. In both cases, where text appears, it is separated from the verb or reply code by a space character. Complete definitions of commands and replies appear in Section 4.

Verbs and argument values (e.g., "TO:" or "to:" in the RCPT command and extension name keywords) are not case sensitive, with the sole exception in this specification of a mailbox local-part (SMTP Extensions may explicitly specify case-sensitive elements). That is, a command verb, an argument value other than a mailbox local-part, and free form text MAY be encoded in upper case, lower case, or any mixture of upper and lower case with no impact on its meaning. The local-part of a mailbox MUST BE treated as case sensitive. Therefore, SMTP implementations MUST take care to preserve the case of mailbox local-parts. In particular, for some hosts, the user "smith" is different from the user "Smith". However, exploiting the case sensitivity of mailbox local-parts impedes interoperability and is discouraged. Mailbox domains follow normal DNS rules and are hence not case sensitive.

A few SMTP servers, in violation of this specification (and RFC 821) may require that command verbs be encoded by clients in upper case. Implementations MAY wish to employ this encoding to accommodate those servers.

The argument clause consists of a variable-length character string ending with the end of the line, i.e., with the character sequence <CRLF>. The receiver will take no action until this sequence is received.

The syntax for each command is shown with the discussion of that command. Common elements and parameters are shown in Section 4.1.2.

Commands and replies are composed of characters from the ASCII character set [4]. When the transport service provides an 8-bit byte (octet) transmission channel, each 7-bit character is transmitted, right justified, in an octet with the high-order bit cleared to zero. More specifically, the unextended SMTP service provides 7-bit transport only. An originating SMTP client that has not successfully negotiated an appropriate extension with a particular server (see the next paragraph) MUST NOT transmit messages with information in the high-order bit of octets. If such messages are transmitted in violation of this rule, receiving SMTP servers MAY clear the high-order bit or reject the message as invalid. In general, a relay SMTP SHOULD assume that the message content it has received is valid and, assuming that the envelope permits doing so, relay it without inspecting that content. Of course, if the content is mislabeled and the data path cannot accept the actual content, this may result in the ultimate delivery of a severely garbled message to the recipient. Delivery SMTP systems MAY reject such messages, or return them as undeliverable, rather than deliver them. In the absence of a server-offered extension explicitly permitting it, a sending SMTP system is not permitted to send envelope commands in any character set other than US-ASCII. Receiving systems SHOULD reject such commands, normally using "500 syntax error - invalid character" replies.

8-bit message content transmission MAY be requested of the server by a client using extended SMTP facilities, notably the "8BITMIME" extension, RFC 6152 [52]. 8BITMIME SHOULD be supported by SMTP servers. However, it MUST NOT be construed as authorization to transmit unrestricted 8-bit material, nor does 8BITMIME authorize transmission of any envelope material encoded in anything other than US-ASCII. 8BITMIME MUST NOT be requested by senders for material with the high bit on that is not in MIME format with an appropriate content-transfer encoding; servers MAY reject such messages.

The metalinguistic notation used in this document corresponds to the "Augmented BNF" used in other Internet mail system documents. The reader who is not familiar with that syntax should consult the ABNF specification in RFC 5234 [15]. Metalanguage terms used in running text are surrounded by pointed brackets (e.g., <CRLF>) for clarity. The reader is cautioned that the grammar expressed in the metalanguage is not comprehensive. There are many instances in which provisions in the text constrain or otherwise modify the syntax or semantics implied by the grammar.

3. The SMTP Procedures: An Overview

This section contains descriptions of the procedures used in SMTP: session initiation, mail transaction, forwarding mail, verifying mailbox names and expanding mailing lists, and opening and closing exchanges. Comments on relaying, a note on mail domains, and a discussion of changing roles are included at the end of this section. Several complete scenarios are presented in Appendix C.

3.1. Session Initiation

An SMTP session (or "mail session") is initiated when a client opens a connection to a server and the server responds with an opening message.

SMTP server implementations MAY include identification of their software and version information in the connection greeting reply after the 220 code, a practice that permits more efficient isolation and repair of any problems. Implementations MAY make provision for SMTP servers to disable the software and version announcement where it causes security concerns. While some systems also identify their contact point for mail problems, this is not a substitute for maintaining the required "postmaster" address (see Section 4).

The SMTP protocol allows a server to formally reject a mail session while still allowing the initial connection as follows: a 521 response MAY be given in the initial connection opening message instead of the 220. A server taking this approach MUST still wait for the client to send a QUIT (see Section 4.1.1.10) before closing the connection and SHOULD respond to any intervening commands with "503 bad sequence of commands". Since an attempt to make an SMTP connection to such a system is probably in error, a server returning a 521 response on connection opening SHOULD provide enough information in the reply text to facilitate debugging of the sending system. See Section 4.2.4.2.

3.2. Client Initiation

Once the server has sent the greeting (welcoming) message and the client has received it, the client normally sends the EHLO command to the server, indicating the client's identity. In addition to opening the session, use of EHLO indicates that the client is able to process service extensions and requests that the server provide a list of the extensions it supports. Older SMTP systems that are unable to support service extensions, and contemporary clients that do not require service extensions in the mail session being initiated, MAY use HELO instead of EHLO. Servers MUST NOT return the extended EHLO-style response to a HELO command. For a particular connection attempt, if the server returns a "command not recognized" response to EHLO, the client SHOULD be able to fall back and send HELO.

In the EHLO (or HELO) command, the host sending the command identifies itself; the command may be interpreted as saying "Hello, I am <domain>" (and, in the case of EHLO, "and I support service extension requests").

3.3. Mail Transactions

There are three steps to normal SMTP mail transactions. The transaction starts with a MAIL command that gives the sender identification. (In general, the MAIL command may be sent only when no mail transaction is in progress; see Section 4.1.4.) In a normal session, a series of one or more RCPT commands follows, giving the receiver information. Then, a DATA command initiates transfer of the mail data and is terminated by the "end of mail" data indicator, which also confirms (and terminates) the transaction.

Mail transactions are also terminated by the RSET command (Section 4.1.1.5), the sending of an EHLO or the equivalent HELO command (Section 3.2), or the sending of a QUIT command (Section 3.8). The QUIT command terminates not only any active mail transaction but the SMTP connection itself.

The first step in the procedure is the MAIL command.

```
MAIL FROM:<reverse-path> [SP <mail-parameters> ] <CRLF>
```

This command tells the SMTP-receiver that a new mail transaction is starting and to reset all its state tables and buffers, including any recipients or mail data. The <reverse-path> portion of the first or only argument contains the source mailbox (between "<" and ">" brackets), which can be used to report errors (see Section 4.2 for a discussion of error reporting). If accepted, the SMTP server returns a "250 OK" reply. If the mailbox specification is not acceptable for

some reason, the server MUST return a reply indicating whether the failure is permanent (i.e., will occur again if the client tries to send the same address again) or temporary (i.e., the address might be accepted if the client tries again later). Despite the apparent scope of this requirement, there are circumstances in which the acceptability of the reverse-path may not be determined until one or more forward-paths (in RCPT commands) can be examined. In those cases, the server MAY reasonably accept the reverse-path (with a 250 reply) and then report problems after the forward-paths are received and examined. Normally, failures produce 550 or 553 replies.

Historically, the <reverse-path> was permitted to contain more than just a mailbox; however source routing is now deprecated (see Appendix D.2).

The optional <mail-parameters> are associated with negotiated SMTP service extensions (see Section 2.2).

The second step in the procedure is the RCPT command. This step of the procedure can be repeated any number of times.

```
RCPT TO:<forward-path> [ SP <rcpt-parameters> ] <CRLF>
```

The first or only argument to this command includes a forward-path (normally a mailbox local-part and domain, always surrounded by "<" and ">" brackets) identifying one recipient. If accepted, the SMTP server returns a "250 OK" reply and stores the forward-path. If the recipient is known not to be a deliverable address, the SMTP server returns a 550 reply, typically with a string such as "no such user - " and the mailbox name (other circumstances and reply codes are possible).

Historically, the <forward-path> was permitted to contain a source routing list of hosts and the destination mailbox; however, source routes are now deprecated (see Appendix D.2). Clients MUST NOT assume that any SMTP server on the Internet can be used as their mail processing (relaying) site. If a RCPT command appears without a previous MAIL command, the server MUST return a 503 "Bad sequence of commands" response. The optional <rcpt-parameters> are associated with negotiated SMTP service extensions (see Section 2.2).

There are two ways that sender-SMTPs can determine the next-hop system to which to send the message. One is to use the DNS and MX records as described in Section Section 5.1. The other involves a next-hop destination or choice of destinations that are configured into the sender-SMTP to deal with special circumstances such as forwarding all messages to a particular host for further processing.

Since it has been a common source of errors, it is worth noting that spaces are not permitted on either side of the colon following FROM in the MAIL command or TO in the RCPT command. The syntax is exactly as given above.

The third step in the procedure is the DATA command (or some alternative specified in a service extension).

```
DATA <CRLF>
```

If accepted, the SMTP server returns a 354 Intermediate reply and considers all succeeding lines up to but not including the end of mail data indicator to be the message text. When the end of text is successfully received and stored, the SMTP-receiver sends a "250 OK" reply.

Since the mail data is sent on the transmission channel, the end of mail data must be indicated so that the command and reply dialog can be resumed. An SMTP client indicates the end of the mail data by sending a line containing only a "." (period or full stop, hex 2E), that is the character sequence "<CRLF>.<CRLF>". A transparency procedure is used to prevent this from interfering with the user's text (see Section 4.5.2).

The end of mail data indicator also confirms the mail transaction and tells the SMTP server to now process the stored recipients and mail data. If accepted, the SMTP server returns a "250 OK" reply. The DATA command can fail at only two points in the protocol exchange:

If there was no MAIL, or no RCPT, command, or all such commands were rejected, the server MAY return a "command out of sequence" (503) or "no valid recipients" (554) reply in response to the DATA command. If one of those replies (or any other 5yz reply) is received, the client MUST NOT send the message data; more generally, message data MUST NOT be sent unless a 354 reply is received.

If the verb is initially accepted and the 354 reply issued, the DATA command should fail only if the mail transaction was incomplete (for example, no recipients), if resources were unavailable (including, of course, the server unexpectedly becoming unavailable), or if the server determines that the message should be rejected for policy or other reasons.

However, in practice, some servers do not perform recipient verification until after the message text is received. These servers SHOULD treat a failure for one or more recipients as a "subsequent failure" and return a mail message as discussed in Section 6 and, in particular, in Section 6.1. Using a "550 mailbox not found" (or equivalent) reply code after the data are accepted makes it difficult or impossible for the client to determine which recipients failed.

When the RFC 822 format ([17], [16]) is being used, the mail data include the header fields such as those named Date, Subject, To, Cc, and From. Server SMTP systems SHOULD NOT reject messages based on perceived defects in the RFC 822 or MIME (RFC 2045 [30]) message header section or message body. In particular, they MUST NOT reject messages in which the numbers of Resent-header fields do not match or Resent-to appears without Resent-from and/or Resent-date.

Mail transaction commands MUST be used in the order discussed above.

3.4. Address Modification and Expansion

3.4.1. Forwarding for Address Correction or Updating

Forwarding support is most often required to consolidate and simplify addresses within, or relative to, some enterprise and less frequently to establish addresses to link a person’s prior address with a current one. Silent forwarding of messages (without server notification to the sender), for security or non-disclosure purposes, is common in the contemporary Internet.

In both the enterprise and the "new address" cases, information hiding (and sometimes security) considerations argue against exposure of the "final" address through the SMTP protocol as a side effect of the forwarding activity. This may be especially important when the final address may not even be reachable by the sender. Consequently, the "forwarding" mechanisms described in Section 3.2 of RFC 821, and especially the 251 (corrected destination) and 551 reply codes from RCPT must be evaluated carefully by implementers and, when they are available, by those configuring systems (see also Section 7.4).

In particular:

* Servers MAY forward messages when they are aware of an address change. When they do so, they MAY either provide address-updating information with a 251 code, or may forward "silently" and return a 250 code. However, if a 251 code is used, they MUST NOT assume that the client will actually update address information or even return that information to the user.

Alternately,

* Servers MAY reject messages or return them as non-deliverable when they cannot be delivered precisely as addressed. When they do so, they MAY either provide address-updating information with a 551 code, or may reject the message as undeliverable with a 550 code and no address-specific information. However, if a 551 code is used, they MUST NOT assume that the client will actually update address information or even return that information to the user.

SMTP server implementations that support the 251 and/or 551 reply codes SHOULD provide configuration mechanisms so that sites that conclude that they would undesirably disclose information can disable or restrict their use. See Section 7.4 for further discussion of that issue.

3.4.2. Aliases and Mailing Lists

Many SMTP-capable hosts support address expansion for multiple delivery via one or both of the alias and the list models.

When a message is delivered or forwarded to each address of an expanded list form, the return address in the envelope ("MAIL FROM:") MUST be changed to be the address of a person or other entity who administers the list. This change to the MAIL command does not affect the header section of the message.

An important mail facility is a mechanism for multi-destination delivery of a single message, by transforming (or "expanding" or "exploding") a pseudo-mailbox address into a list of destination mailbox addresses. When a message is sent to such a pseudo-mailbox (sometimes called an "exploder"), copies are forwarded or redistributed to each mailbox in the expanded list. Servers SHOULD simply utilize the addresses on the list; application of heuristics or other matching rules to eliminate some addresses, such as that of the originator, is strongly discouraged. We classify such a pseudo-mailbox as an "alias" or a "list", depending upon the expansion rules.

3.4.2.1. Simple Aliases

To expand an alias, the recipient mailer simply replaces the pseudo-mailbox address in the envelope with each of the expanded addresses in turn; the rest of the envelope and the message body are left unchanged. The message is then delivered or forwarded to each expanded address.

3.4.2.2. Mailing Lists

Processing of a mailing list may be said to operate by "redistribution" rather than by "forwarding" (as in the simple alias case in the subsection above). To expand a list, the recipient mailer replaces the pseudo-mailbox address in the envelope with each of the expanded addresses in turn. The return (backward-pointing) address in the envelope is changed so that all error messages generated by the final deliveries will be returned to a list administrator, not to the message originator, who generally has no control over the contents of the list and will typically find error messages annoying. Note that the key difference between handling simple aliases Section 3.4.2.1 and redistribution (this subsection) is the change to the backward-pointing address. When a system managing a list constrains its processing to the very limited set of modifications and actions described here, it is acting as part of an MTA; such list processing, like alias processing, can be treated as a continuation of email transit.

Mailing list management systems do exist that perform additional, sometimes extensive, modifications to a message and its envelope. Such mailing lists need to be viewed as MUAs that accept a message delivery and then submit a new message for multiple recipients.

3.5. Commands for Debugging Addresses

3.5.1. Overview

SMTP provides commands to verify a user name or obtain the content of a mailing list. This is done with the VRFY and EXPN commands, which have character string arguments. Implementations SHOULD support VRFY and EXPN (however, see Section 3.5.2 and Section 7.3).

For the VRFY command, the string is a user name or a user name and domain (see below). If a normal (i.e., 250) response is returned, the response MAY include the full name of the user and MUST include the mailbox of the user. It MUST be in one of the following forms:

```
User Name <local-part@domain>
<local-part@domain>
local-part@domain
```

When a name that is the argument to VRFY could identify more than one mailbox, the server MAY either note the ambiguity or identify the alternatives. In other words, any of the following are legitimate responses to VRFY:

```
553 User ambiguous
```

or

```
553- Ambiguous; Possibilities are
553-Joe Smith <jsmith@foo.com>
553-Harry Smith <hsmith@foo.com>
553 Melvin Smith <dweep@foo.com>
```

or

```
553-Ambiguous; Possibilities
553- <jsmith@foo.com>
553- <hsmith@foo.com>
553 <dweep@foo.com>
```

Under normal circumstances, a client receiving a 553 reply would be expected to expose the result to the user. Use of exactly the forms given, and the "user ambiguous" or "ambiguous" keywords, possibly supplemented by extended reply codes, such as those described in RFC 3463 [12], will facilitate automated translation into other languages as needed. Of course, a client that was highly automated or that was operating in another language than English might choose to try to translate the response to return some other indication to the user than the literal text of the reply, or to take some automated action such as consulting a directory service for additional information before reporting to the user.

For the EXPN command, the string identifies a mailing list, and the successful (i.e., 250) multiline response MAY include the full name of the users and MUST give the mailboxes on the mailing list.

In some hosts, the distinction between a mailing list and an alias for a single mailbox is a bit fuzzy, since a common data structure may hold both types of entries, and it is possible to have mailing lists containing only one mailbox. If a request is made to apply VRFY to a mailing list, a positive response MAY be given if a message so addressed would be delivered to everyone on the list, otherwise an error SHOULD be reported (e.g., "550 That is a mailing list, not a user" or "252 Unable to verify members of mailing list"). If a request is made to expand a user name, the server MAY return a positive response consisting of a list containing one name, or an error MAY be reported (e.g., "550 That is a user name, not a mailing list").

In the case of a successful multiline reply (normal for EXPN), exactly one mailbox is to be specified on each line of the reply. The case of an ambiguous request is discussed above.

"User name" is a fuzzy term and has been used deliberately. An implementation of the VRFY or EXPN commands MUST include at least recognition of local mailboxes as "user names". However, since current Internet practice often results in a single host handling mail for multiple domains, hosts, especially hosts that provide this functionality, SHOULD accept the "local-part@domain" form as a "user name"; hosts MAY also choose to recognize other strings as "user names".

The case of expanding a mailbox list requires a multiline reply, such as:

```
C: EXPN Example-People
S: 250-Jon Postel <Postel@isi.edu>
S: 250-Fred Fonebone <Fonebone@physics.foo-u.edu>
S: 250 Sam Q. Smith <SQSmith@specific.generic.com>
```

or

```
C: EXPN Executive-Washroom-List
S: 550 Access Denied to You.
```

The character string arguments of the VRFY and EXPN commands cannot be further restricted due to the variety of implementations of the user name and mailbox list concepts. On some systems, it may be appropriate for the argument of the EXPN command to be a file name for a file containing a mailing list, but again there are a variety of file naming conventions on the Internet. Similarly, historical variations in what is returned by these commands are such that the response should be interpreted very carefully, if at all, and SHOULD generally only be used for diagnostic purposes.

3.5.2. VRFY and EXPN Normal Response

When normal (2yz or 551) responses are returned from a VRFY or EXPN request, the reply MUST include the <Mailbox> name using a "<local-part@domain>" construction, where "domain" is a fully-qualified domain name. In circumstances exceptional enough to justify violating the intent of this specification, free-form text MAY be returned. In order to facilitate parsing by both computers and people, addresses SHOULD appear in pointed brackets. When addresses, rather than free-form debugging information, are returned, EXPN and VRFY MUST return only valid domain addresses that are usable in SMTP RCPT commands. Consequently, if an address implies delivery to a program or other system, the mailbox name used to reach that target MUST be given. Paths (explicit source routes) MUST NOT be returned by VRFY or EXPN.

Server implementations SHOULD support both VRFY and EXPN. For security reasons, implementations MAY provide local installations a way to disable either or both of these commands through configuration options or the equivalent (see Section 7.3). When these commands are supported, they are not required to work across relays when relaying is supported. Since they were both optional in RFC 821, but VRFY was made mandatory in RFC 1123 [10], if EXPN is supported, it MUST be listed as a service extension in an EHLO response. VRFY MAY be listed as a convenience but, since support for it is required, SMTP clients are not required to check for its presence on the extension list before using it.

3.5.3. Meaning of VRFY or EXPN Success Response

A server MUST NOT return a 250 code in response to a VRFY or EXPN command unless it has actually verified the address. In particular, a server MUST NOT return 250 if all it has done is to verify that the syntax given is valid. If only a syntax check is made, 502 (Command not implemented) or 500 (Syntax error, command unrecognized) SHOULD be returned. As stated elsewhere, implementation (in the sense of actually validating addresses and returning information) of VRFY and EXPN are strongly recommended. Hence, implementations that return 500 or 502 for VRFY are not in full compliance with this specification.

There may be circumstances where an address appears to be valid but cannot reasonably be verified in real time, particularly when a server is acting as a mail exchanger for another server or domain. "Apparent validity", in this case, would normally involve at least syntax checking and might involve verification that any domains specified were ones to which the host expected to be able to relay mail. In these situations, reply code 252 SHOULD be returned. These cases parallel the discussion of RCPT verification in Section 2.1. Similarly, the discussion in Section 3.4.1 applies to the use of reply codes 251 and 551 with VRFY (and EXPN) to indicate addresses that are recognized but that would be forwarded or rejected were mail received for them. Implementations generally SHOULD be more aggressive about address verification in the case of VRFY than in the case of RCPT, even if it takes a little longer to do so.

3.5.4. Semantics and Applications of EXPN

EXPN is often very useful in debugging and understanding problems with mailing lists and multiple-target-address aliases. Some systems have attempted to use source expansion of mailing lists as a means of eliminating duplicates. The propagation of aliasing systems with mail on the Internet for hosts (typically with MX and CNAME DNS records), for mailboxes (various types of local host aliases), and in

various proxying arrangements has made it nearly impossible for these strategies to work consistently, and mail systems SHOULD NOT attempt them.

3.6. Relaying and Mail Routing

3.6.1. Mail eXchange Records and Relaying

A relay SMTP server is usually the target of a DNS MX record that designates it, rather than the final delivery system. The relay server may accept or reject the task of relaying the mail in the same way it accepts or rejects mail for a local user. If it accepts the task, it then becomes an SMTP client, establishes a transmission channel to the next SMTP server specified in the DNS (according to the rules in Section 5), and sends it the mail. If it declines to relay mail to a particular address for policy reasons, a 550 response SHOULD be returned.

This specification does not deal with the verification of return paths. Server efforts to verify a return path and actions to be taken under various circumstances are outside the scope of this specification.

It is important to note that MX records can point to SMTP servers that act as gateways into other environments, not just SMTP relays and final delivery systems; see Sections 3.7 and 5.

If an SMTP server has accepted the task of relaying the mail and later finds that the destination is incorrect or that the mail cannot be delivered for some other reason, then it MUST construct an "undeliverable mail" notification message and send it to the originator of the undeliverable mail (as indicated by the reverse-path). Formats specified for non-delivery reports by other standards (see, for example, RFC 3461 [40] and RFC 3464 [41]) SHOULD be used if possible.

This notification message must be from the SMTP server at the relay host or the host that first determines that delivery cannot be accomplished. Of course, SMTP servers MUST NOT send notification messages about problems transporting notification messages. One way to prevent loops in error reporting is to specify a null reverse-path in the MAIL command of a notification message. When such a message is transmitted, the reverse-path MUST be set to null (see Section 4.5.5 for additional discussion). A MAIL command with a null reverse-path appears as follows:

```
MAIL FROM:<>
```

As discussed in Section 6.4, a relay SMTP has no need to inspect or act upon the header section or body of the message data and MUST NOT do so except to add its own "Received:" header field (Section 4.4.1) and possibly other trace header fields and, optionally, to attempt to detect looping in the mail system (see Section 6.3). Of course, this prohibition also applies to any modifications of these header fields or text (see also Section 7.9).

3.6.2. Message Submission Systems as Relays

Many mail-sending clients exist, especially in conjunction with facilities that receive mail via POP3 or IMAP, that have limited capability to support some of the requirements of this specification, such as the ability to queue messages for subsequent delivery attempts. For these clients, it is common practice to make private arrangements to send all messages to a single server for processing and subsequent distribution. SMTP, as specified here, is not ideally suited for this role. A standardized mail submission protocol has been developed that is gradually superseding practices based on SMTP (see RFC 6409 [49]). In any event, because these arrangements are private and fall outside the scope of this specification, they are not described here.

3.7. Mail Gatewaying

While the relay function discussed above operates within the Internet SMTP transport service environment, MX records or various forms of explicit routing may require that an intermediate SMTP server perform a translation function between one transport service and another. As discussed in Section 2.3.10, when such a system is at the boundary between two transport service environments, we refer to it as a "gateway" or "gateway SMTP".

Gatewaying mail between different mail environments, such as different mail formats and protocols, is complex and does not easily yield to standardization. However, some general requirements may be given for a gateway between the Internet and another mail environment.

3.7.1. Header Fields in Gatewaying

Header fields MAY be rewritten when necessary as messages are gatewayed across mail environment boundaries. This may involve inspecting the message body or interpreting the local-part of the destination address in spite of the prohibitions in Section 6.4.

Other mail systems gatewayed to the Internet often use a subset of the RFC 822 header section or provide similar functionality with a different syntax, but some of these mail systems do not have an equivalent to the SMTP envelope. Therefore, when a message leaves the Internet environment, it may be necessary to fold the SMTP envelope information into the message header section. A possible solution would be to create new header fields to carry the envelope information (e.g., "X-SMTP-MAIL:" and "X-SMTP-RCPT:"); however, this would require changes in mail programs in foreign environments and might risk disclosure of private information (see Section 7.2).

3.7.2. Received Lines in Gatewaying

When forwarding a message into or out of the Internet environment, a gateway MUST prepend a Received: line ("header field", see Section 4.4.1), but it MUST NOT alter in any way a Received: line that is already in the header section.

"Received:" header fields of messages originating from other environments may not conform exactly to this specification. However, the most important use of Received: lines is for debugging mail faults, and this debugging can be severely hampered by well-meaning gateways that try to "fix" a Received: line. As another consequence of trace header fields arising in non-SMTP environments, receiving systems MUST NOT reject mail based on the format of a trace header field and SHOULD be extremely robust in the light of unexpected information or formats in those header fields.

The gateway SHOULD indicate the environment and protocol in the "via" clauses of Received header field(s) that it supplies.

3.7.3. Addresses in Gatewaying

From the Internet side, the gateway SHOULD accept all valid address formats in SMTP commands and in the RFC 822 header section, and all valid RFC 822 messages. Addresses and header fields generated by gateways MUST conform to applicable standards (including this one and RFC5322bis [16]). Gateways are, of course, subject to the same rules for handling source routes as those described for other SMTP systems in Section 3.3.

3.7.4. Other Header Fields in Gatewaying

The gateway MUST ensure that all header fields of a message that it forwards into the Internet mail environment meet the requirements for Internet mail. In particular, all addresses in "From:", "To:", "Cc:", etc., header fields MUST be transformed (if necessary) to satisfy the standard header syntax of RFC5322bis [16], MUST reference only fully-qualified domain names, and MUST be effective and useful for sending replies. The translation algorithm used to convert mail from the Internet protocols to another environment's protocol SHOULD ensure that error messages from the foreign mail environment are delivered to the reverse-path from the SMTP envelope, not to an address in the "From:", "Sender:", or similar header fields of the message.

3.7.5. Envelopes in Gatewaying

Similarly, when forwarding a message from another environment into the Internet, the gateway SHOULD set the envelope return path in accordance with an error message return address, if supplied by the foreign environment. If the foreign environment has no equivalent concept, the gateway must select and use a best approximation, with the message originator's address as the default of last resort.

3.7.6. Other Gateway Issues

In general, gateways between the Internet and other mail systems SHOULD attempt to preserve any layering semantics across the boundaries between the two mail systems involved. Gateway-translation approaches that attempt to take shortcuts by mapping (such as mapping envelope information from one system to the message header section or body of another) have generally proven to be inadequate in important ways. Systems translating between environments that do not support both envelopes and a header section and Internet mail must be written with the understanding that some information loss is almost inevitable.

3.8. Terminating Sessions and Connections

An SMTP connection is terminated when the client sends a QUIT command. The server responds with a positive reply code, after which it closes the connection.

An SMTP server MUST NOT intentionally close the connection under normal operational circumstances (see Section 7.8) except:

* After receiving a QUIT command and responding with a 221 reply.

* After detecting the need to shut down the SMTP service and returning a 421 reply code. This reply code can be issued after the server receives any command or, if necessary, asynchronously from command receipt (on the assumption that the client will receive it after the next command is issued).

* After a timeout, as specified in Section 4.5.3.2, occurs waiting for the client to send a command or data.

In particular, a server that closes connections in response to commands that are not understood is in violation of this specification. Servers are expected to be tolerant of unknown commands, issuing a 500 reply and awaiting further instructions from the client.

An SMTP server that is forcibly shut down via external means SHOULD attempt to send a line containing a 421 reply code to the SMTP client before exiting. The SMTP client will normally read the 421 reply code after sending its next command.

SMTP clients that experience a connection close, reset, or other communications failure due to circumstances not under their control (in violation of the intent of this specification but sometimes unavoidable) SHOULD, to maintain the robustness of the mail system, treat the mail transaction as if a 421 response had been received and act accordingly.

There are circumstances, contrary to the intent of this specification, in which an SMTP server may receive an indication that the underlying TCP connection has been closed or reset. To preserve the robustness of the mail system, SMTP servers should be prepared for this condition and SHOULD treat it as if a QUIT had been received before the connection disappeared.

4. The SMTP Specifications

4.1. SMTP Commands

4.1.1. Command Semantics and Syntax

The SMTP commands define the mail transfer or the mail system function requested by the user. SMTP commands are character strings terminated by <CRLF>. The commands themselves are alphabetic characters terminated by <SP> if parameters follow and <CRLF> otherwise. (In the interest of improved interoperability, SMTP receivers SHOULD tolerate trailing white space before the terminating <CRLF>.) The syntax of the local part of a mailbox MUST conform to receiver site conventions and the syntax specified in Section 4.1.2.

The SMTP commands are discussed below. The SMTP replies are discussed in Section 4.2.

A mail transaction involves several data objects that are communicated as arguments to different commands. The reverse-path is the argument of the MAIL command, the forward-path is the argument of the RCPT command, and the mail data is the argument of the DATA command. These arguments or data objects must be transmitted and held, pending the confirmation communicated by the end of mail data indication that finalizes the transaction. The model for this is that distinct buffers are provided to hold the types of data objects; that is, there is a reverse-path buffer, a forward-path buffer, and a mail data buffer. Specific commands cause information to be appended to a specific buffer, or cause one or more buffers to be cleared.

Several commands (RSET, DATA, QUIT) are specified as not permitting parameters. In the absence of specific extensions offered by the server and accepted by the client, clients MUST NOT send such parameters and servers SHOULD reject commands containing them as having invalid syntax.

4.1.1.1. Extended HELLO (EHLO) or HELLO (HELO)

These commands are used to identify the SMTP client to the SMTP server. The argument clause contains the fully-qualified domain name of the SMTP client, if one is available. In situations in which the SMTP client system does not have a meaningful domain name (e.g., when its address is dynamically allocated and no reverse mapping record is available), the client SHOULD send an address literal (see Section 4.1.3). Additional discussion of domain names in SMTP commands appears in Section 2.3.5.

RFC 2821, and some earlier informal practices, encouraged following the literal by information that would help to identify the client system. That convention was not widely supported, and many SMTP servers considered it an error. In the interest of interoperability, it is probably wise for servers to be prepared for this string to occur, but SMTP clients MUST NOT send it.

The SMTP server identifies itself to the SMTP client in the connection greeting reply and in the response to this command.

A client SMTP SHOULD start an SMTP session by issuing the EHLO command. If the SMTP server supports the SMTP service extensions, it will give a successful response, a failure response, or an error response. If the SMTP server, in violation of this specification, does not support any SMTP service extensions, it will generate an error response. Older client SMTP systems MAY, as discussed above,

use HELO (as specified in RFC 821) instead of EHLO, and servers MUST support the HELO command and reply properly to it. In any event, a client MUST issue HELO or EHLO before starting a mail transaction.

These commands, and a "250 OK" reply to one of them, confirm that both the SMTP client and the SMTP server are in the initial state, that is, there is no transaction in progress and all state tables and buffers are cleared.

Syntax:

```
ehlo           = "EHLO" SP ( Domain / address-literal ) CRLF

helo           = "HELO" SP Domain CRLF
```

Normally, the response to EHLO will be a multiline reply. Each line of the response contains a keyword and, optionally, one or more parameters. Following the normal syntax for multiline replies, these keywords follow the code (250) and a hyphen for all but the last line, and the code and a space for the last line. The syntax for a positive response, using the ABNF notation and terminal symbols of RFC 5234 [15], is:

```
ehlo-ok-rsp =    ( "250" SP Domain [ SP ehlo-greet ] CRLF )
                 / ( "250-" Domain [ SP ehlo-greet ] CRLF
                 *( "250-" ehlo-line CRLF )
                 "250" SP ehlo-line CRLF )

ehlo-greet =     1*(%d0-9 / %d11-12 / %d14-127)
                 ; string of any characters other than CR or LF

ehlo-line =      ehlo-keyword *( SP ehlo-param )

ehlo-keyword =   (ALPHA / DIGIT) *(ALPHA / DIGIT / "-")
                 ; additional syntax of ehlo-params depends on
                 ; ehlo-keyword

ehlo-param =     1*(%d33-126)
                 ; any CHAR excluding <SP> and all
                 ; control characters (US-ASCII 0-31 and 127
                 ; inclusive)
```

Although EHLO keywords may be specified in upper, lower, or mixed case, they MUST always be recognized and processed in a case-insensitive manner. This is simply an extension of practices specified in RFC 821 and Section 2.4.

The EHLO response MUST contain keywords (and associated parameters if required) for all commands not listed as "required" in Section 4.5.1.

4.1.1.2. MAIL (MAIL)

This command is used to initiate a mail transaction in which the mail data is delivered to an SMTP server that may, in turn, deliver it to one or more mailboxes or pass it on to another system (possibly using SMTP). The argument clause contains a reverse-path and may contain optional parameters. In general, the MAIL command may be sent only when no mail transaction is in progress, see Section 4.1.4.

The reverse-path consists of the sender mailbox. Historically, that mailbox might optionally have been preceded by a list of hosts, but that behavior is now deprecated (see Appendix D.2). In some types of reporting messages for which a reply is likely to cause a mail loop (for example, mail delivery and non-delivery notifications), the reverse-path may be null (see Section 3.6).

This command clears the reverse-path buffer, the forward-path buffer, and the mail data buffer, and it inserts the reverse-path information from its argument clause into the reverse-path buffer.

If service extensions were negotiated, the MAIL command may also carry parameters associated with a particular service extension.

Syntax:

```
mail = "MAIL FROM:" Reverse-path
               [SP Mail-parameters] CRLF
```

4.1.1.3. RECIPIENT (RCPT)

This command is used to identify an individual recipient of the mail data; multiple recipients are specified by multiple uses of this command. The argument clause contains a forward-path and may contain optional parameters.

The forward-path consists of the required destination mailbox. When mail reaches its ultimate destination, the SMTP server inserts it into the destination mailbox in accordance with its host mail conventions.

Prior versions of the SMTP specification included text and examples in this section of use of the deprecated source route construct. If desired, see Appendix D.2 for discussion of that mechanism.

This command appends its forward-path argument to the forward-path buffer; it does not change the reverse-path buffer nor the mail data buffer.

For example, mail received at relay host xyz.com with envelope commands

```
MAIL FROM:<userx@y.foo.org>
RCPT TO:<userc@d.bar.org>
```

will result in a DNS lookup for d.bar.org and transmission to the host specified in the most-preferred MX record that is available (or by the address record if there are no MX records). It will use envelope commands identical to the above, i.e.,

```
MAIL FROM:<userx@y.foo.org>
RCPT TO:<userc@d.bar.org>
```

Since hosts are not required to relay mail at all, xyz.com MAY also reject the message entirely when the RCPT command is received, using a 550 code (since this is a "policy reason").

If the SMTP server determines that a message sent to the mailbox in the forward-path is not deliverable, it MUST either return an appropriate reply code (see Section 4.2.2) or generate a non-delivery notification.

If there were multiple failed recipients, either a single notification listing all of the failed recipients or separate notification messages MUST be sent for each failed recipient. For economy of processing by the sender, the former SHOULD be used when possible. All notification messages about undeliverable mail MUST be sent using the MAIL command and MUST use a null return path as discussed in Section 3.6.

If service extensions were negotiated, the RCPT command may also carry parameters associated with a particular service extension offered by the server. The client MUST NOT transmit parameters other than those associated with a service extension offered by the server in its EHLO response.

Syntax:

```
rcpt = "RCPT TO:" ( "<Postmaster@" Domain ">" / "<Postmaster>" /
                Forward-path ) [SP Rcpt-parameters] CRLF
```

Note that, in a departure from the usual rules for local-parts, the "Postmaster" string shown above is treated as case-insensitive.

4.1.1.4. DATA (DATA)

The receiver normally sends a 354 response to DATA, and then treats the lines (strings ending in <CRLF> sequences, as described in Section 2.3.8) following the command as mail data from the sender. This command causes the mail data to be appended to the mail data buffer. Unless some other character or non-character encoding is negotiated with an SMTP extension, the mail data may contain any of the 128 ASCII character codes. Experience has indicated that use of ASCII or ASCII-derived control characters other than SP, HT, CR, and LF may cause problems and SHOULD be avoided when possible.

The mail data are terminated by a line containing only a period, that is, the character sequence "<CRLF>.<CRLF>", where the first <CRLF> is actually the terminator of the previous line (see Section 4.5.2). This is the end of mail data indication. The first <CRLF> of this terminating sequence is also the <CRLF> that ends the final line of the data (message text) or, if there was no mail data, ends the DATA command itself (the "no mail data" case does not conform to this specification since it would require that neither the trace header fields required by this specification nor the message header section required by RFC 5322bis [16] be transmitted). An extra <CRLF> MUST NOT be added, as that would cause an empty line to be added to the message. The only exception to this rule would arise if the message body were passed to the originating SMTP-sender with a final "line" that did not end in <CRLF>; in that case, the originating SMTP system MUST either reject the message as invalid or add <CRLF> in order to have the receiving SMTP server recognize the "end of data" condition.

The custom of accepting lines ending only in <LF>, as a concession to non-conforming behavior on the part of some UNIX systems, has proven to cause more interoperability problems than it solves, and SMTP server systems MUST NOT do this, even in the name of improved robustness. In particular, the sequence "<LF>.<LF>" (bare line feeds, without carriage returns) MUST NOT be treated as equivalent to <CRLF>.<CRLF> as the end of mail data indication.

Receipt of the end of mail data indication requires the server to process the stored mail transaction information. This processing consumes the information in the reverse-path buffer, the forward-path buffer, and the mail data buffer, and on the completion of this command these buffers are cleared. If the processing is successful, the receiver MUST send an OK reply. If the processing fails, the receiver MUST send a failure reply. The SMTP model does not allow for partial failures at this point: either the message is accepted by the server for delivery and a positive response is returned or it is not accepted and a failure reply is returned (see Section 4.4.3 for additional discussion). In sending a positive "250 OK" completion

reply to the end of data indication, the receiver takes full responsibility for the message (see Section 6.1). Errors that are diagnosed subsequently MUST be reported in a mail message.

The server must give special treatment to cases in which processing following the end of mail data indication is only partially successful. This could happen if, after accepting several recipients and the mail data, the SMTP server finds that the mail data could be successfully delivered to some, but not all, of the recipients. In such cases, the response to the DATA command MUST be an OK reply. However, the SMTP server MUST compose and send an "undeliverable mail" notification message to the originator of the message.

When the SMTP server accepts a message either for relaying or for final delivery, it inserts a trace record (also referred to interchangeably as a "time stamp line", "Received" line, or "Received:" header field) at the top of the mail data. This trace record indicates the identity of the host that sent the message, the identity of the host that received the message (and is inserting this time stamp), and the date and time the message was received. Relayed messages will have multiple time stamp lines. Details for formation of these lines, including their syntax, is specified in Section 4.4.

Additional discussion about the operation of the DATA command appears in Section 3.3.

Syntax:

```
data = "DATA" CRLF
```

4.1.1.5. RESET (RSET)

This command specifies that the current mail transaction will be aborted. Any stored sender, recipients, and mail data MUST be discarded, and all buffers and state tables cleared. The receiver MUST send a "250 OK" reply to a RSET command with no arguments. A reset command may be issued by the client at any time. It is effectively equivalent to a NOOP (i.e., it has no effect) if issued immediately after EHLO or HELO, before either of those commands is issued in the session, after an end of data indicator has been sent and acknowledged, or immediately before a QUIT. An SMTP server MUST NOT close the connection as the result of receiving a RSET; that action is reserved for QUIT (see Section 4.1.1.10).

Since EHLO implies some additional processing and response by the server, RSET will normally be more efficient than reissuing that command, even though the formal semantics are the same.

Syntax:

```
rset = "RSET" CRLF
```

4.1.1.6. VERIFY (VRFY)

This command asks the receiver to confirm that the argument identifies a user or mailbox. If it is a user name, information is returned as specified in Section 3.5.

This command has no effect on the reverse-path buffer, the forward-path buffer, or the mail data buffer.

Syntax:

```
vrfy = "VRFY" SP String CRLF
```

4.1.1.7. EXPAND (EXPN)

This command asks the receiver to confirm that the argument identifies a mailing list, and if so, to return the membership of that list. If the command is successful, a reply is returned containing information as described in Section 3.5. This reply will have multiple lines except in the trivial case of a one-member list.

This command has no effect on the reverse-path buffer, the forward-path buffer, or the mail data buffer, and it may be issued at any time.

Syntax:

```
expn = "EXPN" SP String CRLF
```

4.1.1.8. HELP (HELP)

This command causes the server to send helpful information to the client. The command MAY take an argument (e.g., any command name) and return more specific information as a response.

This command has no effect on the reverse-path buffer, the forward-path buffer, or the mail data buffer, and it may be issued at any time.

SMTP servers SHOULD support HELP without arguments and MAY support it with arguments.

Syntax:

```
help = "HELP" [ SP String ] CRLF
```

4.1.1.9. NOOP (NOOP)

This command does not affect any parameters or previously entered commands. It specifies no action other than that the receiver send a "250 OK" reply.

This command has no effect on the reverse-path buffer, the forward-path buffer, or the mail data buffer, and it may be issued at any time. If a parameter string is specified, servers SHOULD ignore it.

Syntax:

```
noop = "NOOP" [ SP String ] CRLF
```

4.1.1.10. QUIT (QUIT)

This command specifies that the receiver MUST send a "221 OK" reply, and then close the transmission channel.

The receiver MUST NOT intentionally close the transmission channel until it receives and replies to a QUIT command (even if there was an error). The sender MUST NOT intentionally close the transmission channel until it sends a QUIT command, and it SHOULD wait until it receives the reply (even if there was an error response to a previous command). If the connection is closed prematurely due to violations of the above or system or network failure, the server MUST cancel any pending transaction, but not undo any previously completed transaction, and generally MUST act as if the command or transaction in progress had received a temporary error (i.e., a 4yz response).

The QUIT command may be issued at any time. Any current uncompleted mail transaction will be aborted.

Syntax:

```
quit = "QUIT" CRLF
```

4.1.1.11. Mail-Parameter and Rcpt-Parameter Error Responses

If the server SMTP does not recognize or cannot implement one or more of the parameters associated with a particular MAIL or RCPT command, it will return code 555.

If, for some reason, the server is temporarily unable to accommodate one or more of the parameters associated with a MAIL or RCPT command, and if the definition of the specific parameter does not mandate the use of another code, it should return code 455.

Errors specific to particular parameters and their values will be specified in the document that defines the parameter.

4.1.2. Command Argument Syntax

The syntax of the argument clauses of the above commands (using the syntax specified in RFC 5234 [15] where applicable) is given below. Some terminals not defined in this document, but are defined elsewhere, specifically:

* In the "core" syntax in Appendix B of RFC 5234 [15]: ALPHA, CRLF, DIGIT, HEXDIG, and SP .

* In the message format syntax in RFC5322bis [16]: atext, CFWS, date-time, and FWS msg-id.

```
Reverse-path   = Path / "<>"

Forward-path   = Path

Path           = "<" Mailbox ">"

Mail-parameters  = esmtp-param *(SP esmtp-param)

Rcpt-parameters  = esmtp-param *(SP esmtp-param)

esmtp-param    = esmtp-keyword ["=" esmtp-value]

esmtp-keyword  = (ALPHA / DIGIT) *(ALPHA / DIGIT / "-")

esmtp-value    = 1*(%d33-60 / %d62-126)
                 ; any CHAR excluding "=", SP, and control
                 ; characters.  If this string is an email address,
                 ; i.e., a Mailbox, then the "xtext" syntax [40]
                 ; SHOULD be used.

Keyword        = Ldh-str

Argument       = Atom

Domain         = sub-domain *("." sub-domain)
```

```
sub-domain     = Let-dig [Ldh-str]

Let-dig        = ALPHA / DIGIT

Ldh-str        = *( ALPHA / DIGIT / "-" ) Let-dig

address-literal  = "[" ( IPv4-address-literal /
                IPv6-address-literal /
                General-address-literal ) "]"
                ; See Section 4.1.3

Mailbox        = Local-part "@" ( Domain / address-literal )

Local-part     = Dot-string / Quoted-string
                ; MAY be case-sensitive

Dot-string     = Atom *("."  Atom)

Atom           = 1*atext

Quoted-string  = DQUOTE 1*QcontentSMTP DQUOTE

QcontentSMTP   = qtextSMTP / quoted-pairSMTP

quoted-pairSMTP  = %d92 %d32-126
                ; i.e., backslash followed by any ASCII
                ; graphic (including itself) or SPace

qtextSMTP      = %d32-33 / %d35-91 / %d93-126
                ; i.e., within a quoted string, any
                ; ASCII graphic or space is permitted
                ; without backslash-quoting except
                ; double-quote and the backslash itself.

String         = Atom / Quoted-string
```

Note that the backslash, "\", is a quote character, which is used to indicate that the next character is to be used literally (instead of its normal interpretation). For example, "Joe\,Smith" indicates a single nine-character user name string with the comma being the fourth character of that string.

While the above definition for Local-part is relatively permissive, for maximum interoperability, a mailbox SHOULD NOT be defined with Local-part requiring (or using) the Quoted-string form or with the Local-part being case-sensitive. Further, when comparing a Local-part (e.g., to a specific mailbox name), all quoting MUST be treated as equivalent. A sending system SHOULD transmit the form that uses the minimum quoting possible.

For example, the following three local-parts are equivalent and MUST compare equal: "ab cd ef", "ab\ cd ef" and "ab\ \cd ef". Similarly, "fred" and fred (i.e., with and without quotes) MUST compare equal. White space reduction MUST NOT be applied to the Local-part by intermediate systems. As particular examples, systems that are not making final delivery MUST NOT make assumptions about the relationships among "ab cd"@example.com and "ab cd"@example.com or even " "@example.com and ""@example.com.

In the absence of extensions, systems MUST NOT define mailboxes in such a way as to require the use in SMTP of non-ASCII characters (octets with the high order bit set to one) or ASCII "control characters" (decimal value 0-31 and 127) [4][5]. Extensions have been standardized for such use [43][44]. When these extensions are not in use, these characters MUST NOT be used in MAIL or RCPT commands or other commands that require mailbox names.

To promote interoperability and consistent with long-standing guidance about conservative use of the DNS in naming and applications (e.g., see Section 2.3.1 of the base DNS document, RFC 1035 [7]), characters outside the set of alphabetic characters, digits, and hyphen MUST NOT appear in domain name labels for SMTP clients or servers. In particular, the underscore character is not permitted. SMTP servers that receive a command in which invalid character codes have been employed, and for which there are no other reasons for rejection, MUST reject that command with a 501 response (this rule, like others, could be overridden by appropriate SMTP extensions).

4.1.3. Address Literals

Sometimes a host is not known to the domain name system and communication (and, in particular, communication to report and repair the error) is blocked. To bypass this barrier, a special literal form of the address is allowed as an alternative to a domain name. For IPv4 addresses, this form uses four small decimal integers separated by dots and enclosed by brackets such as [192.0.2.1], which indicates an (IPv4) Internet Address in sequence-of-octets form. For IPv6 and other forms of addressing that might eventually be standardized, the form consists of a standardized "tag" that identifies the address syntax, a colon, and the address itself, in a

format specified as part of the relevant standards (i.e., RFC 5952 [14] for IPv6).

Specifically:

```
IPv4-address-literal  = Snum 3("."  Snum)

IPv6-address-literal  = "IPv6:" IPv6-addr

General-address-literal  = Standardized-tag ":" 1*dcontent

Standardized-tag  = Ldh-str
                  ; Standardized-tag MUST be specified in a
                  ; Standards-Track RFC and registered with IANA
                  ; See Section 8.1.2.

dcontent          = %d33-90 / ; Printable US-ASCII
                  %d94-126 ; excl. "[", "\", "]"

Snum              = 1*3DIGIT
                  ; representing a decimal integer
                  ; value in the range 0 through 255

IPv6-addr         = 6( h16 ":" ) ls32
                  / "::" 5( h16 ":" ) ls32
                  / [ h16 ] "::" 4( h16 ":" ) ls32
                  / [ *1( h16 ":" ) h16 ] "::" 3( h16 ":" ) ls32
                  / [ *2( h16 ":" ) h16 ] "::" 2( h16 ":" ) ls32
                  / [ *3( h16 ":" ) h16 ] "::" h16 ":" ls32
                  / [ *4( h16 ":" ) h16 ] "::" ls32
                  / [ *5( h16 ":" ) h16 ] "::" h16
                  / [ *6( h16 ":" ) h16 ] "::"
                  ; This definition is consistent with the one for
                  ; URIs [48].

ls32              = ( h16 ":" h16 ) / IPv4-address-literal
                  ; least-significant 32 bits of address

h16               = 1*4HEXDIG
                  ; 16 bits of address represented in hexadecimal
```

4.1.4. Order of Commands

There are restrictions on the order in which these commands may be used.

A session that will contain mail transactions MUST first be initialized by the use of the EHLO command. An SMTP server SHOULD accept commands for non-mail transactions (e.g., VRFY, EXPN, or NOOP) without this initialization.

An EHLO command MAY be issued by a client later in the session. If it is issued after the session begins and the EHLO command is acceptable to the SMTP server, the SMTP server MUST clear all buffers and reset the state exactly as if a RSET command had been issued (specifically, it terminates any mail transaction that was in progress, see Section 3.3). In other words, the sequence of RSET followed immediately by EHLO is redundant, but not harmful other than in the performance cost of executing unnecessary commands. However the response to an additional EHLO command MAY be different from that from prior ones; the client MUST rely only on the responses from the most recent EHLO command.

If the EHLO command is not acceptable to the SMTP server, 501, 500, 502, or 550 failure replies MUST be returned as appropriate. The SMTP server MUST stay in the same state after transmitting these replies that it was in before the EHLO was received.

The SMTP client MUST, if possible, ensure that the domain parameter to the EHLO command is a primary host name as specified for this command in Section 2.3.5. If this is not possible (e.g., when the client's address is dynamically assigned and the client does not have an obvious name), an address literal SHOULD be substituted for the domain name.

An SMTP server MAY verify that the domain name argument in the EHLO command has an address record matching the IP address of the client by looking up the domain name and making the comparison.

The NOOP, HELP, EXPN, VRFY, and RSET commands can be used at any time during a session, or without previously initializing a session. SMTP servers SHOULD process these normally (that is, not return a 503 code) even if no EHLO command has yet been received; clients SHOULD open a session with EHLO before sending these commands.

If these rules are followed, the example in RFC 821 that shows "550 access denied to you" in response to an EXPN command is incorrect unless an EHLO command precedes the EXPN or the denial of access is based on the client's IP address or other authentication or authorization-determining mechanisms.

A mail transaction begins with a MAIL command and then consists of one or more RCPT commands, and a DATA command, in that order. A mail transaction may be aborted by the RSET, a new EHLO, or the QUIT command.

SMTP extensions (see Section 2.2) may create additional commands that initiate, abort, or end the transaction. More generally, any new command MUST clearly document any effect it has on the transaction state.

There may be zero or more transactions in a session. The MAIL command MUST NOT be sent if a mail transaction is already open, i.e., it should be sent only if no mail transaction had been started in the session, or if the previous one successfully concluded with a successful DATA command, or if the previous one was aborted, e.g., with a RSET or new EHLO.

If the transaction beginning command argument is not acceptable, a 501 failure reply MUST be returned and the SMTP server MUST stay in the same state. If the commands in a transaction are out of order to the degree that they cannot be processed by the server, a 503 failure reply MUST be returned and the SMTP server MUST stay in the same state.

The last command in a session MUST be the QUIT command. The QUIT command SHOULD be used by the client SMTP to request connection closure, even when no session opening command was sent and accepted.

4.2. SMTP Replies

Replies to SMTP commands serve to ensure the synchronization of requests and actions in the process of mail transfer and to guarantee that the SMTP client always knows the state of the SMTP server. Every command MUST generate exactly one reply. Even the command pipelining extension mentioned in Section 2.1 does not change this; it merely allows several commands to be issued before the replies for each are sent together.

The details of the command-reply sequence are described in Section 4.3.

An SMTP reply consists of a three digit number (transmitted as three numeric characters) followed by some text unless specified otherwise in this document. The number is for use by automata to determine what state to enter next; the text is for the human user. The three digits contain enough encoded information that the SMTP client need not examine the text and may either discard it or pass it on to the user, as appropriate. Exceptions are as noted elsewhere in this

document. In particular, the 220, 221, 251, 421, and 551 reply codes are associated with message text that must be parsed and interpreted by machines. In the general case, the text may be receiver dependent and context dependent, so there are likely to be varying texts for each reply code. A discussion of the theory of reply codes is given in Section 4.2.1. Formally, a reply is defined to be the sequence: a three-digit code, <SP>, one line of text, and <CRLF>, or a multiline reply (as defined in the same section). Since, in violation of this specification, the text is sometimes not sent, clients that do not receive it SHOULD be prepared to process the code alone (with or without a trailing space character). Only the EHLO, EXPN, and HELP commands are expected to result in multiline replies in normal circumstances; however, multiline replies are allowed for any command.

In ABNF, server responses are:

```
Greeting       = ( "220 " (Domain / address-literal)
                 [ SP textstring ] CRLF ) /
                 ( "220-" (Domain / address-literal)
                 [ SP textstring ] CRLF
                 *( "220-" [ textstring ] CRLF )
                 "220" [ SP textstring ] CRLF )

textstring     = 1*(%d09 / %d32-126) ; HT, SP, Printable US-ASCII

Reply-line     = *( Reply-code "-" [ textstring ] CRLF )
                 Reply-code [ SP textstring ] CRLF

Reply-code     = %x32-35 %x30-35 %x30-39
```

where "Greeting" appears only in the 220 response that announces that the server is opening its part of the connection. (Other possible server responses upon connection follow the syntax of Reply-line.)

An SMTP server SHOULD send only the reply codes listed in this document or additions to the list as discussed below. An SMTP server SHOULD use the text shown in the examples in messages where the text is parsed and interpreted by machines, as discussed above.

An SMTP client MUST determine its actions only by the reply code, not by the text (except for the "change of address" 251 and 551 and, if necessary, 220, 221, and 421 replies); in the general case, any text, including no text at all (although senders SHOULD NOT send bare codes), MUST be acceptable. The space (blank) following the reply code is considered part of the text. A Sender-SMTP MUST first test the whole 3 digit reply code it receives, as well as any accompanying supplemental codes or information (see RFC 3463 [12] and RFC 5248

[51]). If the full reply code is not recognized, and the additional information is not recognized or missing, the Sender-SMTP MUST use the first digit (severity indication) of a reply code it receives.

The lists of codes that appear below MUST NOT be construed as permanent. Use of existing codes with enhanced status codes [12] is strongly preferred to adding new ones. For that reason and others, the addition of new codes should be a rare and significant activity. Such an addition should carefully specify the information (including enhanced status codes), to be included in the textual part of the response. Enhanced status codes are specified in RFC 3463, the updates or successors to that specification, and the associated registry [51]). If new codes are necessary, they may be added as the result of new Standards-Track specifications. Consequently, a sender-SMTP MUST be prepared to handle codes not specified in this document and MUST do so by interpreting the first digit only.

In the absence of extensions negotiated with the client, SMTP servers MUST NOT send reply codes whose first digits are other than 2, 3, 4, or 5. Clients that receive such out-of-range codes SHOULD normally treat them as fatal errors and terminate the mail transaction.

4.2.1. Reply Code Severities and Theory

The three digits of the reply code each have a special significance. The first digit denotes whether the response is good, bad, or incomplete. An unsophisticated SMTP client, or one that receives an unexpected code, will be able to determine its next action (proceed as planned, redo, retrench, etc.) by examining this first digit. An SMTP client that wants to know approximately what kind of error occurred (e.g., mail system error, command syntax error) may examine the second digit, which encodes responses in specific functional categories. The third digit and any supplemental information that may be present is reserved for the finest gradation of information.

There are four values for the first digit of the reply code:

2yz Positive Completion reply
: The requested action has been successfully completed. A new request may be initiated.

3yz Positive Intermediate reply
: The command has been accepted, but the requested action is being held in abeyance, pending receipt of further information. The SMTP client should send another command specifying this information. This reply is used in command sequence groups (i.e., in DATA).

4yz Transient Negative Completion reply
The command was not accepted, and the requested action did not occur. However, the error condition is temporary, and the action may be requested again. The sender should return to the beginning of the command sequence (if any). It is difficult to assign a meaning to "transient" when two different sites (receiver- and sender-SMTP agents) must agree on the interpretation. Each reply in this category might have a different time value, but the SMTP client SHOULD try again. A rule of thumb to determine whether a reply fits into the 4yz or the 5yz category (see below) is that replies are 4yz if they can be successful if repeated without any change in command form or in properties of the sender or receiver (that is, the command is repeated identically and the receiver does not put up a new implementation).

5yz Permanent Negative Completion reply
The command was not accepted and the requested action did not occur. The SMTP client SHOULD NOT repeat the exact request (in the same sequence). Even some "permanent" error conditions can be corrected, so the human user may want to direct the SMTP client to reinitiate the command sequence by direct action at some point in the future (e.g., after the spelling has been changed, or the user has altered the account status).

It is worth noting that the file transfer protocol (FTP) [19] uses a very similar code architecture and that the SMTP codes are based on the FTP model. However, SMTP uses a one-command, one-response model (while FTP is asynchronous) and FTP's 1yz codes are not part of the SMTP model.

The second digit encodes responses in specific categories:

x0z Syntax: These replies refer to syntax errors, syntactically correct commands that do not fit any functional category, and unimplemented or superfluous commands.

x1z Information: These are replies to requests for information, such as status or help.

x2z Connections: These are replies referring to the transmission channel.

x3z Unspecified.

x4z Unspecified.

x5z Mail system: These replies indicate the status of the receiver

mail system vis-a-vis the requested transfer or other mail system action.

The third digit gives a finer gradation of meaning in each category specified by the second digit. The list of replies illustrates this. Each reply text is recommended rather than mandatory, and may even change according to the command with which it is associated. On the other hand, the reply codes must strictly follow the specifications in this section. Receiver implementations should not invent new codes for slightly different situations from the ones described here, but rather adapt codes already defined.

For example, a command such as NOOP, whose successful execution does not offer the SMTP client any new information, will return a 250 reply. The reply is 502 when the command requests an unimplemented non-site-specific action. A refinement of that is the 504 reply for a command that is implemented, but that requests an unimplemented parameter.

The reply text may be longer than a single line; in these cases the complete text must be marked so the SMTP client knows when it can stop reading the reply. This requires a special format to indicate a multiple line reply.

The format for multiline replies requires that every line, except the last, begin with the reply code, followed immediately by a hyphen, "-" (also known as minus), followed by text. The last line will begin with the reply code, followed immediately by <SP>, optionally some text, and <CRLF>. As noted above, servers SHOULD send the <SP> if subsequent text is not sent, but clients MUST be prepared for it to be omitted.

For example:

```
250-First line
250-Second line
250-234 Text beginning with numbers
250 The last line
```

In a multiline reply, the reply code on each of the lines MUST be the same. It is reasonable for the client to rely on this, so it can make processing decisions based on the code in any line, assuming that all others will be the same. In a few cases, there is important data for the client in the reply "text". The client will be able to identify these cases from the current context.

4.2.2. Reply Codes by Function Groups (Second Digit)

500 Syntax error, command unrecognized (This may include errors such as command line too long)
501 Syntax error in parameters or arguments
502 Command not implemented (see Section 4.2.4.1)
503 Bad sequence of commands
504 Command parameter not implemented

211 System status, or system help reply
214 Help message (Information on how to use the receiver or the meaning of a particular non-standard command; this reply is useful only to the human user)

220 <domain> Service ready
221 <domain> Service closing transmission channel
421 <domain> Service not available, closing transmission channel (This may be a reply to any command if the service knows it must shut down)
521 <domain> No mail service here.

250 Requested mail action okay, completed
251 User not local; will forward to <forward-path> (See Section 3.4.1)
252 Cannot VRFY user, but will accept message and attempt delivery (See Section 3.5.3)
354 Start mail input; end with <CRLF>.<CRLF>
450 Requested mail action not taken: mailbox unavailable (e.g., mailbox busy or temporarily blocked for policy reasons, or server temporarily unavailable if returned before a mail transaction is started)
451 Requested action aborted: error in processing
452 Requested action not taken: insufficient system storage (preferred code for "too many recipients", see Section 4.5.3.1.10)
455 Server unable to accommodate parameters
550 Requested action not taken: mailbox unavailable (e.g., mailbox not found, no access, or command rejected for policy reasons)
552 Requested mail action aborted: exceeded storage allocation.
553 Requested action not taken: mailbox name not allowed (e.g., mailbox syntax incorrect). This code is also used as an "ambiguous mailbox" response to VRFY, see Section 3.5.1.
554 Transaction failed (Or, historically in the case of a connection-opening response, "No SMTP service here". 521 is now preferred for that function at connection-opening if the server never accepts mail.)
555 MAIL FROM/RCPT TO parameters not recognized or not implemented
556 No mail service at this domain.

4.2.3. Reply Codes in Numeric Order

211 System status, or system help reply

214 Help message (Information on how to use the receiver or the meaning of a particular non-standard command; this reply is useful only to the human user)

220 <domain> Service ready

221 <domain> Service closing transmission channel

250 Requested mail action okay, completed

251 User not local; will forward to <forward-path> (See Section 3.4.1)

252 Cannot VRFY user, but will accept message and attempt delivery (See Section 3.5.3)

354 Start mail input; end with <CRLF>.<CRLF>

421 <domain> Service not available, closing transmission channel (This may be a reply to any command if the service knows it must shut down)

450 Requested mail action not taken: mailbox unavailable (e.g., mailbox busy or temporarily blocked for policy reasons, or server temporarily unavailable if returned before a mail transaction is started)

451 Requested action aborted: local error in processing

452 Requested action not taken: insufficient system storage (also preferred code for "too many recipients", see Section 4.5.3.1.10)

455 Server unable to accommodate parameters

500 Syntax error, command unrecognized (This may include errors such as command line too long)

501 Syntax error in parameters or arguments

502 Command not implemented (see Section 4.2.4.1)

503 Bad sequence of commands

504 Command parameter not implemented

521 No mail service (See Section 4.2.4.2.)

550 Requested action not taken: mailbox unavailable (e.g., mailbox not found, no access, or command rejected for policy reasons)

551 User not local; please try <forward-path> (See Section 3.4.1)

552 Requested mail action aborted: exceeded storage allocation.

553 Requested action not taken: mailbox name not allowed (e.g., mailbox syntax incorrect). This code is also used as an "ambiguous mailbox" response to VRFY, see Section 3.5.1.

554 Transaction failed (Or, in the case of a connection-opening response, "No SMTP service here" although 521 is now preferred for the latter. See Section 4.2.4.2.)

555 MAIL FROM/RCPT TO parameters not recognized or not implemented

556 No mail service at this domain. (See Section 4.2.4.2.)

4.2.4. Some specific code situations and relationships

4.2.4.1. Reply Code 502

Questions have been raised as to when reply code 502 (Command not implemented) SHOULD be returned in preference to other codes. 502 SHOULD be used when the command is actually recognized by the SMTP server, but not implemented. If the command is not recognized, code 500 SHOULD be returned. Extended SMTP systems MUST NOT list capabilities in response to EHLO for which they will return 502 (or 500) replies.

4.2.4.2. "No mail accepted" situations and the 521, 554, 556, and 450 codes

Codes 521, 554, and 556 are all used to report different types of permanent "no mail accepted" situations. They differ as follows. 521 is an indication from a system answering on the SMTP port that it does not support SMTP service (a so-called "dummy server" as discussed in RFC 7504 [53] and elsewhere). Obviously, it requires that system exist and that a connection can be made successfully to it. Because a system that does not accept any mail cannot meaningfully accept a RCPT command, any commands (other than QUIT) issued after an SMTP server has issued a 521 reply are client (sender) errors.

When a domain does not intend to accept mail and wishes to publish that fact rather than being subjected to connection attempts, the best way to accomplish that is to use the "Null MX" convention. This is done by advertising a single MX RR (see Section 3.3.9 of RFC 1035 [7]) with an RDATA section consisting of preference number 0 and a zero-length label, written in master files as ".", as the exchange domain, to denote that there exists no mail exchanger for that domain. Reply code 556 is then used by a message submission or intermediate SMTP system (see Section 1.1) to report that it cannot forward the message further because it knows from the DNS entry that the recipient domain does not accept mail. If, despite publishing the DNS entry, the host associated with the server domain chooses to respond on the SMTP port, it SHOULD respond with the 556 code as well. The details of the Null MX convention were first defined in RFC 7505 [54]; see that document for additional discussion of the rationale for that convention.

Reply code 556 would normally be used in response to a RCPT command (or extension command with similar intent) when the SMTP system identifies a domain that it can (or has) determined never accepts mail. Other codes, including 554 and the temporary 450, are used for more transient situations and situations in which an SMTP server cannot or will not deliver to (or accept mail for) a particular system or mailbox for policy reasons rather than ones directly related to SMTP processing. The 450 code may also be used to reflect a server being temporarily unavailable at connection time or after the EHLO command is issued (i.e., before a mail transaction is initiated).

4.2.4.3. Reply Codes after DATA and the Subsequent <CRLF>.<CRLF>

When an SMTP server returns a positive completion status (2yz code) after the DATA command is completed with <CRLF>.<CRLF>, it accepts responsibility for:

* delivering the message (if the recipient mailbox exists), or

* if attempts to deliver the message fail due to transient conditions, retrying delivery some reasonable number of times at intervals as specified in Section 4.5.4.

* if attempts to deliver the message fail due to permanent conditions, or if repeated attempts to deliver the message fail due to transient conditions, returning appropriate notification to the sender of the original message (using the address in the SMTP MAIL command).

When an SMTP server returns a temporary error status (4yz) code after the DATA command is completed with <CRLF>.<CRLF>, it MUST NOT make a subsequent attempt to deliver that message. The SMTP client retains responsibility for the delivery of that message and may either return it to the user or requeue it for a subsequent attempt (see Section 4.5.4.1).

The text provided by the server as part of the reply SHOULD be designed to allow the user, or the user's agent, to interpret the return of a transient failure status (by mail message or otherwise) as a non-delivery indication, just as a permanent failure would be interpreted. If the client SMTP successfully handles these conditions, the user will not receive such a reply.

When an SMTP server returns a permanent error status (5yz) code after the DATA command is completed with <CRLF>.<CRLF>, it MUST NOT make any subsequent attempt to deliver the message. As with temporary error status codes, the SMTP client retains responsibility for the message, but SHOULD NOT again attempt delivery to the same server without user review of the message and response and appropriate intervention.

4.3. Sequencing of Commands and Replies

4.3.1. Sequencing Overview

The communication between the sender and receiver is an alternating dialogue, controlled by the sender. As such, the sender issues a command and the receiver responds with a reply. Unless other arrangements are negotiated through service extensions, the sender MUST wait for this response before sending further commands. One important reply is the connection greeting. Normally, a receiver will send a 220 "Service ready" reply when the connection is completed. The sender SHOULD wait for this greeting message before sending any commands.

Note: all the greeting-type replies have the official name (the fully-qualified primary domain name) of the server host as the first word following the reply code. Sometimes the host will have no meaningful name. See Section 4.1.3 for a discussion of alternatives in these situations.

For example,

```
220 ISIF.USC.EDU Service ready
```

or

220 mail.example.com SuperSMTP v 6.1.2 Service ready

or

220 [10.0.0.1] Clueless host service ready

The detailed discussion in the next section lists alternative success and failure replies for each command. These SHOULD be strictly adhered to. A receiver MAY substitute text in the replies, but the meanings and actions implied by the code numbers and by the specific command reply sequence MUST be preserved. However, in order to provide robustness as SMTP is extended and evolves, the discussion in Section 4.2.1 still applies: all SMTP clients MUST be prepared to accept any code that conforms to the discussion in that section and MUST be prepared to interpret it on the basis of its first digit only.

4.3.2. Command-Reply Sequences

Each command is listed with its usual possible replies. The prefixes used before the possible replies are "I" for intermediate, "S" for success, and "E" for error. Since some servers may generate other replies under special circumstances, and to allow for future extension, SMTP clients SHOULD, when possible, interpret only the first digit of the reply and MUST be prepared to deal with unrecognized reply codes by interpreting the first digit only. Unless extended using the mechanisms described in Section 2.2, SMTP servers MUST NOT transmit reply codes to an SMTP client that are other than three digits or that do not start in a digit between 2 and 5 inclusive.

These sequencing rules and, in principle, the codes themselves, can be extended or modified by SMTP extensions offered by the server and accepted (requested) by the client. However, if the target is more precise granularity in the codes, rather than codes for completely new purposes, the system described in RFC 3463 [12] SHOULD be used in preference to the invention of new codes.

In addition to the codes listed below, any SMTP command can return any of the following codes if the corresponding unusual circumstances are encountered:

500 For the "command line too long" case or if the command name was not recognized. Note that producing a "command not recognized" error in response to the required subset of these commands is a violation of this specification. Similarly, producing a "command too long" message for a command line shorter than 512 characters would violate the provisions of Section 4.5.3.1.4.

501 Syntax error in command or arguments. In order to provide for future extensions, commands that are specified in this document as not accepting arguments (DATA, RSET, QUIT) SHOULD return a 501 message if arguments are supplied in the absence of EHLO-advertised extensions.

421 Service shutting down and closing transmission channel

Specific sequences are:

CONNECTION ESTABLISHMENT

- S: 220
  E: 521, 554, 556, 450 (if the system receiving the connection attempt is able to answer but is temporarily not available to receive email)

EHLO or HELO

- S: 250
  E: 504 (a conforming implementation could return this code only in fairly obscure cases), 550, 502 (permitted only with an old-style server that does not support EHLO), 450 (see note immediately above under "CONNECTION ESTABLISHMENT")

MAIL

- S: 250
  E: 552, 451, 452, 550, 553, 503, 455, 555

RCPT

- S: 250, 251 (but see Section 3.4.1 for discussion of 251 and 551)
  E: 550, 551, 552 (obsolete for "too many recipients; see Section 4.5.3.1.10), 553, 450, 451, 452, 503, 455, 555

DATA

- I: 354 -> data -> S: 250

  - E: 552, 554, 451, 452

  - E: 450, 550 (rejections for policy reasons)

- E: 503, 554

RSET

- S: 250

VRFY

- S: 250, 251, 252
  E: 550, 551, 553, 502, 504

EXPN

- S: 250, 252
  E: 550, 500, 502, 504

HELP

- S: 211, 214
  E: 502, 504

NOOP

- S: 250

QUIT

- S: 221

4.4. Trace Information

When inserted by SMTP, trace information is used to provide an audit trail of message handling. In addition, it indicates a route back to the sender of the message.

4.4.1. Received Header Field (Time Stamp)

When an SMTP server receives a message for delivery or further processing, it MUST insert a trace field (often referred to as "time stamp" or "Received" information) at the beginning of the message content, as discussed in Section 4.1.1.4.

This line MUST be structured as follows:

* The FROM clause, which MUST be supplied in an SMTP environment, SHOULD contain both (1) the name of the source host as presented in the EHLO command and (2) an address literal containing the IP address of the source, determined from the TCP connection.

* The ID clause MAY contain an "@" as suggested in the Internet Message format specification [16], but this is not required.

* If the FOR clause appears, it MUST contain exactly one <path> entry, even when multiple RCPT commands have been given. Multiple <path>s raise some security issues and have been deprecated, see Section 7.2.

An Internet mail program MUST NOT change or delete a Received: line that was previously added to the message header section. SMTP servers MUST prepend Received lines to messages; they MUST NOT change the order of existing lines or insert Received lines in any other location.

As the Internet grows, comparability of Received header fields is important for detecting problems, especially slow relays. SMTP servers that create Received header fields SHOULD use explicit offsets in the dates (e.g., -0800), rather than time zone names of any type. Local time (with an offset) SHOULD be used rather than UTC when feasible. This formulation allows slightly more information about local circumstances to be specified. If UTC is needed, the receiver need merely do some simple arithmetic to convert the values. Use of UTC loses information about the time zone-location of the server. If it is desired to supply a time zone name, it SHOULD be included in a comment. If UTC is actually being supplied instead of the local time zone, it should be denoted by a time zone offset of "-0000". Time zones aligned with the prime meridian (e.g., "GMT") are shown as "+0000".

4.4.2. Return-path Header Field

When the delivery SMTP server makes the "final delivery" of a message, it MUST insert a return-path line at the beginning of the mail data. Here, final delivery means the message has left the SMTP environment. Normally, this would mean it had been delivered to the destination user or an associated mail drop, but in some cases it may be further processed and transmitted by another mail system.

It is possible for the mailbox in the return path to be different from the actual sender's mailbox, for example, if error responses are to be delivered to a special error handling mailbox rather than to the message sender. When mailing lists are involved, this arrangement is common and useful as a means of directing errors to the list maintainer rather than the message originator.

A message-originating SMTP system SHOULD NOT send a message that already contains a Return-path header field. SMTP servers performing a relay function MUST NOT inspect the message data, and especially not to the extent needed to determine if Return-path header fields are present. SMTP servers making final delivery MAY remove Return-path header fields before adding their own.

The primary purpose of the Return-path is to designate the address to which messages indicating non-delivery or other mail system failures are to be sent. For this to be unambiguous, exactly one return path SHOULD be present when the message is delivered. Systems using the syntax specified here with non-SMTP transports SHOULD designate an unambiguous address, associated with the transport envelope, to which error reports (e.g., non-delivery messages) should be sent.

It is sometimes difficult for an SMTP server to determine whether it is making final delivery since forwarding or other operations may occur after the message is accepted for delivery. Consequently, any further (forwarding, gateway, or relay) systems MAY remove the return path and rebuild the MAIL command as needed to ensure that exactly one such line appears in a delivered message.

4.4.3. Return-path, Non-SMTP Systems, and Gateways

When SMTP systems, especially relay ones that are receiving messages and then processing them for the next hop, special issues arise and care must be taken. In particular:

* a gateway from SMTP -> elsewhere SHOULD insert a return-path header field, unless it is known that the "elsewhere" transport also uses Internet domain addresses and maintains the envelope sender address separately.

* a gateway from elsewhere -> SMTP SHOULD delete any return-path header field present in the message, and either copy that information to the SMTP envelope or combine it with information present in the envelope of the other transport system to construct the reverse-path argument to the MAIL command in the SMTP envelope.

4.4.4. Additional Trace Fields

"Received" and "Return-path", defined above, are the only two trace fields that are part of SMTP. Additional trace fields, or variations on the definitions here for other mail transports, may be defined and registered as described in [I-D.ietf-emailcore-rfc5322bis].

4.4.5. Trace Information Summary and Analysis

The text above implies that the final mail data will begin with a return path line, followed by one or more time stamp lines. These lines will be followed by the rest of the mail data: first the balance of the mail header section and then the body (RFC 5322bis [16]).

The time stamp line and the return path line are formally defined as follows (the definitions for "FWS" and "CFWS" appear in RFC 5322bis [16]):

```
Return-path-line  = "Return-Path:" FWS Reverse-path <CRLF>

Time-stamp-line  = "Received:" FWS Stamp <CRLF>

Stamp          = From-domain By-domain Opt-info [CFWS] ";"
                 FWS date-time
                 ; where "date-time" is as defined in RFC5322bis [16]
                 ; but the "obs-" forms, especially two-digit
                 ; years, are prohibited in SMTP and MUST NOT be used.

From-domain    = "FROM" FWS Extended-Domain

By-domain      = CFWS "BY" FWS Extended-Domain

Extended-Domain  = Domain /
                 ( Domain FWS "(" TCP-info ")" ) /
                 ( address-literal FWS "(" TCP-info ")" )

TCP-info       = address-literal / ( Domain FWS address-literal )
                 ; Information derived by server from TCP connection
                 ; not client EHLO.

Opt-info       = [Via] [With] [ID] [For]
                 [Additional-Registered-Clauses]

Via            = CFWS "VIA" FWS Link

With           = CFWS "WITH" FWS Protocol

ID             = CFWS "ID" FWS ( Atom / msg-id )
                 ; msg-id is defined in RFC 5322bis [16]

For            = CFWS "FOR" FWS ( Path / Mailbox )

Additional-Registered-Clauses  = 1*(CFWS Atom FWS String)
                 ; See Section 8.1.4.

Link           = "TCP" / Addtl-Link

Addtl-Link     = Atom
```

```
                  ; Additional standard names for links are
                  ; registered with the Internet Assigned Numbers
                  ; Authority (IANA).  "Via" is primarily of value
                  ; with non-Internet transports.  SMTP servers
                  ; SHOULD NOT use unregistered names.

   Protocol       = "ESMTP" / "SMTP" / Addtl-Protocol

   Addtl-Protocol  = Atom
                  ; Additional standard names for protocols are
                  ; registered with the Internet Assigned Numbers
                  ; Authority (IANA) in the "mail parameters"
                  ; registry [13].  SMTP servers SHOULD NOT
                  ; use unregistered names.
```

4.5. Additional Implementation Issues

4.5.1. Minimum Implementation

In order to make SMTP workable, the following minimum implementation MUST be provided by all receivers. The following commands MUST be supported to conform to this specification:

EHLO
HELO
MAIL
RCPT
DATA
RSET
NOOP
QUIT
VRFY

Any system that includes an SMTP server supporting mail relaying or delivery MUST support the reserved mailbox "postmaster" as a case-insensitive local name. This postmaster address is not strictly necessary if the server always returns 554 on connection opening (as described in Section 3.1). The requirement to accept mail for postmaster implies that RCPT commands that specify a mailbox for postmaster at any of the domains for which the SMTP server provides mail service, as well as the special case of "RCPT TO:<Postmaster>" (with no domain specification), MUST be supported.

SMTP systems are expected to make every reasonable effort to accept mail directed to Postmaster from any other system on the Internet. In extreme cases -- such as to contain a denial of service attack or other breach of security -- an SMTP server may block mail directed to Postmaster. However, such arrangements SHOULD be narrowly tailored so as to avoid blocking messages that are not part of such attacks.

4.5.2. Transparency

Without some provision for data transparency, the character sequence "<CRLF>.<CRLF>" ends the mail text and cannot be sent by the user. In general, users are not aware of such "forbidden" sequences. To allow all user composed text to be transmitted transparently, the following procedures are used:

* Before sending a line of mail text, the SMTP client checks the first character of the line. If it is a period, one additional period is inserted at the beginning of the line.

* When a line of mail text is received by the SMTP server, it checks the line. If the line is composed of a single period, it is treated as the end of mail indicator. If the first character is a period and there are other characters on the line, the first character is deleted.

The mail data may contain any of the 128 ASCII characters. All characters are to be delivered to the recipient’s mailbox, including spaces, vertical and horizontal tabs, and other control characters. If the transmission channel provides an 8-bit byte (octet) data stream, the 7-bit ASCII codes are transmitted, right justified, in the octets, with the high-order bits cleared to zero.

In some systems, it may be necessary to transform the data as it is received and stored. This may be necessary for hosts that use a different character set than ASCII as their local character set, that store data in records rather than strings, or which use special character sequences as delimiters inside mailboxes. If such transformations are necessary, they MUST be reversible, especially if they are applied to mail being relayed.

4.5.3. Sizes and Timeouts

4.5.3.1. Size Limits and Minimums

There are several objects that have required minimum/maximum sizes. Every implementation MUST be able to receive objects of at least these sizes. Objects larger than these sizes SHOULD be avoided when possible. However, some Internet mail constructs such as encoded X.400 addresses (RFC 2156 [32]) will often require larger objects. Clients MAY attempt to transmit these, but MUST be prepared for a server to reject them if they cannot be handled by it. To the maximum extent possible, implementation techniques that impose no limits on the length of these objects should be used.

Extensions to SMTP may involve the use of characters that occupy more than a single octet each. This section therefore specifies lengths in octets where absolute lengths, rather than character counts, are intended.

4.5.3.1.1. Local-part

The maximum total length of a user name or other local-part is 64 octets.

4.5.3.1.2. Domain

The maximum total length of a domain name or number is 255 octets.

4.5.3.1.3. Path

The maximum total length of a reverse-path or forward-path is 256 octets (including the punctuation and element separators).

4.5.3.1.4. Command Line

The maximum total length of a command line including the command word and the <CRLF> is 512 octets. SMTP extensions may be used to increase this limit.

4.5.3.1.5. Reply Line

The maximum total length of a reply line including the reply code and the <CRLF> is 512 octets. More information may be conveyed through multiple-line replies.

4.5.3.1.6. Text Line

The maximum total length of a text line including the <CRLF> is 1000 octets (not counting the leading dot duplicated for transparency). This number may be increased by the use of SMTP Service Extensions.

4.5.3.1.7. Message Content

The maximum total length of a message content (including any message header section as well as the message body) MUST BE at least 64K octets. Since the introduction of Internet Standards for multimedia mail (RFC 2045 [30]), message lengths on the Internet have grown dramatically, and message size restrictions should be avoided if at all possible. SMTP server systems that must impose restrictions SHOULD implement the "SIZE" service extension of RFC 1870 [11], and SMTP client systems that will send large messages SHOULD utilize it when possible.

4.5.3.1.8. Recipient Buffer

The minimum total number of recipients that MUST be buffered is 100 recipients. Rejection of messages (for excessive recipients) with fewer than 100 RCPT commands is a violation of this specification. The general principle that relaying SMTP server MUST NOT, and delivery SMTP servers SHOULD NOT, perform validation tests on message header fields suggests that messages SHOULD NOT be rejected based on the total number of recipients shown in header fields. A server that imposes a limit on the number of recipients MUST behave in an orderly fashion, such as rejecting additional addresses over its limit rather than silently discarding addresses previously accepted. A client that needs to deliver a message containing over 100 RCPT commands SHOULD be prepared to transmit in 100-recipient "chunks" if the server declines to accept more than 100 recipients in a single message.

4.5.3.1.9. Treatment When Limits Exceeded

Errors due to exceeding these limits may be reported by using the reply codes. Some examples of reply codes are:

500 Line too long.

or

501 Path too long

or

452 Too many recipients (see below)

or

552 Too much mail data (historically also used for too many recipients (see below).

4.5.3.1.10. Too Many Recipients Code

RFC 821 [6] incorrectly listed the error where an SMTP server exhausts its implementation limit on the number of RCPT commands ("too many recipients") as having reply code 552. The correct reply code for this condition is 452. At the time RFC 5321 was written, the use of reply code 552 by servers was sufficiently common that client implementation were advised to simply treat it as if 452 had been sent. That advice is no longer necessary or useful.

When a conforming SMTP server encounters this condition, it has at least 100 successful RCPT commands in its recipient buffer. If the server is able to accept the message, then at least these 100 addresses will be removed from the SMTP client's queue. When the client attempts retransmission of those addresses that received 452 responses, at least 100 of these will be able to fit in the SMTP server's recipient buffer. Each retransmission attempt that is able to deliver anything will be able to dispose of at least 100 of these recipients.

If an SMTP server has an implementation limit on the number of RCPT commands and this limit is exhausted, it MUST use a reply code of 452. If the server has a configured site-policy limitation on the number of RCPT commands, it MAY instead use a 5yz reply code. In particular, if the intent is to prohibit messages with more than a site-specified number of recipients, rather than merely limit the number of recipients in a given mail transaction, it would be reasonable to return a 503 response to any DATA command received subsequent to the 452 code or to simply return the 503 after DATA without returning any previous negative response.

4.5.3.2. Timeouts

An SMTP client MUST provide a timeout mechanism. It MUST use per-command timeouts rather than somehow trying to time the entire mail transaction. Timeouts SHOULD be easily reconfigurable, preferably without recompiling the SMTP code. To implement this, a timer is set for each SMTP command and for each buffer of the data transfer. The latter means that the overall timeout is inherently proportional to the size of the message.

Based on extensive experience with busy mail-relay hosts, the minimum per-command timeout values SHOULD be as follows:

4.5.3.2.1. Initial 220 Message: 5 Minutes

An SMTP client process needs to distinguish between a failed TCP connection and a delay in receiving the initial 220 greeting message. Many SMTP servers accept a TCP connection but delay delivery of the 220 message until their system load permits more mail to be processed.

4.5.3.2.2. MAIL Command: 5 Minutes

4.5.3.2.3. RCPT Command: 5 Minutes

A longer timeout is required if processing of mailing lists and aliases is not deferred until after the message was accepted.

4.5.3.2.4. DATA Initiation: 2 Minutes

This is while awaiting the "354 Start Input" reply to a DATA command.

4.5.3.2.5. Data Block: 3 Minutes

This is while awaiting the completion of each TCP SEND call transmitting a chunk of data.

4.5.3.2.6. DATA Termination: 10 Minutes.

This is while awaiting the "250 OK" reply. When the receiver gets the final period terminating the message data, it typically performs processing to deliver the message to a user mailbox. A spurious timeout at this point would be very wasteful and would typically result in delivery of multiple copies of the message, since it has been successfully sent and the server has accepted responsibility for delivery. See Section 6.1 for additional discussion.

4.5.3.2.7. Server Timeout: 5 Minutes.

An SMTP server SHOULD have a timeout of at least 5 minutes while it is awaiting the next command from the sender.

4.5.4. Retry Strategies

The common structure of a host SMTP implementation includes user mailboxes, one or more areas for queuing messages in transit, and one or more daemon processes for sending and receiving mail. The exact structure will vary depending on the needs of the users on the host and the number and size of mailing lists supported by the host. We describe several optimizations that have proved helpful, particularly

for mailers supporting high traffic levels.

Any queuing strategy MUST include timeouts on all activities on a per-command basis. A queuing strategy MUST NOT send error messages in response to error messages under any circumstances.

4.5.4.1. Sending Strategy

The general model for an SMTP client is one or more processes that periodically attempt to transmit outgoing mail. In a typical system, the program that composes a message has some method for requesting immediate attention for a new piece of outgoing mail, while mail that cannot be transmitted immediately MUST be queued and periodically retried by the sender. A mail queue entry will include not only the message itself but also the envelope information.

The sender MUST delay retrying a particular destination after one attempt has failed. In general, the retry interval SHOULD be at least 30 minutes; however, more sophisticated and variable strategies will be beneficial when the SMTP client can determine the reason for non-delivery.

Retries continue until the message is transmitted or the sender gives up; the give-up time generally needs to be at least 4-5 days. It MAY be appropriate to set a shorter maximum number of retries for non-delivery notifications and equivalent error messages than for standard messages. The parameters to the retry algorithm MUST be configurable.

A client SHOULD keep a list of hosts it cannot reach and corresponding connection timeouts, rather than just retrying queued mail items.

Experience suggests that failures are typically transient (the target system or its connection has crashed), favoring a policy of two connection attempts in the first hour the message is in the queue, and then backing off to one every two or three hours.

The SMTP client can shorten the queuing delay in cooperation with the SMTP server. For example, if mail is received from a particular address, it is likely that mail queued for that host can now be sent. Application of this principle may, in many cases, eliminate the requirement for an explicit "send queues now" function such as ETRN, RFC 1985 [29].

The strategy may be further modified as a result of multiple addresses per host (see below) to optimize delivery time versus resource usage.

An SMTP client may have a large queue of messages for each unavailable destination host. If all of these messages were retried in every retry cycle, there would be excessive Internet overhead and the sending system would be blocked for a long period. Note that an SMTP client can generally determine that a delivery attempt has failed only after a timeout of several minutes, and even a one-minute timeout per connection will result in a very large delay if retries are repeated for dozens, or even hundreds, of queued messages to the same host.

At the same time, SMTP clients SHOULD use great care in caching negative responses from servers. In an extreme case, if EHLO is issued multiple times during the same SMTP connection, different answers may be returned by the server. More significantly, 5yz responses to the MAIL command MUST NOT be cached.

When a mail message is to be delivered to multiple recipients, and the SMTP server to which a copy of the message is to be sent is the same for multiple recipients, then only one copy of the message SHOULD be transmitted. That is, the SMTP client SHOULD use the command sequence: MAIL, RCPT, RCPT, ..., RCPT, DATA instead of the sequence: MAIL, RCPT, DATA, ..., MAIL, RCPT, DATA. However, if there are very many addresses, a limit on the number of RCPT commands per MAIL command MAY be imposed. This efficiency feature SHOULD be implemented.

Similarly, to achieve timely delivery, the SMTP client MAY support multiple concurrent outgoing mail transactions. However, some limit may be appropriate to protect the host from devoting all its resources to mail.

4.5.4.2. Receiving Strategy

The SMTP server SHOULD attempt to keep a pending listen on the SMTP port (specified by IANA as port 25) at all times. This requires the support of multiple incoming TCP connections for SMTP. Some limit MAY be imposed, but servers that cannot handle more than one SMTP transaction at a time are not in conformance with the intent of this specification.

As discussed above, when the SMTP server receives mail from a particular host address, it could activate its own SMTP queuing mechanisms to retry any mail pending for that host address.

4.5.5. Messages with a Null Reverse-Path

There are several types of notification messages that are required by existing and proposed Standards to be sent with a null reverse-path, namely non-delivery notifications as discussed in Section 3.6.1 and Section 3.6.2, other kinds of Delivery Status Notifications (DSNs, RFC 3461 [40]), and Message Disposition Notifications (MDNs, RFC 8098 [45]). All of these kinds of messages are notifications about a previous message, and they are sent to the reverse-path of the previous mail message. (If the delivery of such a notification message fails, that usually indicates a problem with the mail system of the host to which the notification message is addressed. For this reason, at some hosts the MTA is set up to forward such failed notification messages to someone who is able to fix problems with the mail system, e.g., via the postmaster alias.)

All other types of messages (i.e., any message which is not required by a Standards-Track RFC to have a null reverse-path) SHOULD be sent with a valid, non-null reverse-path.

Implementers of automated email processors should be careful to make sure that the various kinds of messages with a null reverse-path are handled correctly. In particular, such systems SHOULD NOT reply to messages with a null reverse-path, and they SHOULD NOT add a non-null reverse-path, or change a null reverse-path to a non-null one, to such messages when forwarding.

5. Address Resolution and Mail Handling

5.1. Locating the Target Host

Unless special circumstances exist as described in Section 3.3, once an SMTP client lexically identifies a domain to which mail will be delivered for processing (as described in Sections 2.3.5 and 3.6), a DNS lookup MUST be performed to resolve the domain name as specified in RFC 1035 [7] and RFC 1123 Section 5.3.5 [10]). The names are required to be fully-qualified domain names (FQDNs) as discussed in Section 2.3.5.

The lookup first attempts to locate an MX record associated with the name. If a CNAME record is found, the resulting name is processed as if it were the initial name. If a non-existent domain error is returned, this situation MUST be reported as an error. If a temporary error is returned, the message MUST be queued and retried later (see Section 4.5.4.1). If an empty list of MXs is returned, the address is treated as if it was associated with an implicit MX RR with a preference of 0, pointing to that host. If MX records are present, but none of them are usable, or the implicit MX is unusable, this situation MUST be reported as an error.

When the lookup succeeds, the mapping can result in a list of alternative delivery addresses rather than a single address. This can be due to multiple MX records, multihoming, or both. To provide reliable mail transmission, the SMTP client MUST be able to try (and be prepared to retry) each of the relevant addresses in this list in order (see below), until a delivery attempt succeeds. However, as discussed more generally in Section 7.8 there MAY also be a configurable limit on the number of alternate addresses that can be tried. In any case, the SMTP client SHOULD try at least two addresses.

If one or more MX RRs are found for a given name, SMTP systems MUST NOT utilize any address RRs associated with that name unless they are located using the MX RRs; the "implicit MX" rule above applies only if there are no MX records present. If MX records are present, but none of them are usable, this situation MUST be reported as an error. That domain name also MUST be a primary host name, i.e., it is not allowed to be an alias.

When a domain name associated with an MX RR is looked up and the associated data field obtained, the data field of that response MUST contain a domain name that conforms to the specifications of Section 2.3.5. That domain name, when queried, MUST return at least one address record (e.g., A or AAAA RR) that gives the IP address of the SMTP server to which the message should be directed. An MX record with a CNAME as its target is a misconfiguration, as explained in RFC 2181, Section 10.3 [33]. However, implementations SHOULD still process CNAME responses when received, since a significant number of servers on the Internet are configured with MX records pointing to CNAMEs.

Two types of information are used to rank the host addresses: multiple MX records, and multihomed hosts.

MX records contain a numerical preference indication that MUST be used in sorting if more than one such record appears. Lower numbers are more preferred than higher ones. The sender-SMTP MUST inspect

the list for any of the names or addresses by which it might be known in mail transactions. If a matching record is found, all records at that preference level and higher-numbered ones MUST be discarded from consideration. If there are no records left at that point, it is an error condition, and a 5yz reply code generated (terminating the mail transaction) or the message MUST be returned as undeliverable. If there is a single MX record at the most-preferred preference level, the data field associated with that record is used as the next destination. Otherwise, if there are multiple records with the same preference and there is no clear reason to favor one (e.g., by recognition of an easily reached address), then the sender-SMTP MUST randomize them to spread the load across multiple mail exchangers for a specific organization.

The destination host (from either the data field of the preferred MX record or from an address record found in an implicit MX) may be multihomed. In those cases the domain name resolver will return a list of alternative IP addresses. It is the responsibility of the domain name resolver interface to have ordered this list by decreasing preference if necessary, and the SMTP sender MUST try them in the order presented.

Although the capability to try multiple alternative addresses is required, specific installations may want to limit or disable the use of alternative addresses. The question of whether a sender should attempt retries using the different addresses of a multihomed host has been controversial. The main argument for using the multiple addresses is that it maximizes the likelihood of timely delivery, and indeed sometimes the likelihood of any delivery; the counter-argument is that it may result in unnecessary resource use. Note that resource use is also strongly determined by the sending strategy discussed in Section 4.5.4.1.

If an SMTP server receives a message with a destination for which it is a designated Mail eXchanger, it MAY relay the message (potentially after having rewritten the MAIL FROM and/or RCPT TO addresses), make final delivery of the message, or hand it off using some mechanism outside the SMTP-provided transport environment. Of course, neither of the latter require that the list of MX records be examined further.

If it determines that it should relay the message without rewriting the address, it MUST process the MX records as described above to determine candidates for delivery.

5.2. IPv6 and MX Records

In the contemporary Internet, SMTP clients and servers may be hosted on IPv4 systems, IPv6 systems, or dual-stack systems that are compatible with either version of the Internet Protocol. The host domains to which MX records point may, consequently, contain "A RR"s (IPv4), "AAAA RR"s (IPv6), or any combination of them. While RFC 3974 [47] discusses some operational experience in mixed environments, it was not comprehensive enough to justify standardization, and some of its recommendations appear to be inconsistent with this specification. The appropriate actions to be taken either will depend on local circumstances, such as performance of the relevant networks and any conversions that might be necessary, or will be obvious (e.g., an IPv6-only client need not attempt to look up A RRs or attempt to reach IPv4-only servers). Designers of SMTP implementations that might run in IPv6 or dual-stack environments should study the procedures above, especially the comments about multihomed hosts, and, preferably, provide mechanisms to facilitate operational tuning and mail interoperability between IPv4 and IPv6 systems while considering local circumstances.

6. Problem Detection and Handling

6.1. Reliable Delivery and Replies by Email

When the receiver-SMTP accepts a piece of mail (by sending a "250 OK" message in response to DATA), it is accepting responsibility for delivering or relaying the message. This is a serious responsibility. The message MUST be preserved in a way which is robust against predictable loss (such as reboots, server crashes, disk failures, or resource shortages) until the next system has taken responsibility, or the message has been deliberately discarded. Some reasons that a receiver-SMTP may choose not to deliver or relay a message are discussed in the next subsection and in Section 7.8.

If there is a delivery failure after acceptance of a message, the receiver-SMTP MUST formulate and mail a notification message. This notification MUST be sent using a null ("<>") reverse-path in the envelope. The recipient of this notification MUST be the address from the envelope return path (or the Return-Path: line). However, if this address is null ("<>"), the receiver-SMTP MUST NOT send a notification. Obviously, nothing in this section can or should prohibit local decisions (i.e., as part of the same system environment as the receiver-SMTP) to log or otherwise transmit information about null address events locally if that is desired.

Some delivery failures after the message is accepted by SMTP will be unavoidable. For example, it may be impossible for the receiving SMTP server to validate all the delivery addresses in RCPT command(s) due to a "soft" domain system error, because the target is a mailing list (see earlier discussion of RCPT), or because the server is acting as a relay and has no immediate access to the delivering system.

To avoid receiving duplicate messages as the result of timeouts, a receiver-SMTP MUST seek to minimize the time required to respond to the final <CRLF>.<CRLF> end of data indicator. See RFC 1047 [21] for a discussion of this problem.

6.2. Unwanted, Unsolicited, and "Attack" Messages

Utility and predictability of the Internet mail system requires that messages that can be delivered should be delivered, regardless of any syntax or other faults associated with those messages and regardless of their content. If they cannot be delivered, and cannot be rejected by the SMTP server during the SMTP transaction, they should be "bounced" (returned with non-delivery notification messages) as described above. In today's world, in which many SMTP server operators have discovered that the quantity of undesirable bulk email vastly exceeds the quantity of desired mail and in which accepting a message may trigger additional undesirable traffic by providing verification of the address, those principles may not be practical.

As discussed in Section 7.8 and Section 7.9 below, dropping mail without notification of the sender is permitted in practice. However, it is extremely dangerous and violates a long tradition and community expectations that mail is either delivered or returned. If silent message-dropping is misused, it could easily undermine confidence in the reliability of the Internet's mail systems. So silent dropping of messages should be considered only in those cases where there is very high confidence that the messages are seriously fraudulent, pose a significant risk, or are otherwise inappropriate.

To stretch the principle of delivery if possible even further, it may be a rational policy to not deliver mail that has an invalid return address, although the history of the network is that users are typically better served by delivering any message that can be delivered. Reliably determining that a return address is invalid can be a difficult and time-consuming process, especially if the putative sending system is not directly accessible or does not fully and accurately support VRFY and, even if a "drop messages with invalid return addresses" policy is adopted, it SHOULD be applied only when there is near-certainty that the return addresses are, in fact, invalid.

Conversely, if a message is rejected because it is found to contain hostile content (a decision that is outside the scope of an SMTP server as defined in this document), rejection ("bounce") messages SHOULD NOT be sent unless the receiving site is confident that those messages will be usefully delivered. The preference and default in these cases is to avoid sending non-delivery messages when the incoming message is determined to contain hostile content.

6.3. Loop Detection

Simple counting of the number of "Received:" header fields in a message has proven to be an effective, although rarely optimal, method of detecting loops in mail systems. SMTP servers using this technique SHOULD use a large rejection threshold, normally at least 100 Received entries. Whatever mechanisms are used, servers MUST contain provisions for detecting and stopping trivial loops.

6.4. Compensating for Irregularities

Unfortunately, variations, creative interpretations, and outright violations of Internet mail protocols do occur; some would suggest that they occur quite frequently. The debate as to whether a well-behaved SMTP receiver or relay should reject a malformed message, attempt to pass it on unchanged, or attempt to repair it to increase the odds of successful delivery (or subsequent reply) began almost with the dawn of structured network mail and shows no signs of abating. Advocates of rejection claim that attempted repairs are rarely completely adequate and that rejection of bad messages is the only way to get the offending software repaired. Advocates of "repair" or "deliver no matter what" argue that users prefer that mail go through it if at all possible and that there are significant market pressures in that direction. In practice, these market pressures may be more important to particular vendors than strict conformance to the standards, regardless of the preference of the actual developers.

The problems associated with ill-formed messages were exacerbated by the introduction of the split-UA mail reading protocols (Post Office Protocol (POP) version 2 [18], Post Office Protocol (POP) version 3 [28], IMAP version 2 [23], and PCMAIL [22]). These protocols encouraged the use of SMTP as a posting (message submission) protocol, and SMTP servers as relay systems for these client hosts (which are often only intermittently connected to the Internet). Historically, many of those client machines lacked some of the mechanisms and information assumed by SMTP (and indeed, by the mail format protocol, RFC 822 [17]). Some could not keep adequate track of time; others had no concept of time zones; still others could not identify their own names or addresses; and, of course, none could satisfy the assumptions that underlay RFC 822's conception of authenticated addresses.

In response to these weak SMTP clients, many SMTP systems now complete messages that are delivered to them in incomplete or incorrect form. This strategy is generally considered appropriate when the server can identify or authenticate the client, and there are prior agreements between them. By contrast, there is at best great concern about fixes applied by a relay or delivery SMTP server that has little or no knowledge of the user or client machine. Many of these issues are addressed by using a separate protocol, such as that defined in RFC 6409 [49], for message submission, rather than using originating SMTP servers for that purpose.

The following changes to a message being processed MAY be applied when necessary by an originating SMTP server, or one used as the target of SMTP as an initial posting (message submission) protocol:

* Addition of a message-id field when none appears

* Addition of a date, time, or time zone when none appears

* Correction of addresses to proper FQDN format

The less information the server has about the client, the less likely these changes are to be correct and the more caution and conservatism should be applied when considering whether or not to perform fixes and how. These changes MUST NOT be applied by an SMTP server that provides an intermediate relay function.

In all cases, properly operating clients supplying correct information are preferred to corrections by the SMTP server. In all cases, documentation SHOULD be provided in trace header fields and/or header field comments for actions performed by the servers.

7. Security Considerations

SMTP is not a secure protocol as that term is understood in the contemporary Internet. It was designed, and is specified in this document, to transport a mail message without any provisions to prevent or detect reading of messages, or alteration of their content, as they move from original sender to final recipient. Similarly, it contains no mechanisms to ensue that the identity of the message originator is real or used with authorization. In the last few decades, mechanisms have been developed as SMTP extensions or supplemental protocols to mitigate those problems and vulnerabilities. In addition to those outlined in the subsections below, the security considerations that motivate those mechanisms and their limitations are discussed in the Applicability Statement document [56]. Pointers to the individual documents (containing their own Security Considerations sections) appear there as well. See Section 1.3 for additional discussion.

7.1. Mail Security and Spoofing

The authenticity of SMTP mail is inherently insecure in that it is feasible for even fairly casual users to negotiate directly with receiving and relaying SMTP servers and create messages that will trick a naive recipient into believing that they came from somewhere else. Constructing such a message so that the "spoofed" behavior cannot be detected by an expert is somewhat more difficult, but not sufficiently so as to be a deterrent to someone who is determined and knowledgeable. Consequently, as knowledge of Internet mail increases, so does the knowledge that SMTP mail inherently cannot be authenticated, or integrity checks provided, at the transport level. Real mail security lies only in end-to-end methods involving the message bodies, such as those that use digital signatures (see RFC 1847 [26] and, e.g., Pretty Good Privacy (PGP) in RFC 9580 [50] or Secure/Multipurpose Internet Mail Extensions (S/MIME) in RFC 8551 [46]).

Various protocol extensions and configuration options that provide authentication at the transport level (e.g., from an SMTP client to an SMTP server) improve somewhat on the traditional situation described above. However, in general, they only authenticate one server to another rather than a chain of relays and servers, much less authenticating users or user machines. Consequently, unless they are accompanied by careful handoffs of responsibility in a carefully designed trust environment, they remain inherently weaker than end-to-end mechanisms that use digitally signed messages rather than depending on the integrity of the transport system.

Efforts to make it more difficult for users to set envelope return path and header "From" fields to point to valid addresses other than their own are largely misguided: they frustrate legitimate applications in which mail is sent by one user on behalf of another, in which error (or normal) replies should be directed to a special address, or in which a single message is sent to multiple recipients on different hosts. (Systems that provide convenient ways for users to alter these header fields on a per-message basis should attempt to establish a primary and permanent mailbox address for the user so that Sender header fields within the message data can be generated sensibly.)

This specification does not further address the authentication issues associated with SMTP other than to advocate that useful functionality not be disabled in the hope of providing some small margin of protection against a user who is trying to fake mail.

7.2. Hiding Addresses from Trace

Addresses that do not appear in the message header section may appear in the RCPT commands to an SMTP server for a number of reasons. The two most common involve the use of a mailing address as a "list exploder" (a single address that resolves into multiple addresses) and the appearance of "blind copies". When more than one RCPT command is present, and in order to avoid defeating some of the purpose of these mechanisms, SMTP clients and servers SHOULD NOT copy the RCPT command arguments into the header section, either as part of trace header fields or as informational or private-extension header fields. See Section 7.6 for discussion of some related issues.

There is no inherent relationship between either "reverse" (from the MAIL command) or "forward" (RCPT) addresses in the SMTP transaction ("envelope") and the addresses in the header section. Receiving systems SHOULD NOT attempt to deduce such relationships and use them to alter the header section of the message for delivery. The popular "Apparently-to" header field is a violation of this principle as well as a common source of unintended information disclosure and SHOULD NOT be used.

7.3. VRFY, EXPN, and Security

As discussed in Section 3.5, individual sites may want to disable either or both of VRFY or EXPN for security reasons (see below). As a corollary to the above, implementations that permit this MUST NOT appear to have verified addresses that are not, in fact, verified. If a site disables these commands for security reasons, the SMTP server MUST return a 252 response, rather than a code that could be confused with successful or unsuccessful verification.

Returning a 250 reply code with the address listed in the VRFY command after having checked it only for syntax violates this rule. Of course, an implementation that "supports" VRFY by always returning 550 whether or not the address is valid is equally not in conformance.

On the public Internet, the contents of mailing lists have become popular as an address information source for so-called "spammers." The use of EXPN to "harvest" addresses has increased as list administrators have installed protections against inappropriate uses of the lists themselves. However, VRFY and EXPN are still useful for authenticated users and within an administrative domain. For example, VRFY and EXPN are useful for performing internal audits of how email gets routed to check and to make sure no one is automatically forwarding sensitive mail outside the organization. Sites implementing SMTP authentication may choose to make VRFY and EXPN available only to authenticated requestors. Implementations SHOULD still provide support for EXPN, but sites SHOULD carefully evaluate the tradeoffs.

Whether disabling VRFY provides any real marginal security depends on a series of other conditions. In many cases, RCPT commands can be used to obtain the same information about address validity. On the other hand, especially in situations where determination of address validity for RCPT commands is deferred until after the DATA command is received, RCPT may return no information at all, while VRFY is expected to make a serious attempt to determine validity before generating a reply code (see discussion above).

7.4. Mail Rerouting Based on the 251 and 551 Reply Codes

Before a client uses the 251 or 551 reply codes from a RCPT command to automatically update its future behavior (e.g., updating the user's address book), it should be certain of the server's authenticity. If it does not, it may be subject to a man in the middle attack.

7.5. Information Disclosure in Announcements

There has been an ongoing debate about the tradeoffs between the debugging advantages of announcing server type and version (and, sometimes, even server domain name) in the greeting response or in response to the HELP command and the disadvantages of exposing information that might be useful in a potential hostile attack. The utility of the debugging information is beyond doubt. Those who argue for making it available point out that it is far better to actually secure an SMTP server rather than hope that trying to conceal known vulnerabilities by hiding the server's precise identity

will provide more protection. Sites are encouraged to evaluate the tradeoff with that issue in mind; implementations SHOULD minimally provide for making type and version information available in some way to other network hosts.

7.6. Information Disclosure in Trace Fields

In some circumstances, such as when mail originates from within a LAN whose hosts are not directly on the public Internet, trace (e.g., "Received") header fields produced in conformance with this specification may disclose host names and similar information that would not normally be available. This ordinarily does not pose a problem, but sites with special concerns about name disclosure should be aware of it. Also, the optional FOR clause should not be supplied when the same message is sent to multiple recipients in the same mail transaction in order not to inadvertently disclose the identities of "blind copy" recipients to others.

7.7. Information Disclosure in Message Forwarding

As discussed in Section 3.4.1, use of the 251 or 551 reply codes to identify the replacement address associated with a mailbox may inadvertently disclose sensitive information. Sites that are concerned about those issues should ensure that they select and configure servers appropriately.

7.8. Local Operational Requirements and Resistance to Attacks

In recent years, there has been an increase of attacks on SMTP servers, either in conjunction with attempts to discover addresses for sending unsolicited messages or simply to make the servers inaccessible to others (i.e., as an application-level denial of service attack). There may also be important local circumstances that justify departures from some of the limits specified in this documents especially ones involving maximums or minimums. While the means of doing so are beyond the scope of this Standard, rational operational behavior requires that servers be permitted to detect such attacks and take action to defend themselves. For example, if a server determines that a large number of RCPT commands are being sent, most or all with invalid addresses, as part of such an attack, it would be reasonable for the server to close the connection after generating an appropriate number of 5yz (normally 550) replies.

7.9. Scope of Operation of SMTP Servers

It is a well-established principle that an SMTP server may refuse to accept mail for any operational or technical reason that makes sense to the site providing the server. However, cooperation among sites and installations makes the Internet possible. If sites take excessive advantage of the right to reject traffic, the ubiquity of email availability (one of the strengths of the Internet) will be threatened; considerable care should be taken and balance maintained if a site decides to be selective about the traffic it will accept and process.

Relay function through arbitrary sites, as part of hostile efforts to hide the actual origins of mail, has become so common that most sites limit the use of the relay function to known or identifiable sources.

Implementations SHOULD provide the capability to perform this type of analysis of message sources and potential message rejection as a result. When mail is rejected for these or other policy reasons, a 550 code SHOULD be used in response to EHLO (or HELO), MAIL, or RCPT as appropriate.

8. IANA Considerations

All of the registries described below were created long before work began on the current version of this document. While the document specifies several changes to improve the information available, clarity, and organization of some of those registries (including explicitly specifying fields for the "Service Extensions" one (Section 8.1.1, little new has been added and this section was written to optimize community understanding of the registries and the changes being made. The final subsection below (Section 8.3) is a detailed summary for IANA, as they requested, of specifics of those changes.

8.1. SMTP-related Registries

IANA maintains several registries in support of this specification, each one described in a subsection below. The first two of them were originally created even before RFC 2821 was published in April 2001. The others were defined or expanded by RFC 5321. The subsections that follow describe the general purpose and intent for those registries and substantive modifications to existing ones. Information about actual registry structure requirements appears in Section 8.2.

8.1.1. Simple Mail Transfer Protocol (SMTP) Service Extensions

The "Simple Mail Transfer Protocol (SMTP) Service Extensions" registry [58], often referred to in this document as [the] "Service Extension Registry", consists of SMTP service extensions with the associated keywords, and, as needed, parameters, verbs, and related information.

8.1.1.1. Registration Models

In order to accommodate both a significant review of proposed extensions for those who find that useful and a minimally restrictive registration procedure for those who simply want to avoid name conflicts and similar problems, this registry supports two different registration models. Additional discussion of the reasons for this organization appears in Section 2.2.2.

As noted in Section 2.2.2, SMTP Extensions MUST be registered.

The would-be registrant shall pick between the two models described below. If the first is attempted and proves unsuccessful, the second may then be chosen:

Model 1: IETF Review and Approval

> The document goes through the normal IETF review and approval process, culminating in a published Standards Track, BCP, Experimental, or, in rare cases specifically approved by the IESG, an IETF Stream Informational RFC. The intent is that the extension and its specification will represent careful IETF community review and consensus on its technical merits, utility, and clarity of explanation. The change controller for all such extensions will be the IETF.
>
> This model is approximately equivalent to "IETF Review" as described in RFC 8126/ BCP 26 [3], but involves a stronger preference for a Standards Track or Experimental publication as a result.

Model 2: Simple Registration

> The sole purpose of this option is to get the extension name and contact information registered in order to minimize the risk of the same extension name being used for different purposes. The intent is that there be no barrier to such registrations other than the time and effort required to submit the request itself. Registrants are

encouraged to provide documentation of the extension, its interactions with other specifications, etc., and to consult individuals or groups with SMTP experience for advice, but none of that is required. The change controller for all such extensions will be the registrant unless otherwise specified in the registration request.

Even if this model is chosen, it is expected that registrants will supply all of the information in the list below and as described above and in Section 2.2.2 as either part of the registration or in supplemental documents that will be referenced from the registry. However, the primary goals of getting extensions registered according to this model are to avoid conflicts about naming (e.g., two different deployed extensions with the same name or keyword) and to either identify a stable and generally available specification or to establish contact information for additional information. Consequently, if no information is available for some of the listed items, notably Section 8.1.1.3, Paragraph 3, Item 8 and Section 8.1.1.3, Paragraph 3, Item 9, the registration should be made and the registry entry should show the absence of such missing data with "Not supplied" in the appropriate field.

This model is approximately equivalent to "First Come First Served" as described in RFC 8126/ BCP 26. [3]

IANA will modify the structure of the Service Extension Registry to add a column entry that will specify the model chosen. Cf. Section 8.1.1.3, Paragraph 3, Item 6

8.1.1.2. Add VRFY to the Registry

// RFC Editor: This section temporarily retained to preserve
// numbering and other references. But its contents are now part of
// Section 8.3.3.3 and this section should be removed by the RPC if
// not sooner.

8.1.1.3. SMTP Service Extension Registration Template

The following information shall be supplied as part of a Service Extension registration application and will be incorporated into the registry. Some information is optional if the second model (see above) is used but should be explicitly noted as not provided. Except as noted, a reference to an easily available and stable document may be provided rather than including the actual information

in the registry.

// 20240114: IANA recommended adding something like the following to
// the "RegMethod" definition below, but, if it is useful, it applies
// to the whole registration. It may not belong here either;
// suggestions welcome. I've also omitted explicit instructions for
// updating "Legacy" procedures as a waste of time: if Eric or anyone
// else wants to change those undocumented procedures, they should
// find a new keyword and register it properly. I'm even dubious
// about most modifications to FCFS registrations: if anything
// substantive is changed, it risks exactly the conflict between one
// definition (the old one) and another (the new one) that Model 2 is
// intended to avoid.
Once a registration is complete, any modification requests must follow the same procedure as the original registration except that "FCFS" registration can be changed to "IETF" ones if appropriate permissions are given and the "Model 1" procedures followed.

(1) EHLO Keyword: The textual name of the SMTP service extension EHLO Keyword values are ASCII case-insensitive letter-digit-hyphen strings, see "ehlo-keyword" ABNF non terminal from Section 4.1.1.1 for more details.

// 2025-01-14: Above added at IANA's request, confirmed by Alexey,
// but I strongly object as this clutters up the template further
// and, if carried to its logical extreme, copies much of the syntax
// from elsewhere in the document into the registry template. Worst
// case, I should figure out how to add the syntax for these entries
// to the index -- see example under "Extension Registry Template
// Components, EHLO Keyword" in the index below if the problem
// discussed in Appendix G does not get me/us.
(text must be incorporated in the registry);

(2) Description: A summary of the purpose of the extension and what it is expected to accomplish (text must be incorporated in the registry).

(3) EHLO Parameters: The syntax and possible values of any parameters associated with an extension keyword returned by the server in the EHLO command response. Explicitly note "None" if no parameters.

(4) Additional verbs: Any additional SMTP verbs associated with the extension and their arguments if any. These additional verbs will usually be, but are not required to be, the same as, or begin with, the EHLO keyword value, e.g., if the EHLO keyword value were "EXMP", an associated verb might be "EXMP-XYZ". See Section 2.4, Paragraph 2 and Section 4.1.2 of <<This Document<< for information about syntax. Identify as "none" if there are no additional verbs (text must be incorporated in the registry).

// RFC Editor: Note in draft, particularly to IANA as well as the
// RPC: The cross-reference to information in earlier sections of
// rfc5321bis was added here because it seemed particularly
// important. Similar cross-references can be added to additional
// registry entry types if they are felt to be necessary or this one
// can be dropped if it is not believe to be useful.

(5) MAIL/RCPT Parameter Values: Any new parameters the extension associates with the MAIL or RCPT verbs (text must be incorporated in the registry). Identify as "none" if the extension does not add any.

(6) RegMethod: Keyword values in the registry will indicate a level of approval as "IETF" or "FCFS" (designating, respectively, Model 1 of 2 as described in Section 8.1.1.1 or "Legacy" for the entries for "VERB" and "ONEX" (both approved exceptionally prior to this specification). All other registration approved prior to completion of this specification will have "IETF" in this field.

(7) Message submission Use and Values: Keyword indicating relevance for use in message submission as described in Section 7 of RFC 6409 [49]. For any registration prior to the publication of <<This document>> for which this information was not specified in RFC 6409 or the registration request, this entry in the registry will be set to "MUST NOT".

(8) Behavior and Impact: A description of how support for the extension affects the behavior of a server and client SMTP;

(9) Length Added: The increment by which the extension is increasing the maximum length of the commands MAIL and/or RCPT over that specified in this Standard or other registrations that might reasonably be expected to interact with it;

(10) Contact: Provides contact information for the submitter or other responsible party and identification of the change controller. Under most circumstances the two will be the same (e.g., "IETF" for SMTP extensions specified in IETF Stream RFCs) but might be identified separately for some "Model 2" registrations.

8.1.2. Address Literal Tags

The "Address Literal Tags" [62] registry consists of "tags" that identify forms of domain literals other than those for IPv4 addresses (specified in RFC 821 and in this document). This registry also goes back more than two decades. Its initial, and so far only, entry is for IPv6 addresses (usage and syntax are specified elsewhere in this document). No additional literal types are anticipated at this time. If that prediction is incorrect, registration is required and requires standardization (the IETF "Standards Action" procedure defined in RFC 8126/ BCP 26 [3]) before being used.

8.1.3. Mail Transmission Types

// RFC Editor (and other readers): being renamed to ’Mail // Transmission Types for the "Received:" header field’ (see // Section 8.3.4, Paragraph 1, Item 3).

The "Mail Transmission Types" registry group [59], established by RFC 821 and renewed by this specification, is a registry of link and protocol identifiers to be used with the "via" and "with" subclauses of the time stamp ("Received:" header field) described in Section 4.4. Link and protocol identifiers in addition to those specified in this document may be registered only according to the "RFC Required" procedure described in RFC 8126/ BCP 26 [3]. Each of "via", "with", and "Additional Registered Clauses (keyword and values) has its own subregistry.

// 20250114: When the registry structure was checked this evening,
// "Additional-registered-clauses" had already been incorporated into
// the "Mail Transmission Types" registry group, so the following
// paragraph/ instruction can presumably be dropped.
An additional subregistry has been added to the "VIA link types" and "WITH protocol types" subregistries of this registry to contain registrations of "Additional-registered-clauses" as described above and in the subsection that follows.

8.1.4. Additional Registered "Received:" Clauses

As mentioned in Section 4.4.4 above, additional clauses for the "Received:" header field may be added by future specifications (details below). IANA has created a registry for such clauses [60] which should be renamed to "Additional Registered 'Received:' Clauses" for clarity. The registry will contain the Clause Name; the name of any associated enabling Service Extension (blank if there is no Service Extension involved); a short description; a syntax summary; and a reference to a more complete specification. Only the field for the name of the enabling Service Extension is new with this specification.

Additional clauses may be registered only by the "IETF Review" procedure described in RFC 8126/ BCP 26 [3].

As new clauses are defined, they may, in principle, specify creation of separate registries (specific to them) if the Strings consist of reserved terms or keywords rather than less restricted strings.

8.2. Specification of Registry Group and Registry Structure

The Mail Parameters Registry Group [57] should be reorganized as follows.

// RFC Editor: Note in draft (this note should be removed by the RPC
// in final processing if not sooner): items in the list below that
// have been completely subsumed by items in Section 8.3 have been
// marked accordingly to preserve the numbers for review. Those that
// are left contain explanatory information for the provisions in
// Section 8.3. There is probably a better way to handle those
// explanations.

(1) The registry for Address Literals [62] should be consolidated into the Mail Parameters Registry Group. Those literals are specified only for email purposes and have no established meaning elsewhere.

(2) Please insert a category for "Associated registries located elsewhere" after the "Registries included below" group at the top of the MAIL Parameters Group page. There should be one entry in the new category: the name "Message Headers" and a link to the appropriate registry [61]. Also that registry group should be renamed to "SMTP-related Registries": there is no obvious reason to capitalize "MAIL": all the the registries and subregistries in the group are SMTP-related, and the most important of them is not about parameters at all. "MAIL Parameters" should probably be retained as a comment or parenthetical note because it is almost certainly referenced from other documents.

(3) ((Removed to Section 8.3))

(4) ((Removed to Section 8.3))

(5) Remaining numbered items in this subsection should be separated into different fields in the registry; i.e., none of them should be combined into "Description". Information that is not supplied with the registration or supplemental documents should be explicitly identified as "Not supplied by submitter" or with an explicit note pointing to that phrase.

(6) ((Removed to Section 8.3))

(7) ((Removed to Section 8.3))

(8) ((Removed to Section 8.3))

(9) ((Removed to Section 8.3))

8.3. Registry Changes with <<This Document>>

While, as noted at the beginning of this Section, this document does not introduce any new registries, it specifies modifications to several existing ones. Those modifications are listed below for IANA's convenience, in what should be a convenient order for applying them. Some of the information below is inevitably redundant with information and explanations supplied earlier in this Section or elsewhere in the document.

// RFC Editor: Note that the changes made as a result of this section
// will change the URLs, and sometimes even the names, of entries in
// the References section. Those references should be updated to
// reflect the new names and locations and this note removed.

8.3.1. Changes to the Registry for Address Literals

(1) Consolidate the Address Literals Registry into the "Mail Parameters" Registry Group.

(2) If needed to conform to other IANA practices, leave a note at the location of the existing/old registry explaining the change and pointing to the new location.

(3) Update any registries that point to this one (there probably aren't any) to reflect the new location and insure that URLs that reflect the old location point to the new one.

8.3.2. Changes to the top-level "MAIL Parameters" Registry Group

// RFC Editor: Unresolved as to whether this note should appear in
// the published version.
Historical note:
The quantity and complexity of the changes below are largely due to registry organization decisions made in the fairly distant past by IANA and, in retrospect, not made optimally. For example, Section 2.2.2 of RFC 2821 specified information that must be specified when extensions are registered. IANA chose to capture that information in four fields -- "EHLO Keyword", "Description", "Reference", and "Note" -- leaving some of it out and the rest to presumably be captured in the the references and notes. RFC 5321 carried that text forward, as did versions of the current document until the middle of 2022, when the WG started to make decisions to be more explicit about what belonged in the registry (content, not organization). When combined with IANA's expecting much more explicit instructions, the result became the more detailed and complex registry instructions below, reorganizing material that should have been present all along rather than demanding significant new material.

(1) Rename the registry group to "SMTP-related Registries", including a note about the older name.

(2) Update all registries that refer to this one to use the new name but insure that URLs based on the older name continue to work and are treated as permanent.

(3) Update the "Registries included below" list to reflect the addition of the Address Literals Registry as described in Section 8.3.1 above.

(4) After the "Registries included below" listing, add a new category named "Associated registries located elsewhere". It should have one entry, "Message Headers" [61].

(5) Registries not identified in subsections below, i.e., "Registered-states", "Multicast Email SMTP Extensions", and "SMTP Server Limits", are unchanged.

8.3.3. Changes to Simple Mail Transfer Protocol (SMTP) Service Extensions Registry

8.3.3.1. Registry Header Information Changes

(1) The "Reference" header information that refers to RFC 5321 Section 2.2 should be changed to reference Section 8.1.1 and Section 2.2.2 of this document in that order.

(2) A note should be added to the header information indicating what information fields may be supplied by reference to a stable and easily available external document rather than spelled out in the registry entry. Information about which fields these are appear in Section 8.1.1.3 and Section 8.1.1.3, Paragraph 3, Item 10.

(3) The "Registration Procedure(s)" header entry under "SMTP Serice Extensions" should be changed to "Either IETF review and approval or a variation on first come first serve, both described in Section 8.1.1.1 of <<This Document>>".

8.3.3.2. Fields for Registry Entries

(1) Add fields to the list of registered extensions "Additional verbs", "MAIL/RCPT Parameter Values" as specified in detail in Section 8.1.1.3 and "Contact" as described in Section 8.1.1.3, Paragraph 3, Item 10. For extensions registered prior to the date this document is posted, the value of those fields should be a reference to the document that now appears in the "Reference" field unless other information is readily available.

(2) Add a field for "EHLO Parameters" after the "MAIL/RCPT Parameter Values" one. The values for this field consist of additional information supplied by the server along with the EHLO keyword name. For registrations prior to publication of this document, the value will be copied from the former "SMTP Service Extension Parameters" registry, with "None" for any Service Extensions that did not appear in that registry.

(3) Add a field to the registry named "RegMethod" with values "IETF", "FCFS", and "Legacy", the first two reflecting the method choice in Section 8.1.1. For registrations prior to approval of this document, "Legacy" will be applied to the "VERB" and "ONEX" registrations and all others will have "IETF" in this field.

8.3.3.3. Additional Registry Entry

While it is not strictly an extension (nor is EXPN), to improve clarity IANA should add VRFY to this registry, immediately before the entry for EXPN. Fields should be:

EHLO keyword: VRFY
Description: VRFY command as specified in <<This document>>
Reference: <<This document>>
Note: Implementation support for VRFY is required in servers but its listing in the EHLO response is optional". See Section 3.5.2 in <<This document>> for details on this subject.
Parameters: None
Additional verbs: None
MAIL/RCPT Parameter Values: None
RegMethod: IETF
Message submission Use and Values: MUST NOT
Behavior and Impact: VRFY command as specified in <<This document>>
Length Added: Zero
Contact: IETF
Change Controller: IETF

8.3.4. Changes to Mail Transmission Types registry

(1) In the first sentence of the Note for the registry replace "The Simple Mail Transfer Protocol [RFC821][RFC5321] and the Standard for the Format of ARPA Internet Text Messages [RFC822] specify that..." with "The Simple Mail Transfer Protocol (<<This Document>>) and the Standard for Internet Message Formats [16] specify that..."

(2) The Reference in the registry header should be to Section 8.1.3 of <<This Document>> and [16]

(3) To improve clarity, the title of this registry should be changed to 'Mail Transmission Types for the "Received:" header field'.

9. Acknowledgments

Many people contributed to the development of RFCs 2821 and 5321. Those documents should be consulted for those acknowledgments.

Neither this document nor RFCs 2821 or 5321 would have been possible without the many contribution and insights of the late Jon Postel and Ned Freed. Jon Postel's contributions of course include the original specification of SMTP in RFC 821. A considerable quantity of text from RFC 821 still appears in this document as do several of Jon's original examples that have been updated only as needed to reflect other changes in the specification. Ned Freed's many contributions from multiple perspectives, as author or co-author of several of the documents that were folded into this one, and an extremely careful reader who identified and proposed corrections to problems that others missed, were similarly invaluable.

The following filed errata against RFC 5321 that were not rejected at the time of submission: Jasen Betts, Adrien de Croy, Guillaume Fortin-Debigare, Roberto Javier Godoy, David Romerstein, Dominic Sayers, Rodrigo Speller, Alessandro Vesely, and Brett Watson. Some of those individuals made additional suggestions after the EMAILCORE WG was initiated. In addition to the above, several of whom continued to make other suggestions, specific suggestions that led to corrections and improvements in early versions of the current specification were received from Dave Crocker, Ned Freed, Arnt Gulbrandsen, Tony Hansen, Barry Leiba, Ivar Lumi, Pete Resnick, Hector Santos, Paul Smith and others.

// RFC Editor: Despite the message to me on 2020-05-03 recommending
// the sentence form that follows, Last Call comments pointed out
// that, as a sentence starting without an upper-case letter, it
// looks odd. Borrowing from the rest of that discussion, it would
// look less obviously odd if rearranged so that the name was not the
// first word, but that would not solve the perceived (but not
// actual) problem of not capitalizing the name. This is not a
// useful IETF discussion: up to you if you are inclined to change
// the 2020 advice.
chetti contributed an analysis that clarified the ABNF productions that implicitly reference other documents.

The EMAILCORE Working Group was chartered in September 2020 with Alexey Melnikov and Seth Blank as co-chairs. Todd Herr replaced Seth Blank early in 2021. Without their leadership and technical contributions, and the efforts of WG participants under their guidance, this document would never have been completed. Many participants in the WG reviewed the document or portions of it and

made comments that resulted in improvement. During the last stages of working group consideration of this document, careful reviews of the specification in its entirety by Alexey Melnikov, John Levine, Rob Sayre, and Tim Wicinski contributed significantly to the clarity of the final version and its relationship to other IETF work.

Additional thanks are due to Alexey Melnikov for rewriting what is now Appendix E.2 into a form that should be intelligible for the long term and to Donald Eastlake for a comprehensive Last Call review that identified text to be clarified and issues that should have probably been better explained.

10. References

10.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[2] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[3] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[4] ANSI, "USA Code for Information Interchange", ANSI X3.4-1968, 1968. ANSI X3.4-1968 has been replaced by newer versions with slight modifications, but the 1968 version remains definitive for the Internet. The 1968 version is also described for Internet purposes in RFC 20 [5].

[5] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <https://www.rfc-editor.org/info/rfc20>.

[6] Postel, J., "Simple Mail Transfer Protocol", STD 10, RFC 821, DOI 10.17487/RFC0821, August 1982, <https://www.rfc-editor.org/info/rfc821>.

[7] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[8] Mockapetris, P., "Domain names - implementation and specification", Section 2.3.1, RFC 1035, November 1987, <https://www.rfc-editor.org/rfc/rfc1035>. // [RFC Editor: please remove this note]] [[Reviewers and other // readers: the mess in this annotation, with embedded "//" markings // as part of CREF comments and no separation, even with a line // break, of the annotation from the rest of the reference and of the // text starting with "This section of RFC 1035..", which is intended // to remain in the RFC form from the comments above it, are the // result of an xml2rfc bug that was reported in January 2024 and // that was considered not worth fixing because the comment will // disappear in the RFC version and, with luck, everything else will // work out. The separation of the intended text from the botched // comment in the txt form was accomplished by hand-editing the // latter. After a rather detailed analysis, there is no standards // track document that explicitly updates RFC 1035 (or 1034) to allow // leading digits in "preferred syntax" domain names. What we have // is a situation in which RFC 1123 allows leading digits in "host // names" but does not do so in the section that updates the DNS // specs. As pointed out in RFC 9499, "host name" has been used and // misused for several different things, including as only the first // (leftmost) label in a fully-qualified domain name. The discussion // of the "host name" terminology in that RFC has been read to settle // the issue by equating "host name" syntax to "preferred name // syntax", but it does not update 1123 and, equally important, uses // "often meant" to describe the relationship between the two terms // and, in the subsequent paragraph and as mentioned above, talks // about the other things that "host name" might mean (including only // the first label "of a fully-qualified domain name). The (non- // comment) text below in this reference is an attempt to explain the // situation without making claims about, e.g., document updates that // are not supported by RFC metadata or the structure of RFC 1123. This section of RFC 1035 defined the "preferred name syntax" as excluding leading digits in those names. Whether the restriction was accidental or deliberate, at least one second-level domain name starting with a digit had appeared in the DNS by the end of 1986, almost a year before RFC 1035 appeared. The restriction was removed for "host names" in RFC 1123 [10]. The terminology description for the DNS [9] clarified that "host name" was "often meant to be a domain name that follows the rules... of 'preferred name syntax'". So the syntax for "domain name" in Section 2.3.5 above is

consistent with the DNS specifications as generally interpreted.

[9] Hoffman, P. and K. Fujiwara, "DNS Terminology", BCP 219, RFC 9499, DOI 10.17487/RFC9499, March 2024, <https://www.rfc-editor.org/info/rfc9499>.

[10] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <https://www.rfc-editor.org/info/rfc1123>.

[11] Klensin, J., Freed, N., and K. Moore, "SMTP Service Extension for Message Size Declaration", STD 10, RFC 1870, DOI 10.17487/RFC1870, November 1995, <https://www.rfc-editor.org/info/rfc1870>.

[12] Vaudreuil, G., "Enhanced Mail System Status Codes", RFC 3463, DOI 10.17487/RFC3463, January 2003, <https://www.rfc-editor.org/info/rfc3463>.

[13] Newman, C., "ESMTP and LMTP Transmission Types Registration", RFC 3848, DOI 10.17487/RFC3848, July 2004, <https://www.rfc-editor.org/info/rfc3848>.

[14] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <https://www.rfc-editor.org/info/rfc5952>.

[15] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[16] Resnick, P., "Internet Message Format", 2024, <https://datatracker.ietf.org/doc/draft-ietf-emailcore-rfc5322bis/>. Note to RFC Editor and WG: This reference, and citations to it (including mentions of "RFC 5322bis", "RFC5322bis", and other variations) should be updated to the correct RFC number and other information when work on rfc5321bis (this document) and rfc5322bis is complete. All references to the original RFC 5322 has been removed from this specification. Once that update has been accomplished, this note should be removed.

10.2. Informative References

[17] Crocker, D., "STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES", STD 11, RFC 822, DOI 10.17487/RFC0822, August 1982, <https://www.rfc-editor.org/info/rfc822>.

[18] Butler, M., Postel, J., Chase, D., Goldberger, J., and J. Reynolds, "Post Office Protocol: Version 2", RFC 937, DOI 10.17487/RFC0937, February 1985, <https://www.rfc-editor.org/info/rfc937>.

[19] Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 959, DOI 10.17487/RFC0959, October 1985, <https://www.rfc-editor.org/info/rfc959>.

[20] Partridge, C., "Mail routing and the domain system", STD 10, RFC 974, DOI 10.17487/RFC0974, January 1986, <https://www.rfc-editor.org/info/rfc974>.

[21] Partridge, C., "Duplicate messages and SMTP", RFC 1047, DOI 10.17487/RFC1047, February 1988, <https://www.rfc-editor.org/info/rfc1047>.

[22] Lambert, M., "PCMAIL: A distributed mail system for personal computers", RFC 1056, DOI 10.17487/RFC1056, June 1988, <https://www.rfc-editor.org/info/rfc1056>.

[23] Crispin, M., "Interactive Mail Access Protocol: Version 2", RFC 1176, DOI 10.17487/RFC1176, August 1990, <https://www.rfc-editor.org/info/rfc1176>.

[24] Klensin, J., Freed, N., Ed., Rose, M., Stefferud, E., and D. Crocker, "SMTP Service Extensions", RFC 1425, DOI 10.17487/RFC1425, February 1993, <https://www.rfc-editor.org/info/rfc1425>.

[25] Durand, A. and F. Dupont, "SMTP 521 Reply Code", RFC 1846, DOI 10.17487/RFC1846, September 1995, <https://www.rfc-editor.org/info/rfc1846>.

[26] Galvin, J., Murphy, S., Crocker, S., and N. Freed, "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted", RFC 1847, DOI 10.17487/RFC1847, October 1995, <https://www.rfc-editor.org/info/rfc1847>.

[27] Klensin, J., Freed, N., Rose, M., Stefferud, E., and D. Crocker, "SMTP Service Extensions", STD 10, RFC 1869, DOI 10.17487/RFC1869, November 1995, <https://www.rfc-editor.org/info/rfc1869>.

[28] Myers, J. and M. Rose, "Post Office Protocol - Version 3", STD 53, RFC 1939, DOI 10.17487/RFC1939, May 1996, <https://www.rfc-editor.org/info/rfc1939>.

[29] De Winter, J., "SMTP Service Extension for Remote Message Queue Starting", RFC 1985, DOI 10.17487/RFC1985, August 1996, <https://www.rfc-editor.org/info/rfc1985>.

[30] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <https://www.rfc-editor.org/info/rfc2045>.

[31] Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, DOI 10.17487/RFC2047, November 1996, <https://www.rfc-editor.org/info/rfc2047>.

[32] Kille, S., "MIXER (Mime Internet X.400 Enhanced Relay): Mapping between X.400 and RFC 822/MIME", RFC 2156, DOI 10.17487/RFC2156, January 1998, <https://www.rfc-editor.org/info/rfc2156>.

[33] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <https://www.rfc-editor.org/info/rfc2181>.

[34] Freed, N. and K. Moore, "MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations", RFC 2231, DOI 10.17487/RFC2231, November 1997, <https://www.rfc-editor.org/info/rfc2231>.

[35] Klensin, J., Ed., "Simple Mail Transfer Protocol", RFC 2821, DOI 10.17487/RFC2821, April 2001, <https://www.rfc-editor.org/info/rfc2821>.

[36] Freed, N., "SMTP Service Extension for Command Pipelining", STD 60, RFC 2920, DOI 10.17487/RFC2920, September 2000, <https://www.rfc-editor.org/info/rfc2920>.

[37] Freed, N., "Behavior of and Requirements for Internet Firewalls", RFC 2979, DOI 10.17487/RFC2979, October 2000, <https://www.rfc-editor.org/info/rfc2979>.

[38] Vaudreuil, G., "SMTP Service Extensions for Transmission of Large and Binary MIME Messages", RFC 3030, DOI 10.17487/RFC3030, December 2000, <https://www.rfc-editor.org/info/rfc3030>.

[39] Eastlake 3rd, D., Manros, C., and E. Raymond, "Etymology of "Foo"", RFC 3092, DOI 10.17487/RFC3092, April 2001, <https://www.rfc-editor.org/info/rfc3092>.

[40] Moore, K., "Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notifications (DSNs)", RFC 3461, DOI 10.17487/RFC3461, January 2003, <https://www.rfc-editor.org/info/rfc3461>.

[41] Moore, K. and G. Vaudreuil, "An Extensible Message Format for Delivery Status Notifications", RFC 3464, DOI 10.17487/RFC3464, January 2003, <https://www.rfc-editor.org/info/rfc3464>.

[42] Melnikov, A., Ed. and B. Leiba, Ed., "Internet Message Access Protocol (IMAP) - Version 4rev2", RFC 9051, DOI 10.17487/RFC9051, August 2021, <https://www.rfc-editor.org/info/rfc9051>.

[43] Klensin, J. and Y. Ko, "Overview and Framework for Internationalized Email", RFC 6530, DOI 10.17487/RFC6530, February 2012, <https://www.rfc-editor.org/info/rfc6530>.

[44] Yao, J. and W. Mao, "SMTP Extension for Internationalized Email", RFC 6531, DOI 10.17487/RFC6531, February 2012, <https://www.rfc-editor.org/info/rfc6531>.

[45] Hansen, T., Ed. and A. Melnikov, Ed., "Message Disposition Notification", STD 85, RFC 8098, DOI 10.17487/RFC8098, February 2017, <https://www.rfc-editor.org/info/rfc8098>.

[46] Schaad, J., Ramsdell, B., and S. Turner, "Secure/ Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/info/rfc8551>.

[47] Nakamura, M. and J. Hagino, "SMTP Operational Experience in Mixed IPv4/v6 Environments", RFC 3974, DOI 10.17487/RFC3974, January 2005, <https://www.rfc-editor.org/info/rfc3974>.

[48] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[49] Gellens, R. and J. Klensin, "Message Submission for Mail", STD 72, RFC 6409, DOI 10.17487/RFC6409, November 2011, <https://www.rfc-editor.org/info/rfc6409>.

[50] Wouters, P., Ed., Huigens, D., Winter, J., and Y. Niibe, "OpenPGP", RFC 9580, DOI 10.17487/RFC9580, July 2024, <https://www.rfc-editor.org/info/rfc9580>.

[51] Hansen, T. and J. Klensin, "A Registry for SMTP Enhanced Mail System Status Codes", BCP 138, RFC 5248, DOI 10.17487/RFC5248, June 2008, <https://www.rfc-editor.org/info/rfc5248>.

[52] Klensin, J., Freed, N., Rose, M., and D. Crocker, Ed., "SMTP Service Extension for 8-bit MIME Transport", STD 71, RFC 6152, DOI 10.17487/RFC6152, March 2011, <https://www.rfc-editor.org/info/rfc6152>.

[53] Klensin, J., "SMTP 521 and 556 Reply Codes", RFC 7504, DOI 10.17487/RFC7504, June 2015, <https://www.rfc-editor.org/info/rfc7504>.

[54] Levine, J. and M. Delany, "A "Null MX" No Service Resource Record for Domains That Accept No Mail", RFC 7505, DOI 10.17487/RFC7505, June 2015, <https://www.rfc-editor.org/info/rfc7505>.

[55] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <https://www.rfc-editor.org/info/rfc5321>.

[56] Klensin, J.C., Ed., Murchison, K., Ed., and E. Sam, Ed., "Applicability Statement for IETF Core Email Protocols", 6 August 2021, <https://datatracker.ietf.org/doc/draft-ietf-emailcore-as/>.

[57] Internet Assigned Number Authority (IANA), "Mail Parameters", 2022, <https://www.iana.org/assignments/mail-parameters>.

[58] Internet Assigned Number Authority (IANA), "IANA Mail Parameters: SMTP Service Extensions", 2022, <https://www.iana.org/assignments/mail-parameters/mail-parameters.xhtml#mail-parameters-2>.

[59] Internet Assigned Number Authority (IANA), "IANA Mail Parameters: Mail Transmission Types", 2022, <https://www.iana.org/assignments/mail-parameters/mail-parameters.xhtml#mail-parameters-5>.

[60] Internet Assigned Number Authority (IANA), "IANA Mail Parameters: Additional-registered-clauses", 2022, <https://www.iana.org/assignments/mail-parameters/mail-parameters.xhtml#mail-parameters-8>.

[61] Internet Assigned Number Authority (IANA), "Message Headers", 2022, <https://www.iana.org/assignments/message-headers/message-headers.xhtml>.

[62] Internet Assigned Number Authority (IANA), "Address Literal Tags", 2007, <http://www.iana.org/assignments/address-literal-tags>.

[63] RFC Editor, "RFC Errata - RFC 5321", 2019, <https://www.rfc-editor.org/errata/rfc5321>. (Captured 2019-11-19)

[64] IANA, "SMTP Service Extensions", 2021, <https://www.iana.org/assignments/mail-parameters/mail-parameters.xhtml#mail-parameters-2>. Notes in draft: RFC Editor: Please adjust date field to reflect whatever you want for a registry that is updated periodically. IANA: Please determine if the above URL is a sufficiently stable reference and adjust as appropriate if it is not.

[65] RFC Editor, "RFC Errata: RFC 5321", 2022, <https://www.rfc-editor.org/errata/rfc5321>. Captured 2024-06-20.

Appendix A. TCP Transport Service

The TCP connection supports the transmission of 8-bit bytes. The SMTP data is 7-bit ASCII characters. Each character is transmitted as an 8-bit byte with the high-order bit cleared to zero. Service extensions may modify this rule to permit transmission of full 8-bit data bytes as part of the message body, or, if specifically designed to do so, in SMTP commands or responses.

Appendix B. Generating SMTP Commands from Internet Message Format Header Fields

Under ideal circumstances, SMTP servers as specified in this document would receive complete information, including proper envelope information, either from prior SMTP clients or from Message Submission systems that conform to RFC 6409. SMTP servers that receive complete information would interact with the message body only by prepending trace information as discussed in Section 4.4 above.

On the other hand, there are systems in use that do not provide the complete information in the expected form. Some are "gateways" as described in Section 2.3.10 (possibly including the special case of firewalls) and Section 3.7. Some of those systems use an Internet Message Format [16] header section (or something similar without other information) as a substitute for a mail submission protocol that conforms to RFC 6409 [49] or otherwise require that SMTP-receivers make up commands from information in what SMTP considers the message body before such a message is transmitted to the next system by the corresponding SMTP-sender. This Appendix discusses some of the issues and appropriate actions when those situations are encountered.

Nothing in this appendix, or elsewhere in this specification, encourages or allows an SMTP-receiver to alter message headers that are compliant with Internet Message Format specifications before the message is passed on to another system by the corresponding SMTP-sender. When such messages are, or appear to be, compliant in that way, the message content is to be altered only by the addition, at the beginning of the content, of trace fields as specified in Section 4.4.

While direct communication between a MUA and MTA (rather than through a Submission Server [49]) is a private matter, not covered by any Internet Standard, there are problems with this approach. For example, there have been repeated problems with proper handling of "bcc" copies and redistribution lists when information that conceptually belongs to the mail envelope is not separated early in processing from header field information (and kept separate).

It is recommended that an MUA provide its initial ("submission client") MTA with an envelope separate from the message itself. However, if the envelope is not supplied, the envelope SHOULD be generated as follows:

1. Each recipient address from a TO, CC, or BCC header field SHOULD be copied to a RCPT command (generating multiple message copies if that is required for queuing or delivery). This includes any addresses listed in a RFC 822 "group". Any BCC header fields SHOULD then be removed from the header section.

2. The return address in the MAIL command SHOULD, if possible, be derived from the system's identity for the submitting (local) user, and the "From:" header field otherwise. If there is a system identity available, it SHOULD also be copied to the Sender header field if it is different from the address in the From header field. (Any Sender header field that was already there SHOULD be removed.) Systems may provide a way for submitters to override the envelope return address, but may want to restrict its use to privileged users. This will not prevent mail forgery, but may lessen its incidence; see Section 7.1.

When an MTA is being used in this way, it bears responsibility for ensuring that the message being transmitted is valid. The mechanisms for checking that validity, and for handling (or returning) messages that are not valid at the time of arrival, are part of the MUA-MTA interface and not covered by this specification.

A submission protocol based on Standard RFC 822 information alone MUST NOT be used to gateway a message from a foreign (non-SMTP) mail system into an SMTP environment. Additional information to construct an envelope must come from some source in the other environment, whether supplemental header fields or the foreign system's envelope.

Attempts to gateway messages using only their header "To" and "Cc" fields have repeatedly caused mail loops and other behavior adverse to the proper functioning of the Internet mail environment. These problems have been especially common when the message originates from an Internet mailing list and is distributed into the foreign environment using envelope information. When these messages are then processed by a header-section-only remailer, loops back to the Internet environment (and the mailing list) are almost inevitable.

Appendix C. Scenarios

This section presents complete scenarios of several types of SMTP sessions. In the examples, "C:" indicates what is said by the SMTP client, and "S:" indicates what is said by the SMTP server.

C.1. A Typical SMTP Transaction Scenario

This SMTP example shows mail sent by Smith at host bar.com, and to Jones, Green, and Brown at host foo.com. Here we assume that host bar.com contacts host foo.com directly. The mail is accepted for Jones and Brown. Green does not have a mailbox at host foo.com.

```
S: 220 foo.com Simple Mail Transfer Service Ready
C: EHLO bar.com
S: 250-foo.com greets bar.com
S: 250-8BITMIME
S: 250-SIZE
S: 250-DSN
S: 250 HELP
C: MAIL FROM:<Smith@bar.com>
S: 250 OK
C: RCPT TO:<Jones@foo.com>
S: 250 OK
C: RCPT TO:<Green@foo.com>
S: 550 No such user here
C: RCPT TO:<Brown@foo.com>
S: 250 OK
C: DATA
S: 354 Start mail input; end with <CRLF>.<CRLF>
C: Blah blah blah...
C: ...etc. etc. etc.
C: .
S: 250 OK
C: QUIT
S: 221 foo.com Service closing transmission channel
```

C.2. Aborted SMTP Transaction Scenario

```
S: 220 foo.com Simple Mail Transfer Service Ready
C: EHLO bar.com
S: 250-foo.com greets bar.com
S: 250-8BITMIME
S: 250-SIZE
S: 250-DSN
S: 250 HELP
C: MAIL FROM:<Smith@bar.com>
S: 250 OK
C: RCPT TO:<Jones@foo.com>
S: 250 OK
C: RCPT TO:<Green@foo.com>
S: 550 No such user here
C: RSET
S: 250 OK
C: QUIT
S: 221 foo.com Service closing transmission channel
```

C.3. Relayed Mail Scenario

Step 1 -- Source Host to Relay Host
The source host performs a DNS lookup on XYZ.COM (the destination address) and finds DNS MX records specifying xyz.com as the best preference and foo.com as a lower preference. It attempts to open a connection to xyz.com and fails. It then opens a connection to foo.com, with the following dialogue:

```
S: 220 foo.com Simple Mail Transfer Service Ready
C: EHLO bar.com
S: 250-foo.com greets bar.com
S: 250-8BITMIME
S: 250-SIZE
S: 250-DSN
S: 250 HELP
C: MAIL FROM:<JQP@bar.com>
S: 250 OK
C: RCPT TO:<Jones@XYZ.COM>
S: 250 OK
C: DATA
S: 354 Start mail input; end with <CRLF>.<CRLF>
C: Date: Thu, 21 May 1998 05:33:29 -0700
C: From: John Q. Public <JQP@bar.com>
C: Subject:  The Next Meeting of the Board
C: To: Jones@xyz.com
C:
C: Bill:
C: The next meeting of the board of directors will be
C: on Tuesday.
C:                          John.
C: .
S: 250 OK
C: QUIT
S: 221 foo.com Service closing transmission channel
```

Step 2 -- Relay Host to Destination Host
foo.com, having received the message, now does a DNS lookup on xyz.com. It finds the same set of MX records, but cannot use the one that points to itself (or to any other host as a worse preference). It tries to open a connection to xyz.com itself and succeeds. Then we have:

```
S: 220 xyz.com Simple Mail Transfer Service Ready
C: EHLO foo.com
S: 250 xyz.com is on the air
C: MAIL FROM:<JQP@bar.com>
S: 250 OK
C: RCPT TO:<Jones@XYZ.COM>
S: 250 OK
C: DATA
S: 354 Start mail input; end with <CRLF>.<CRLF>
C: Received: from bar.com by foo.com ; Thu, 21 May 1998
C:     05:33:29 -0700
C: Date: Thu, 21 May 1998 05:33:29 -0700
C: From: John Q. Public <JQP@bar.com>
C: Subject:  The Next Meeting of the Board
C: To: Jones@xyz.com
C:
C: Bill:
C: The next meeting of the board of directors will be
C: on Tuesday.
C:                          John.
C: .
S: 250 OK
C: QUIT
S: 221 xyz.com Service closing transmission channel
```

C.4. Verifying and Sending Scenario

```
S: 220 foo.com Simple Mail Transfer Service Ready
C: EHLO bar.com
S: 250-foo.com greets bar.com
S: 250-8BITMIME
S: 250-SIZE
S: 250-DSN
S: 250-VRFY
S: 250 HELP
C: VRFY Crispin
S: 250 Mark Crispin <Admin.MRC@foo.com>
C: MAIL FROM:<EAK@bar.com>
S: 250 OK
C: RCPT TO:<Admin.MRC@foo.com>
S: 250 OK
C: DATA
S: 354 Start mail input; end with <CRLF>.<CRLF>
C: Blah blah blah...
C: ...etc. etc. etc.
C: .
S: 250 OK
C: QUIT
S: 221 foo.com Service closing transmission channel
```

Appendix D. Deprecated Features of RFC 821

A few features of RFC 821 have proven to be problematic and SHOULD NOT be used in Internet mail. Some of these features were deprecated in RFC 2821 in 2001; source routing and two-digit years in dates were deprecated even earlier, by RFC 1123 in 1989. Of the domain literal forms, RFC 1123 required support only for the dotted decimal form. With the possible exception of old, hardware-embedded, applications, there is no longer any excuse for these features to appear on the contemporary Internet.

D.1. TURN

This command, described in RFC 821, raises important security issues since, in the absence of strong authentication of the host requesting that the client and server switch roles, it can easily be used to divert mail from its correct destination. Its use is deprecated; SMTP systems SHOULD NOT use it unless the server can authenticate the client. For most of the cases in which TURN might be useful, the ETRN extension [29], mentioned in Section 4.5.4.1, may be a better choice.

D.2. Source Routing

RFC 821 utilized the concept of explicit source routing to get mail from one host to another via a series of relays. Source routes could appear in either the <forward-path> or <reverse-path> to show the hosts through which mail would be routed to reach the destination. The requirement to utilize source routes in regular mail traffic was eliminated by the introduction of the domain name system "MX" record by RFC 974 in early 1986 and the last significant justification for them was eliminated by the introduction, in RFC 1123, of a clear requirement that addresses following an "@" must all be fully-qualified domain names. Issues involving local aliases for mailboxes were addressed by the introduction of a separate specification for mail submission [49]. Consequently, there are no remaining justifications for the use of source routes other than support for very old SMTP clients. Even use in mail system debugging is unlikely to work because almost all contemporary systems either ignore or reject them.

Historically, for relay purposes, the forward-path may have been a source route of the form "@ONE,@TWO:JOE@THREE", where ONE, TWO, and THREE MUST be fully-qualified domain names. This form was used to emphasize the distinction between an address and a route. The mailbox (here, JOE@THREE) is an absolute address, and the route is information about how to get there. The two concepts should not be confused.

SMTP servers MAY continue to accept source route syntax as specified in this appendix. If they do so, they SHOULD ignore the routes and utilize only the target domain in the address. If they do utilize the source route, the message MUST be sent to the first domain shown in the address.

In particular, a server MUST NOT guess at shortcuts within the source route.

SMTP clients MUST NOT attempt to utilize explicit source routing.

If source routes appear in mail received by an SMTP server contrary to the requirements and recommendations in this specification, RFC 821 and the text below should be consulted for the mechanisms for constructing and updating the forward-path. A server that is reached by means of a source route (e.g., its domain name appears first in the list in the forward-path) MUST remove its domain name from any forward-paths in which that domain name appears before forwarding the message and MAY remove all other source routing information. Any source route information in the reverse-path SHOULD be removed by servers conforming to this specification.

The following information is provided for historical information so that the source route syntax and application can be understood if needed.

Syntax:
The original form of the <Path> production in Section 4.1.2 was:

```
Path  = "<" [ A-d-l ":" ] Mailbox ">"

A-d-l  = At-domain *( "," At-domain )

At-domain  = "@" Domain
```

For example, suppose that a delivery service notification must be sent for a message that arrived with:
MAIL FROM:<@a.example,@b.example:user@d.example>
The notification message MUST be sent using:
RCPT TO:<user@d.example>

D.3. HELO

As discussed in Sections 3.1 and 4.1.1, EHLO SHOULD be used rather than HELO when the server will accept the former. Servers MUST continue to accept and process HELO in order to support older clients.

D.4. #-literals

RFC 821 provided for specifying an Internet address as a decimal integer host number prefixed by a pound sign, "#". In practice, that form has been obsolete since the introduction of TCP/IP. It is deprecated and MUST NOT be used.

D.5. Dates and Years

When dates are inserted into messages by SMTP clients or servers (e.g., in trace header fields), four-digit years MUST BE used. Two-digit years are deprecated; three-digit years were never permitted in the Internet mail system.

D.6. Sending versus Mailing

In addition to specifying a mechanism for delivering messages to user mailboxes, RFC 821 provided additional, optional, commands to deliver messages directly to the user's terminal screen. These commands (SEND, SAML, SOML) were rarely implemented, and changes in workstation technology and the introduction of other protocols may have rendered them obsolete even where they are implemented.

Clients SHOULD NOT use SEND, SAML, or SOML commands. If a server implements them, the implementation model specified in RFC 821 [6] MUST be used and the command names MUST be published in the response to the EHLO command.

Appendix E. Summary of changes from RFC 5321 (published in October 2008) to <<This Document>>

As discussed in Section 1.2, this specification combines material from several earlier ones. The most numerous changes from RFC 5321 have been editorial in nature. Those changes have included correcting long-standing ambiguities and errors, improving terminology and its consistent use, updating references to documents that have been replaced, adding additional cross-references within the document, and reorganizing material to make it easier to follow. In general, those changes are not called out in the list below. The order of changes in the list below is not significant.

E.1. General Change Listing

// RFC Editor: The list that follows was numbered in order to make
// review discussion convenient. Unless you prefer it, I'd rather
// have a bullet list in the final version to reinforce the "order
// not significant" message.

1. All of the outstanding errata [65] filed against RFC 5321 have been addressed. That list does include some editorial issues.

2. The discussion of SMTP Service Extensions and how they are registered with IANA has been extensively revised and a new registration model defined. The reasons for this are discussed in Section 2.2.2.

3. Corrected, updated, or clarified a few ABNF syntax errors.

4. Improved the descriptions of the applicability of several reply codes. Also included descriptions of codes added since RFC 5321 was published.

5. Removed the former Appendix C that described Source Routes. Information about them that is still relevant appears in Appendix D.2.

6. An index was added to make it easier for readers to find specific terminology, ABNF productions, command arguments, and so on. Several additional cross-references have been added for the same reasons.

7. Clarified the relationship between mail transactions, repeated uses of EHLO within an SMTP session, and command arguments and responses between transactions.

8. Improved the discussion of the distinction of Message Submission Agents (MSAs), particularly those described in RFC 6409 [49], and Mail Transfer Agents (MTAs) as exemplified by this specification. This document does not alter RFC 6409 in any way.

9. // 20241108: This bullet item added per Ticket #107 and WG discussion
// this date.

// While I believe the following reflects the decision of the WG at
// the meeting, it does put us into an odd place. We are saying that
// the text has been reworked (true) but the note may not belong in
// this section if those changes are not substantive (I think that
// they are). If they are, we should be saying more about what
// changed.
The discussion of "trace information" has been reworked to make it more clear and more consistent with the discussion in the Message Format specification [16]. While the textual changes are extensive, it is not believed that any of them make substantive changes to the SMTP definition.

10. In Section 4.1.1.1 a SHOULD NOT prohibition on extra text following an argument to EHLO or HELO was changed to a MUST NOT. While this change makes some previously conforming implementations (ones that followed the advice of RFC 2821) non-conforming, there are not believed to be any contemporary sender implementations that send that text. Requirements and recommendations for SMTP server implementations are unchanged.

11. At the request of IANA, the IANA Considerations Section (Section 8) and some related material have been extensively rewritten to provide more detailed specifications for registry contents and organization.

E.2. Disposition of Errata Filed Against RFC 5321

This document addresses the following errata filed against RFC 5321 since its publication in October 2008. More details on each of these can be found in the RFC Editor's Errata listing for RFC 5321 [63].

1543 Wrong code in description. Section 3.8

1683 ABNF error in Additional-Registered-Clauses. Section 4.4

1820 Wrong description of a mailing list function. Section 3.4.2.2
Note that the identified issue was correct, but a different fix was used.

1851 Location of text on unexpected close Section 4.1.1.5 Text moved to Section 3.8.

2578 Incorrect section reference to core ABNF rules defined in RFC 5234, Section 4.1.2. [15]

3447 Use of normative language (e.g., more "MUST"s); possible confusion in Section 4.4. WG was not able to agree that the claimed confusion was substantive or that change was appropriate.

4198 Description error of which entity tests SMTP reply codes. Section 4.2.

4315 Updated IPv6-addr ABNF. Section 4.1.3.
Note that this is a revision of erratum 2467.

5414 Updated ABNF for Quoted-string to disallow empty string. Section 4.1.2

4055 Description of SPF and DKIM is wrong. Resolved by dropping the sentence from Section 3.6.1.

5711 Missing leading spaces in example in Appendix C.3.

The following errata were considered by the WG but no changes were made to the document: 4079, 4265, 6561

Appendix F. Summary of changes made after draft-ietf-emailcore-rfc5321bis-29 (posted 2024-05-23)

// RFC Editor: Please remove this appendix.

This appendix covers only those changes since draft -29 (the "cleaned-up" version with detailed change log and ticket information removed). For earlier changes and related information, see <draft-ietf-emailcore-rfc5321bis-28>

F.1. Summary of changes from draft-ietf-emailcore-rfc5321bis-29 (posted 2024-05-23) to -30

1. Added the listing of errata and corrections back in per discussion of the errata listing in rfc5322bis and instructions from a co-chair. The list was edited to put the reports in numerical order and eliminate references to ticket numbers, then significantly rewritten by Alexey Melnikov and his version incorporated with minor adjustments.

2. After an extended discussion on the mailing list about the issues with erratum 3447 (see Appendix E.2), the co-chair concluded that there was no consensus for change. See <https://mailarchive.ietf.org/arch/msg/emailcore/cvG47YOfQwc8fj-E9-jZCzK5Uro>.

3. Small editorial and error corrections.

F.2. Summary of changes from draft-ietf-emailcore-rfc5321bis-30 (posted 2024-07-05) to -31

draft-ietf-emailcore-rfc5321bis-31 is the version put out for IETF Last Call on 2024-09-26 with the Last Call extending to 2024-10-09.

1. Editorial/typographical corrections.

2. Multiple small corrections of glitches caught by Murray in pre-Last-Call review. Thanks to him and too Alexey for going over the proposed fixes/ dispositions.

F.3. Summary of changes from draft-ietf-emailcore-rfc5321bis-31 (posted 2024-09-09) to -32

Version -32 of this draft was prepared based on IETF Last Call review comments (a collection that turned out to be incomplete when it was posted) to aid the Working Group in considering responses to those comments. In some cases, responses have been tentatively integrated. In others comments have been inserted to act as descriptive placeholders. Some editorial corrections and improvements have been applied as usual.

This version is a special update to assist the WG in processing review comments from IETF Last Call. Everything described in this section that changed the document from -31 is tentative pending WG review and consensus approval.

1. Editorial/typographical corrections, starting with a typo reported by Paul Kyzivat 2024-09-29.

2. Removed note about updates to RFC 8126 after correspondence from IANA and clarity that such an update would not occur, nor would draft-klensin-iana-consid-hybrid be processed, before IETF Last Call ends on 2024-10-10.

3. Per a problem identified in Donald Eastlake's review, added a note to Section 8.3, item (9) about MT-PRIORITY and RFC 6710 and requires require WG action. A similar note was added to Section 4.2.2 which requires either a WG decision to not do that or very careful checking to insure the groups are correct.

4. Note added to Section 1.2 to explicitly remove "Internet Standard" designation from RFC 821. See the note there and discussion on mailing list as to whether it should be here or resolved elsewhere.

5. A few other CREF notes were added to flag outstanding Last Call issues. Existing ticket numbers that were readily at hand have been included in the notes; it does not seem likely that the Chairs will assign tickets to new issues before the IETF 121 schedule requires posting of this draft.

F.4. Summary of changes from draft-ietf-emailcore-rfc5321bis-32 (posted 2024-10-19) to -33

Like -32 before it (see immediately above), -33 is a temporary working draft. It was posted at the beginning of IETF 121 to incorporate additional comments and suggestions after -32 was posted and to facilitate WG discussion during that IETF meeting, presumably leading to another draft for IESG review or an additional IETF Last Call.

1. Rewrote the introductory Note (just below the abstract) to reflect the above. Some of the text is deliberately redundant.

2. Updated an obsolete reference for OpenPGP.

3. Improved terminology by eliminating all instances of "response code" in favor of "reply code"

4. Fixes per Alexey's 2024-10-21 messages

5. Inserted CREF notes for all of the tickets created between the start of IETF Last Call on 2024-09-26 and the time of posting on 2024-11-02. Annotated a few comments created for draft -32 with the ticket numbers.

F.5. Summary of changes from draft-ietf-emailcore-rfc5321bis-33 (posted 2024-11-02) to -34

1. Incorporate changes agreed to, and suggested text from, WG meeting on 2024-11-07 as an aid to the meeting scheduled for the next day.
   See <https://github.com/ietf-wg-emailcore/emailcore/issues> and <https://notes.ietf.org/notes-ietf-121-emailcore> for additional context.

2. Incorporate draft new text to explain the focus of (original) SMTP and the importance and existence of extensions and other tools to mitigate difficulties with that focus on the modern Internet. Point explicitly to the A/S for discussion of those options and additional security considerations. See Section 1.3 and the new introduction to Section 7 for the text.

F.6. Summary of changes from draft-ietf-emailcore-rfc5321bis-34 (posted 2024-11-07) to -35

1. Rewrote the introductory Note again.

2. Added text from Bron and Todd to revise Section 6.1, including including some temporary associated notes. Also added Dave's alternate suggestion and pointers to the discussion.

3. Moved former Appendix D into the main body of the text as new Section 3.7.6. Note that this causes all subsequent appendices to be renumbered.

4. Changed the SHOULD NOT to a MUST NOT in Section 4.1.1 per ticket #107 and accordingly added a new bullet point to Appendix E.1.

5. Per advice from Alexey, rearranged the "Function Groups" for reply codes a bit.

6. Additional minor editorial changes and changes to reflect discussion at WG meeting on 2024-11-08.

F.7. Summary of changes from draft-ietf-emailcore-rfc5321bis-35 (posted 2024-11-11) to -36

1. Small editorial changes to reflect Vijay Gurbani Last Call review, <https://mailarchive.ietf.org/arch/msg/emailcore/kB1CNtf0-hY4mUlFnBD5_jp2uSE> and other discoveries.

2. Reorganized Section 4.2.2 to reflect the original (RFC 821) "second digit" structure and improved the explanation of that arrangement.

3. One case where "must" appeared in lower case but upper case was correct was changed. If finding it is important, search for lower-case "fred".

4. Added text to Section 3.3 and Section 4.1.1.5 to clarify that HELO, not just EHLO, terminates mail transactions.

5. Improved slightly on the TURN description in Appendix D.1.

6. Annotations ("CREF comments") added to reflect additional open tickets; some earlier ones adjusted.

F.8. Summary of changes from draft-ietf-emailcore-rfc5321bis-36 (posted 2024-12-02) to -37

Changes discussed during 2024-12-03 interim meeting.

1. Reordered former fifth and sixth paragraphs of Section 4.1.1.4 (Ticket #125).

2. The sentence about "other contemporary standards" and surrounding text in Section 2.3.1 were rewritten to improve clarity (Ticket #124).

3. Rewrote the introductory paragraph of Section 7 to more closely align it with discussion before, during, and after the interim meeting (Ticket #109 and others).

4. Added very temporary Appendix G to note issues spotted during or after IETF Last Call to avoid spending more review time on them.

F.9. Summary of changes from draft-ietf-emailcore-rfc5321bis-37 (posted 2024-12-07) to -38

1. Revised IANA Considerations (Section 8) to reflect notes from IANA, [IANA #1382893], 2024-12-17 and 2024-12-21 and addressing Ticket #120 and removing former CREF note at the top of the Section. A new CREF note has been introduced to explain the changes.

2. The separate bullet item in Section 8 for "The "SMTP PRIORITY extension Priority Assignment Policy", including the reference to Ticket #103, has been removed on the assumption that it will have been sorted out as a separate administrative action before this document is published.

3. The former Section 8.2 has been moved to the end of Section 8 and rewritten to summarize specific advice/instructions to IANA.

4. Small editorial changes/ fixes.

F.10. Summary of changes from draft-ietf-emailcore-rfc5321bis-38 (posted 2025-01-03) to -39

1. Changed pointer to IPv6 syntax to point to RFC 5952 rather than 4291

2. Removed tracking and related CREF comments, i.e., all such comments other than those addressed to or involving the RFC Editor/RPC or IANA. The latter should be temporary pending resolution of IANA issues and will then be removed. Unless errors have been made, the remaining ones should all be identified with "RFC Editor".

3. Undid some of that cleanup by adding new notes for the convenience of the WG at the interim and surrounding days. If only because the IESG has made it clear that they don’t want to see notes that imply decisions have not yet been made in the WG, those additional notes will not survive into -40.

4. Rewrote what is now the "historical note" at the beginning of Section 8.3.2 about IANA Considerations changes. The IESG was not interested in getting involved with this issue.

5. Aggressively trimmed Section 1, Paragraph 2, removing the comments about the intermediate versions -32 through -36. Added a note about -39 and the interim. That entire note will still disappear before RFC publication.

6. Usual small editorial fixes, this time including correcting a reference and removing an unused one.

Appendix G. Notes to RFC Editor / RPC

Apparently due to glitches in xml2rfc, some index entries embedded in the text with <iref> do not appear in the index. I am guessing the problem may be two <iref>s in the same block of text with the same item but different subitems, but could be wrong. That should be checked and corrected. Also, there is at least one place where the rendering machinery (at least for plaintext) inserts several extra spaces in the middle of a line for no apparent reason.

Obviously this section should be removed once those problems are remedied.

Index

C

Author's Address

John C. Klensin
1770 Massachusetts Ave, Suite 322
Cambridge, MA 02140
United States of America
Email: john-ietf@jck.com