Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 1, 2020

J. Klensin, Ed.

March 30, 2020

Applicability Statement for IETF Core Email Protocols
draft-klensin-email-core-as-00

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.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 1, 2020.

Table of Contents

1. Introduction

In its current form, this draft is a placeholder and beginning of an outline for the Applicability Statement that has been discussed as a complement for proposed revisions of the base protocol specifications for SMTP [RFC5321] (being revised as ID.RFC5321bis [ID.RFC5321bis]) and Internet Message Format [RFC5322] (being revised as ID.RFC5322bis [ID.RFC5322bis]). Among other things, it is expected to capture topics that a potential WG concludes are important but that should not become part of those core documents.

As discussed in RFC 2026 [RFC2026],

> "An Applicability Statement specifies how, and under what circumstances, one or more TSs may be applied to support a particular Internet capability."

That form of a standards track document is appropriate because one of the roles of such a document is to explain the relationship among technical specification, describe how they are used together, and make statements about what is "required, recommended, or elective".

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 RFC 2119 [RFC2119] and RFC 8174 [RFC8174].

2. Applicability of Some SMTP Provisions

Over the years since RFC 5321 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.

[[CREF1: ... Actual content to be supplied after WG consideration.]]

3. Applicability of Message Format Provisions

Placeholder:
I am not sure what, if anything, goes here. If nothing does, we drop the section.

[[CREF2: ... Actual content to be supplied after WG consideration.]]

4. MIME and Its Implications

When the work leading to the original version of the MIME specification was completed in 1992 [RFC1341], the intention was that it be kept separate from the specification for basic mail headers in RFC 822 [RFC0822]. That plan was carried forward into RFC 822's successors, RFC 2822 [RFC2822] and RFC 5322 [RFC5322]. The decision to do so was different from the one made for SMTP, for which the core specification was changed to allow for the extension mechanism [RFC1425] which was then incorporated into RFC 5321 and its predecessor [RFC2821].

Various uses of MIME have become nearly ubiquitous in contemporary email while others may have fallen into disuse or been repurposed from the intent of their original design.

It may be appropriate to make some clear statements about the applicability of MIME and its features.

5. Other Stuff

It is fairly clear that there will be things that do not fit into the sections outlined above. As one example, if the IETF wants to say something specific about signatures over headers or what (non-trace) headers may reasonably be altered in transit, that may be more appropriate to other sections than to any of the three suggested above.

6. Acknowledgments

... To be supplied...
[[CREF3: But don’t forget to mention the discussions on the SMTP list of the reasons for this document in the last half of 2019.]]

7. 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.

8. Security Considerations

All drafts are required to have a security considerations section and this one eventually will.

... To be supplied ...

9. References

9.1. 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>.

9.2. Informative References

[ID.RFC5321bis]
Klensin, J., "Simple Mail Transfer Protocol", December 2019, <https://datatracker.ietf.org/doc/draft-klensin-rfc5321bis/>.

[ID.RFC5322bis]
Resnick, P., "Internet Message Format", December 2019, <https://datatracker.ietf.org/doc/draft-resnick-rfc5322bis/>.

[RFC0822] 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>.

[RFC1341] Borenstein, N. and N. Freed, "MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies", RFC 1341, DOI 10.17487/RFC1341, June 1992, <https://www.rfc-editor.org/info/rfc1341>.

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

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

[RFC2822] Resnick, P., Ed., "Internet Message Format", RFC 2822, DOI 10.17487/RFC2822, April 2001, <https://www.rfc-editor.org/info/rfc2822>.

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

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/info/rfc5322>.

Author's Address

John C Klensin (editor)
1770 Massachusetts Ave, Ste 322
Cambridge, MA 02140
USA

Phone: +1 617 245 1457
Email: john-ietf@jck.com

Network Working Group
Internet-Draft
Obsoletes: 5321, 1846, 7504 (if
approved)
Updates: 1123 (if approved)
Intended status: Standards Track
Expires: January 3, 2021

J. Klensin
July 2, 2020

Simple Mail Transfer Protocol
draft-klensin-rfc5321bis-03

Abstract

This document is a specification of the basic protocol for Internet electronic mail transport. It 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. Although SMTP was designed as a mail transport and delivery protocol, this specification also contains information that is important to its use as a "mail submission" protocol for "split-UA" (User Agent) mail reading systems and mobile environments. This document replaces the earlier version with the same title in RFC 5321.
[[CREF1: Note in Draft: Except for the last sentence, the above is unchanged from 5321 and may need adjusting in the light of RFC 6409 as an Internet Standard.]]

Note on Reading This Working Draft

This working draft is extensively annotated with information about changes made over the decade since RFC 5321 appeared, especially when those changes might be controversial or should get careful review. Anything marked in CREF comments with "[5321bis]" is current. In general, unless those are marked with "[[Note in Draft", in the contents of an "Editor's note", or are in the "Errata Summary" appendix (Appendix H.1, they are just notes on changes that have already been made and where those changes originated. Comments identified as "2821ter" arose after the Last Call on what became RFC5321, sometimes before AUTH48 on that document or a bit later. Those, of course, should still be reviewed. Surviving comments about rfc5321bis-00 followed by a letter indicate intermediate working versions of this draft and can be ignored unless the origin of changes is important. As one can tell from the dates (when they are given), this document has been periodically updated over a very long period of time.

This evolving draft should be discussed on the ietf-smtp@ietf.org list.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 3, 2021.

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 [3] 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, 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 [4], RFC 974 [15], 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 document is a specification of the basic protocol for the 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 [3],
- domain name system requirements and implications for mail transport from RFC 1035 [4] and RFC 974 [15],
- the clarifications and applicability statements in RFC 1123 [5],
- the new error codes added by RFC 1846 [19] and later by RFC 7504 [48], obsoleting both of those documents, and

- material drawn from the SMTP Extension mechanisms in RFC 1869 [21].

- Editorial and clarification changes to RFC 2821 [29] to bring that specification to Draft Standard.

It obsoletes RFC 821, RFC 974, RFC 1869, and RFC 2821 and updates RFC 1123 (replacing the mail transport materials of RFC 1123). However, RFC 821 specifies some features that were not in significant use in the Internet by the mid-1990s and (in appendices) some additional transport models. Those sections are omitted here in the interest of clarity and brevity; readers needing them should refer to RFC 821.

It also includes some additional material from RFC 1123 that required amplification. This material has been identified in multiple ways, mostly by tracking flaming on various lists and newsgroups and problems of unusual readings or interpretations that have appeared as the SMTP extensions have been deployed. Where this specification moves beyond consolidation and actually differs from earlier documents, it supersedes them technically as well as textually.

Although SMTP was designed as a mail transport and delivery protocol, this specification also contains information that is important to its use as a "mail submission" protocol, as recommended for Post Office Protocol (POP) (RFC 937 [13], RFC 1939 [22]) and IMAP (RFC 3501 [36]). In general, the separate mail submission protocol specified in RFC 4409 [42] is now preferred to direct use of SMTP; 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.

A companion document, RFC 5322 [11], discusses message header sections and bodies and specifies formats and structures for them.

1.3. 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 RFC 2119 [1]. 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.

Because this document has a long history and to avoid the risk of various errors and of confusing readers and documents that point to

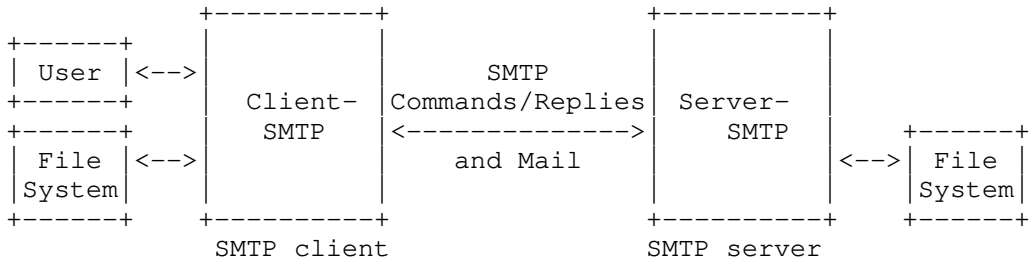
this one, most examples and the domain names they contain are preserved from RFC 2821. Readers are cautioned that these are illustrative examples that should not actually be used in either code or configuration files.

2. The SMTP Model

[[CREF2: [5321bis] [[Editor's Note: There have been extensive and repeated discussions on the SMTP and IETF lists about whether this document should say something about hop-by-hop (MTA-to-MTA) SMTP authentication and, if so, what?? Note that end to end message authentication is almost certainly out of scope for SMTP.]]]]

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, 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 [13], RFC 1939 [22]) or IMAP (RFC 3501 [36]) 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 4409 [42]) 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 [30]).

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 suggested 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, 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 4409 [42]. 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. Explicit "source" routing (see Section 5 and Appendix C and Appendix F.2) SHOULD NOT be used. [[CREF3: [5321bis] JcK 20090123 - redundant sentence removed.]]

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. 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 [32]).

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 and every extension, regardless of its benefits, must be carefully scrutinized 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

The IANA maintains a registry of SMTP service extensions. A corresponding EHLO keyword value is associated with each extension. Each service extension registered with the IANA must be defined in a formal Standards-Track or IESG-approved Experimental protocol document. The definition must include:

- the textual name of the SMTP service extension;
- the EHLO keyword value associated with the extension;
- the syntax and possible values of parameters associated with the EHLO keyword value;
- any additional SMTP verbs associated with the extension (additional verbs will usually be, but are not required to be, the same as the EHLO keyword value);
- any new parameters the extension associates with the MAIL or RCPT verbs;

- o a description of how support for the extension affects the behavior of a server and client SMTP; and

- o the increment by which the extension is increasing the maximum length of the commands MAIL and/or RCPT, over that specified in this Standard.

In addition, any EHLO keyword value starting with an upper or lower case "X" refers to a local SMTP service extension used exclusively through bilateral agreement. Keywords beginning with "X" MUST NOT be used in a registered service extension. Conversely, keyword values presented in the EHLO response that do not begin with "X" MUST correspond to a Standard, Standards-Track, or IESG-approved Experimental SMTP service extension registered with IANA. A conforming server MUST NOT offer non-"X"-prefixed keyword values that are not described in a registered extension.

Additional verbs and parameter names are bound by the same rules as EHLO keywords; specifically, verbs beginning with "X" are local extensions that may not be registered or standardized. Conversely, verbs not beginning with "X" must always be registered.

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 F and Appendix F.6).

The SMTP content is sent in the SMTP DATA protocol unit and has two parts: the header section and the body. If the content conforms to other contemporary standards, the header section consists of a collection of header fields, each consisting of a header name, a colon, and data, structured as in the message format specification (RFC 5322 [11]); the body, if structured, is defined according to MIME (RFC 2045 [24]). The content is textual in nature, expressed using the US-ASCII repertoire [2]. Although SMTP extensions (such as "8BITMIME", RFC 6152 [47]) may relax this restriction for the content body, the content header fields are always encoded using the US-ASCII repertoire. Two MIME extensions (RFC 2047 [25] and RFC 2231 [28]) define an algorithm for representing header values outside the US-ASCII repertoire, while still encoding them using the US-ASCII 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; 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 appear in SMTP transactions on the public Internet, particularly important where top-level domains are involved. These components ("labels" in DNS terminology, RFC 1035 [4]) are restricted for SMTP purposes to consist of a sequence of letters, digits, and hyphens drawn from the ASCII character set [2] and conforming to what RFC 1035 Section 2.3.1 calls the "preferred name syntax". Domain names are used as names of hosts and of other entities in the domain name hierarchy. For example, a domain may refer to an 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 [4] and Section 5 of this specification.

The domain name, as described in this document and in RFC 1035 [4], is the entire, fully-qualified name (often referred to as an "FQDN"). A domain name that is not in FQDN form is no more than a local alias. Local aliases MUST NOT appear in any SMTP transaction.

Only resolvable, fully-qualified domain names (FQDNs) are permitted when domain names are used in SMTP.
[[CREF4: [[5321bis Editor's Note: does "in the public DNS" or equivalent need to be added to "resolvable"???]]]]

In other words, names that can be resolved to MX RRs or address (i.e., A or AAAA) RRs (as discussed in Section 5) are permitted, as are CNAME RRs whose targets can be resolved, in turn, to MX or address RRs.
[[CREF5: [[5321bis Editor's Note: it is not clear whether "In other words" really meant "for example" or it is was intended that the only labels permitted are those that own records in one of the above RR types]]]]
[[CREF6: [[5321bis Editor's Note: More generally, does this section need work to clarify the relationship to private domain names (discussed on SMTP list starting 2013-03-26)]]]]
Local nicknames or unqualified names MUST NOT be used. 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".

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 [34], specifies further structuring of the reply strings, including the use of supplemental and more specific completion codes (see also RFC 5248 [46]).

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. 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. The MIME specification (RFC 2045 [24]) 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, 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 [31]). [[CREF7: [5321bis] [[Note in draft/Placeholder: There has been a request to expand this section, possibly into a more extensive model of Internet mail. Comments from others solicited. In particular, does RFC 5598 make that suggestion OBE?]]]]

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.

2.4. 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) 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 [2]. 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 [47]. 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 in other than 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 [10]. 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 D.

3.1. Session Initiation

An SMTP 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 554 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 554 response on connection opening SHOULD provide enough information in the reply text to facilitate debugging of the sending system.

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 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 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.) 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 the transaction.

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, contemporary systems SHOULD NOT use source routing (see Appendix C).

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 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).

The <forward-path> can contain more than just a mailbox. Historically, the <forward-path> was permitted to contain a source routing list of hosts and the destination mailbox; however, contemporary SMTP clients SHOULD NOT utilize source routes (see Appendix C). Servers MUST be prepared to encounter a list of source routes in the forward-path, but they SHOULD ignore the routes or MAY decline to support the relaying they imply. Similarly, servers MAY decline to accept mail that is destined for other hosts or systems. These restrictions make a server useless as a relay for clients that do not support full SMTP functionality. Consequently, restricted-capability 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). [[CREF8: [5321bis] JcK Note for 2821ter (5321bis): this section would be improved by being more specific about where mail transactions begin and end and then talking about "transaction state" here, rather than specific prior commands. --JcK]]

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. SMTP indicates the end of the mail data by sending a line containing only a "." (period or full stop). 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 ([12], [11]) 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 [24]) 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. 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.

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 either of the following forms:

```
User Name <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 [34], 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 in 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 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 [5], 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. In that case, 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 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. Source Routes and Relaying

In general, the availability of Mail eXchanger records in the domain name system (RFC 1035 [4], RFC 974 [15]) makes the use of explicit source routes in the Internet mail system unnecessary. Many historical problems with the interpretation of explicit source routes have made their use undesirable. SMTP clients SHOULD NOT generate explicit source routes except under unusual circumstances. SMTP servers MAY decline to act as mail relays or to accept addresses that specify source routes. When route information is encountered, SMTP servers MAY ignore the route information and simply send to the final destination specified as the last element in the route and SHOULD do so. There has been an invalid practice of using names that do not appear in the DNS as destination names, with the senders counting on

the intermediate hosts specified in source routing to resolve any problems. If source routes are stripped, this practice will cause failures. This is one of several reasons why SMTP clients MUST NOT generate invalid source routes or depend on serial resolution of names in such routes. [[CREF9: [5321bis] Jck 20091023: "of names..." added for clarity"]]

When source routes are not used, the process described in RFC 821 for constructing a reverse-path from the forward-path is not applicable and the reverse-path at the time of delivery will simply be the address that appeared in the MAIL command.

3.6.2. 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 for use in delivery notifications. Recent work, such as that on SPF [41] and DKIM [43] [44], has been done to provide ways to ascertain that an address is valid or belongs to the person who actually sent the message.
[[5321bis Editor's Note: Proposed erratum (4055) suggests that DKIM and SPF have nothing to do with this and that everything after the first sentence should be dropped. An alternative would be to tune the texts. ???]]
A server MAY attempt to verify the return path before using its address for delivery notifications, but methods of doing so are not defined here nor is any particular method recommended at this time.

3.6.3. Message Submission Servers 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 4409 [42]). In any event, because these arrangements are private and fall outside the scope of this specification, they are not described here.

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 [33] and RFC 3464 [35]) 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) 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).

3.7. 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, 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 RFC 5322 [11]). 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 RFC 5322 [11], 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.

3.8. 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.

3.9. Mailing Lists and Aliases

[[CREF10: [5321bis] If "alias and list models" are explained elsewhere, cross reference". Also note that this section appears to prohibit an exploder from adding List-* headers. That needs to be finessed.]]
An SMTP-capable host SHOULD support both the alias and the list models of address expansion for multiple delivery. 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. However, in this case, the message header section (RFC 5322 [11]) MUST be left unchanged; in particular, the "From" field of the header section is unaffected.

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.

3.9.1. Alias

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.

3.9.2. List

A mailing list may be said to operate by "redistribution" rather than by "forwarding". 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 aliases (Section 3.9.1) and forwarding (this subsection) is the change to the backward-pointing address in this case. When a list constrains its processing to the very limited set of modifications and actions described here, it is attempting to emulate an MTA; such lists can be treated as a continuation in email transit.

There exist mailing lists that perform additional, sometimes extensive, modifications to a message and its envelope. Such mailing lists need to be viewed as full MUAs, which accept a delivery and post a new message.

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).

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 SHOULD 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 [10], 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 excepting only private-use commands as described in Section 4.1.5. Private-use commands MAY be listed.

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 C). 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 normally consists of the required destination mailbox. Sending systems SHOULD NOT generate the optional list of hosts known as a source route. Receiving systems MUST recognize source route syntax but SHOULD strip off the source route specification and utilize the domain name associated with the mailbox as if the source route had not been provided.

Similarly, relay hosts SHOULD strip or ignore source routes, and names MUST NOT be copied into the reverse-path. When mail reaches its ultimate destination (the forward-path contains only a destination mailbox), the SMTP server inserts it into the destination mailbox in accordance with its host mail conventions.

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:<@hosta.int,@jkl.org:userc@d.bar.org>
```

will normally be sent directly on to host d.bar.org with envelope commands

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

As provided in Appendix C, xyz.com MAY also choose to relay the message to hosta.int, using the envelope commands

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

or to jkl.org, using the envelope commands

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

Attempting to use relaying this way is now strongly discouraged. 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 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 2.3.7) following the command as mail data from the sender. This command causes the mail data to be appended to the mail data buffer. The mail data may contain any of the 128 ASCII character codes, although experience has indicated that use of 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 5322 [11] 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. 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, as discussed in Section 4.4.

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" or "Received" line) 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, before EHLO 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.

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.

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 FROM or RCPT TO 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 FROM or RCPT TO 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 parameter's defining RFC.

4.1.2. Command Argument Syntax

The syntax of the argument clauses of the above commands (using the syntax specified in RFC 5234 [10] where applicable) is given below. Some of the productions given below are used only in conjunction with source routes as described in Appendix C. Some terminals not defined in this document, but are defined elsewhere, specifically:

In the "core" syntax in Appendix B of RFC 5234 [10]: ALPHA , CRLF , DIGIT , HEXDIG , and SP

In the message format syntax in RFC 5322 [11]: atext , CFWS , and FWS .

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

Forward-path   = Path

Path           = "<" [ A-d-l ":" ] Mailbox ">"

A-d-l          = At-domain *( "," At-domain )
                 ; Note that this form, the so-called "source
                 ; route", MUST BE accepted, SHOULD NOT be
                 ; generated, and SHOULD be ignored.

At-domain      = "@" Domain

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 [33]
                 ; 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 blackslash-quoting except
                  ; double-quote and the backslash itself.

   String         = Atom / Quoted-string
```

While the above definition for Local-part is relatively permissive, for maximum interoperability, a host that expects to receive mail SHOULD avoid defining mailboxes where the Local-part requires (or uses) the Quoted-string form or where the Local-part is case-sensitive. For any purposes that require generating or comparing Local-parts (e.g., to specific mailbox names), all quoted forms MUST be treated as equivalent, and the sending system SHOULD transmit the form that uses the minimum quoting possible.

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). These characters MUST NOT be used in MAIL or RCPT commands or other commands that require mailbox names.

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.

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 [4]), 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 [123.255.37.2], 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 4291 [9] for IPv6).
[[CREF11: [5321bis] Proposed erratum 4315 (2015-03-27) suggests yet another modification to the IPv6 address literal syntax, based on part on RFC 5952. We should consider whether those, or other, modifications are appropriate and/or whether, given both the issues of spam/malware and servers supporting multiple domains, it it time to deprecate mailboxes containing address literals entirely (EHLO fields may be a different issue). If we are going to allow IPv6 address literals, it may be time to incorporate something by reference rather than including specific syntax here (RFC 5952 is 14 pages long and does not contain any ABNF).]]

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

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 [40].

ls32              = ( h16 ":" h16 ) / IPv4address
                  ; least-significant 32 bits of address

h16               = 1*4HEXDIG
                  ; 16 bits of address represented in hexadecimal
                  [[CREF12: [5321bis](2821ter) 2821bis Last Call
                  comment]]
```

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. 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.

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 actually corresponds to the IP address of the client. [[CREF13: [5321bis] [[Note in draft -- proposed change to "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." --David MacQuigg, david_macquigg@yahoo.com, Friday, 20090130 0637 -0700]]]] However, if the verification fails, the server MUST NOT refuse to accept a message on that basis. Information captured in the verification attempt is for logging and tracing purposes. Note that this prohibition applies to the matching of the parameter to its IP address only; see Section 7.9 for a more extensive discussion of rejecting incoming connections or mail messages.

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.

The MAIL command (or the obsolete SEND, SOML, or SAML commands)

[[5321bis Editor's Note: is there any reason to not clean those commands out of this entirely, replacing them with, e.g., a strong reference to Appendix F.6]]
begins a mail transaction. Once started, a mail transaction consists of a transaction beginning command, 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. There may be zero or more transactions in a session. MAIL (or SEND, SOML, or SAML) 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. [[CREF14: [5321bis] 2821ter note: see comment about changing this convoluted discussion to talk about 'mail transaction' above. --Jck]]

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.

4.1.5. Private-Use Commands

As specified in Section 2.2.2, commands starting in "X" may be used by bilateral agreement between the client (sending) and server (receiving) SMTP agents. An SMTP server that does not recognize such a command is expected to reply with "500 Command not recognized". An extended SMTP server MAY list the feature names associated with these private commands in the response to the EHLO command.

Commands sent or accepted by SMTP systems that do not start with "X" MUST conform to the requirements of Section 2.2.2.

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.

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.
[[CREF15: [5321bis] 20140804: New text to clear up ambiguity.]]
An SMTP server SHOULD use the text shown in the examples whenever appropriate.

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. Whenever possible, a sender-SMTP SHOULD test the first digit (severity indication) of a reply code it receives.
[[CREF16: [5321bis] 20141209 [[Note in Draft: What is that sentence supposed to be tell us? Test the first digit and examine the others only if necessary? Note the interaction between this and various flaps about adding new codes.]]]]

The list of codes that appears below MUST NOT be construed as permanent. While the addition of new codes should be a rare and significant activity, with supplemental information in the textual part of the response (including enhanced status codes [34] and the successors to that specification)
[[CREF17: [5321bis] 20140802: New text for clarity]]
being preferred, new codes may be added as the result of new Standards or 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 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. 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) [14] 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

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)

hangText="521"><domain> No mail service here. [[CREF18: [5321bis]20140804: Specific code introduced with RFC 1846, updated and specified in draft-klensin-smtp-521code.]]

556 No mail service at this domain. [[CREF19: [5321bis] 20140912: Specific code introduced in draft-klensin-smtp-521code-02 (RFC 7504), largely for nullMX]]

250 Requested mail action okay, completed

251 User not local; will forward to <forward-path> (See Section 3.4)

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

455 Server unable to accommodate parameters

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

450 Requested mail action not taken: mailbox unavailable (e.g., mailbox busy or temporarily blocked for policy reasons)

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

451 Requested action aborted: error in processing

551 User not local; please try <forward-path> (See Section 3.4)

452 Requested action not taken: insufficient system storage

552 Requested mail action aborted: exceeded storage allocation

553 Requested action not taken: mailbox name not allowed (e.g., mailbox syntax incorrect)

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

554 Transaction failed (Or, in the case of a connection-opening response, "No SMTP service here")
[[CREF20: [5321bis] [[Note in Draft: Revise above statement in the light of new 521 code??]]]]

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)

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)

451 Requested action aborted: local error in processing

452 Requested action not taken: insufficient system storage

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

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)

552 Requested mail action aborted: exceeded storage allocation

553 Requested action not taken: mailbox name not allowed (e.g., mailbox syntax incorrect)

554 Transaction failed (Or, in the case of a connection-opening response, "No SMTP service here")

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

556 No mail service at this domain.

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, and 556 codes

[[CREF21: [5321bis] This section is new with 5321bis.]]

Codes 521, 554, and 556 are all used to report different types of "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 1846 [19] 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. 556 is 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 (e.g., from a DNS entry [51]) that the recipient domain does not accept mail. It would normally be used in response to a RCPT or similar (extension) command 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.

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.

- o 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 user who originated the message SHOULD be able 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 table below 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. [[CREF22: [5321bis] 20140914: Above sentence is new text based on yet another round of discussions about "invalid codes".]]

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 [34] 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

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)

MAIL

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

RCPT

S: 250, 251 (but see Section 3.4 for discussion of 251 and 551)
E: 550, 551, 552, 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 an SMTP server receives a message for delivery or further processing, it MUST insert trace (often referred to as "time stamp" or "Received" information) [[CREF23: [5321bis] See note on rfc5321bis-00c above]] 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 RFC 822, 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 UT when feasible. This formulation allows slightly more information about local circumstances to be specified. If UT is needed, the receiver need merely do some simple arithmetic to convert the values. Use of UT 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.

When the delivery SMTP server makes the "final delivery" of a message, it inserts a return-path line at the beginning of the mail data. This use of return-path is required; mail systems MUST support it. The return-path line preserves the information in the <reverse-path> from the MAIL command. 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.

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 5322 [11]).

It is sometimes difficult for an SMTP server to determine whether or not 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.

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 RFC 822 syntax 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.

Historical note: Text in RFC 822 that appears to contradict the use of the Return-path header field (or the envelope reverse-path address from the MAIL command) as the destination for error messages is not applicable on the Internet. The reverse-path address (as copied into the Return-path) MUST be used as the target of any mail containing delivery error messages.

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.

The server must give special treatment to cases in which the 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.

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. Note that the key difference between handling aliases (Section 3.9.1) and forwarding (this subsection) is the change to the backward-pointing address in this case. All notification messages about undeliverable mail MUST be sent using the MAIL command (even if they result from processing the obsolete SEND, SOML, or SAML commands) and MUST use a null return path as discussed in Section 3.6.

The time stamp line and the return path line are formally defined as follows (the definitions for "FWS" and "CFWS" appear in RFC 5322 [11]):

```
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 RFC 5322 [11]
                 ; 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 5322 [11]

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

   Additional-Registered-Clauses  = 1* (CFWS Atom FWS String)
                  [[CREF24: [5321bis] 5321 errata #1683, 20090215,
                  Roberto Javier Godoy, rjgodoy@fich.unl.edu.ar]]
                  ; Additional standard clauses may be added in this
                  ; location by future standards and registration with
                  ; IANA.  SMTP servers SHOULD NOT use unregistered
                  ; names.  See Section 8.

   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" / Attdl-Protocol

   Addtl-Protocol = Atom
                  ; Additional standard names for protocols are
                  ; registered with the Internet Assigned Numbers
                  ; Authority (IANA) in the "mail parameters"
                  ; registry [7].  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. See Section 3.6 for special treatment of these conditions in SMTP systems serving a relay function.

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 [26]) 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.

[[CREF25: [5321bis] [[Note in Draft: Klensin 20191126: Given the controversy on the SMTP mailing list between 20191123 and now about maximum lengths, is the above adequate or is further tuning of the limit text below needed?]]]]

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 [24]), 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 [6], 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.

4.5.3.1.10. Too Many Recipients Code

RFC 821 [3] 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. Clients SHOULD treat a 552 code in this case as a temporary, rather than permanent, failure so the logic below works.

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 response code of 452 (but the client SHOULD also be prepared for a 552, as noted above). If the server has a configured site-policy limitation on the number of RCPT commands, it MAY instead use a 5yz response 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 (or 552) 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 [23].

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 3.7, other kinds of Delivery Status Notifications (DSNs, RFC 3461 [33]), and Message Disposition Notifications (MDNs, RFC 3798 [37]). 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

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 (RFC 1035 [4]). The names are expected to be fully-qualified domain names (FQDNs): mechanisms for inferring FQDNs from partial names or local aliases are outside of this specification. Due to a history of problems, SMTP servers used for initial submission of messages SHOULD NOT make such inferences (Message Submission Servers [42] have somewhat more flexibility) and intermediate (relay) SMTP servers MUST NOT make them.

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.

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.

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.
[[5321bis Editor's Note: Depending on how the "null MX" discussion unfolds, some additional text may be in order here (20140718)]]
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. Any other response, specifically including a value that will return a CNAME record when queried, lies outside the scope of this Standard. The prohibition on labels in the data that resolve to CNAMEs is discussed in more detail in RFC 2181, Section 10.3 [27].

When the lookup succeeds, the mapping can result in a list of alternative delivery addresses rather than a single address, because of multiple MX records, multihoming, or both. To provide reliable mail transmission, the SMTP client MUST be able to try (and retry) each of the relevant addresses in this list in order, until a delivery attempt succeeds. However, 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.

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

MX records contain a preference indication that MUST be used in sorting if more than one such record appears (see below). Lower numbers are more preferred than higher ones. If there are multiple destinations 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 (perhaps taken from the preferred MX record) may be multihomed, in which case 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 probability of timely delivery, and indeed sometimes the probability 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 sort the MX records to determine candidates for delivery. The records are first ordered by preference, with the lowest-numbered records being most preferred. The relay host MUST then 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 the message MUST be returned as undeliverable. If records do remain, they SHOULD be tried, best preference first, as described above.

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 [39] 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. It must take this responsibility seriously. It MUST NOT lose the message for frivolous reasons, such as because the host later crashes or because of a predictable resource shortage. Some reasons that are not considered frivolous 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. If the address is an explicit source route, it MUST be stripped down to its final hop.

For example, suppose that an error notification must be sent for a message that arrived with:

```
MAIL FROM:<@a,@b:user@d>
```

The notification message MUST be sent using:

```
RCPT TO:<user@d>
```

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 [16] 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 or 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 [13], Post Office Protocol (POP) version 3 [22], IMAP version 2 [18], and PCMAIL [17]). 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 [12]). 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 4409 [42], 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

7.1. Mail Security and Spoofing

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 [20]

and, e.g., Pretty Good Privacy (PGP) in RFC 4880 [45] or Secure/ Multipurpose Internet Mail Extensions (S/MIME) in RFC 3851 [38]).

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. "Blind" Copies

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". Especially 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 full set of RCPT command arguments into the header section, either as part of trace header fields or as informational or private-extension header fields. [[CREF26: [rfc5321bis] [[Note in draft - Suggestion from 20070124 that got lost: delete "especially" and "the full set of" -- copying the first one can be as harmful as copying all of them, at least without verifying that the addresses do appear in the headers.]] Arnt Gulbrandsen, arnt@oryx.com, 2007.01.24 1121+0100]] Since this rule is often violated in practice, and cannot

be enforced, sending SMTP systems that are aware of "bcc" use MAY find it helpful to send each blind copy as a separate message transaction containing only a single RCPT command.

There is no inherent relationship between either "reverse" (from MAIL, SAML, etc., commands) 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 response code (see discussion above).

7.4. Mail Rerouting Based on the 251 and 551 Response 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 ("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 be supplied with caution or not at all when multiple recipients are involved lest it inadvertently disclose the identities of "blind copy" recipients to others.

7.7. Information Disclosure in Message Forwarding

As discussed in Section 3.4, 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. 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). 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 TO 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.

In recent years, use of the relay function through arbitrary sites has been used as part of hostile efforts to hide the actual origins of mail. Some sites have decided to limit the use of the relay function to known or identifiable sources, and implementations SHOULD provide the capability to perform this type of filtering. 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

IANA maintains three registries in support of this specification, all of which were created for RFC 2821 or earlier. This document expands the third one as specified below. The registry references listed are as of the time of publication; IANA does not guarantee the locations associated with the URLs. The registries are as follows:

- The first, "Simple Mail Transfer Protocol (SMTP) Service Extensions" [49], consists of SMTP service extensions with the associated keywords, and, as needed, parameters and verbs. As specified in Section 2.2.2, no entry may be made in this registry that starts in an "X". Entries may be made only for service

extensions (and associated keywords, parameters, or verbs) that are defined in Standards-Track or Experimental RFCs specifically approved by the IESG for this purpose.

- The second registry, "Address Literal Tags" [50], consists of "tags" that identify forms of domain literals other than those for IPv4 addresses (specified in RFC 821 and in this document). The initial entry in that registry is for IPv6 addresses (specified in this document). Additional literal types require standardization before being used; none are anticipated at this time.

- The third, "Mail Transmission Types" [49], 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 by standardization or by way of an RFC-documented, IESG-approved, Experimental protocol extension. This name space is for identification and not limited in size: the IESG is encouraged to approve on the basis of clear documentation and a distinct method rather than preferences about the properties of the method itself.

  An additional subsection has been added to the "VIA link types" and "WITH protocol types" subsections of this registry to contain registrations of "Additional-registered-clauses" as described above. The registry will contain clause names, a description, a summary of the syntax of the associated String, and a reference. As new clauses are defined, they may, in principle, specify creation of their own registries if the Strings consist of reserved terms or keywords rather than less restricted strings. As with link and protocol identifiers, additional clauses may be registered only by standardization or by way of an RFC-documented, IESG-approved, Experimental protocol extension. The additional clause name space is for identification and is not limited in size: the IESG is encouraged to approve on the basis of clear documentation, actual use or strong signs that the clause will be used, and a distinct requirement rather than preferences about the properties of the clause itself.

In addition, if additional trace header fields (i.e., in addition to Return-path and Received) are ever created, those trace fields MUST be added to the IANA registry established by BCP 90 (RFC 3864) [8] for use with RFC 5322 [11].

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. Those 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.

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. In addition, specific suggestions that led to corrections and improvements in this version were received from Ned Freed, Barry Leiba, Ivar Lumi, Pete Resnick, and others.

chetti contributed an analysis that clarify the ABNF productions that implicitly reference other document.

[[CREF27: Most errata and comments after 2019-07-01 have not yet been captured in this version of the draft.]]

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] American National Standards Institute (formerly United States of America Standards Institute), "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.

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

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

[5] 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>.

[6] 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>.

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

[8] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, DOI 10.17487/RFC3864, September 2004, <https://www.rfc-editor.org/info/rfc3864>.

[9] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[10] 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>.

[11] Resnick, P., "Internet Message Format", RFC 5322, September 2008.

10.2. Informative References

[12] 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>.

[13] 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>.

[14] 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>.

[15] 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>.

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

[17] 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>.

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

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

[20] 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>.

[21] 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>.

[22] 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>.

[23] 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>.

[24] 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>.

[25] 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>.

[26] 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>.

[27] 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>.

[28] 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>.

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

[30] 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>.

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

[32] 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>.

[33] 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>.

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

[35] 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>.

[36] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, DOI 10.17487/RFC3501, March 2003, <https://www.rfc-editor.org/info/rfc3501>.

[37] Hansen, T., Ed. and G. Vaudreuil, Ed., "Message Disposition Notification", RFC 3798, DOI 10.17487/RFC3798, May 2004, <https://www.rfc-editor.org/info/rfc3798>.

[38] Ramsdell, B., Ed., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, DOI 10.17487/RFC3851, July 2004, <https://www.rfc-editor.org/info/rfc3851>.

[39] 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>.

[40] 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>.

[41] Wong, M. and W. Schlitt, "Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1", RFC 4408, DOI 10.17487/RFC4408, April 2006, <https://www.rfc-editor.org/info/rfc4408>.

[42] Gellens, R. and J. Klensin, "Message Submission for Mail", RFC 4409, DOI 10.17487/RFC4409, April 2006, <https://www.rfc-editor.org/info/rfc4409>.

[43] Fenton, J., "Analysis of Threats Motivating DomainKeys Identified Mail (DKIM)", RFC 4686, DOI 10.17487/RFC4686, September 2006, <https://www.rfc-editor.org/info/rfc4686>.

[44] Allman, E., Callas, J., Delany, M., Libbey, M., Fenton, J., and M. Thomas, "DomainKeys Identified Mail (DKIM) Signatures", RFC 4871, DOI 10.17487/RFC4871, May 2007, <https://www.rfc-editor.org/info/rfc4871>.

[45] 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>.

[46] 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>.

[47] 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>.

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

[49] Internet Assigned Number Authority (IANA), "IANA Mail Parameters", 2007, <http://www.iana.org/assignments/mail-parameters>.

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

[51] Levine, J. and M. Delany, "A "Null MX" No Service Resource Record for Domains that Accept No Mail", September 2014, <https://datatracker.ietf.org/doc/draft-ietf-appsawg-nullmx/>.

[52] RFC Editor, "RFC Errata - RFC 5321", 2019, <https://www.rfc-editor.org/errata/rfc5321>.

Captured 2019-11-19

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 RFC 822 Header Fields

Some systems use an RFC 822 header section (only) in a mail submission protocol, or otherwise generate SMTP commands from RFC 822 header fields when such a message is handed to an MTA from a UA. While the MTA-UA protocol 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 the UA provide its initial ("submission client") MTA with an envelope separate from the message itself. However, if the envelope is not supplied, SMTP commands 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. Once this process is completed, the remaining header fields SHOULD be checked to verify that at least one TO, CC, or BCC header field remains. If none do, then a BCC header field with no additional information SHOULD be inserted as specified in [11].

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. Source Routes

Historically, the <reverse-path> was a reverse source routing list of hosts and a source mailbox. The first host in the <reverse-path> was historically the host sending the MAIL command; today, source routes SHOULD NOT appear in the reverse-path. Similarly, the <forward-path> may be a source routing lists of hosts and a destination mailbox. However, in general, the <forward-path> SHOULD contain only a mailbox and domain name, relying on the domain name system to supply routing information if required. The use of source routes is deprecated (see Appendix F.2); while servers MUST be prepared to receive and handle them as discussed in Section 3.3 and Appendix F.2, clients SHOULD NOT transmit them and this section is included in the current specification only to provide context. It has been modified somewhat from the material in RFC 821 to prevent server actions that might confuse clients or subsequent servers that do not expect a full source route implementation.

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.[[CREF28: [5321bis]JcK 20090123: Tightened this and the next paragraph to be clear that this doesn't authorize source route use.]]

If source routes are used contrary to requirements and recommendations elsewhere in this specfiication, 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. The reverse-path SHOULD NOT be updated by servers conforming to this specification.

Notice that the forward-path and reverse-path appear in the SMTP commands and replies, but not necessarily in the message. That is, there is no need for these paths and especially this syntax to appear in the "To:" , "From:", "CC:", etc. fields of the message header section. Conversely, SMTP servers MUST NOT derive final message routing information from message header fields.

When the list of hosts is present despite the recommendations and requirements [[CREF29: [5321bis]JcK 20090123 "and requrements" added]] above, it is a "reverse" source route and indicates that the mail was relayed through each host on the list (the first host in the list was the most recent relay). This list is used as a source route to return non-delivery notices to the sender. If, contrary to the recommendations here, a relay host adds itself to the beginning of the list, it MUST use its name as known in the transport environment to which it is relaying the mail rather than that of the transport environment from which the mail came (if they are different). Note that a situation could easily arise in which some relay hosts add their names to the reverse source route and others do not, generating discontinuities in the routing list. This is another reason why servers needing to return a message SHOULD ignore the source route entirely and simply use the domain as specified in the Mailbox.

Appendix D. 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.

D.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
```

D.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
```

D.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
```

D.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 E. 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.

Appendix F. 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 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. [[CREF30: [5321bis] (2821ter) 2821bis Last Call Comment]]

F.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.

F.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. The requirement to utilize source routes in regular mail traffic was eliminated by the introduction of the domain name system "MX" record 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. Consequently, the only remaining justifications for the use of source routes are support for very old SMTP clients or MUAs and in mail system debugging. They can, however, still be useful in the latter

circumstance and for routing mail around serious, but temporary, problems such as problems with the relevant DNS records.

SMTP servers MUST continue to accept source route syntax as specified in the main body of this document and in RFC 1123. They MAY, if necessary, 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.

Clients SHOULD NOT utilize explicit source routing except under unusual circumstances, such as debugging or potentially relaying around firewall or mail system configuration errors.

F.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.

F.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.

F.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.

F.6. Sending versus Mailing

In addition to specifying a mechanism for delivering messages to user's 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. [[5321bis Editor's Note: does this need a stronger reference to 821, 2821, and/or 5321?]]

Clients SHOULD NOT provide SEND, SAML, or SOML as services. Servers MAY implement them. If they are implemented by servers, the

implementation model specified in RFC 821 MUST be used and the command names MUST be published in the response to the EHLO command.

Appendix G. Other Outstanding Issues

[[RFC Editor: Please remove this section before publication.]]

In December 2019, an issue was raised on the ietf-smtp@ietf.org list that led to a broad discussion of ways in which existing practice had diverged from the specifications and recommendations of RFC 5321 in the more than eleven years since it was published (some of those issues probably affect the boundary between RFC 5321 and 5322 and hence the latter as well). In most cases, those divergences call for revision of the Technical Specification to match the practice, clarification of the specification text in other ways, or a more comprehensive explanation of why the practices recommended by the specification should really be followed.

Those discussions raised two other issues, which were that

- The publication of the Submission Server specification of RFC 6409 in November 2011 may not have been fully reflected in RFC 5321 (despite the even earlier publication of RFC 4409) and

- There may be inconsistencies between the July 2009 Internet Mail Architecture description of RFC 5598 and the model described in RFC 5321. The issue called out in Appendix G.3 below may be an example of one of those inconsistencies.

Those discrepancies should be identified and discussed and decisions made to fix them (and where) or to ignore them and let them continue.

There has also been discussion on the mailing list, perhaps amounting to very rough consensus, that any revision of RFC 5321 and/or 5322 should be accompanied by a separate Applicability Statement document that would make recommendations about applicability or best practices in particular areas rather than trying to get everything into the two technical specifications. This appendix does not attempt to identify which issues should get which treatment.

Until and unless there is a WG with appropriate leadership and tracking mechanisms, this appendix will act as a temporary record of issues that should be discussed and decided upon before a revised SMTP specification (or a related Applicability Statement) is published, issues that have not been reflected in errata (see Appendix H.1 below for those covered by errata).

G.1. IP Address Literals

The specification is unclear about whether IP address literals, particularly IP address literals used as arguments to the EHLO command, are required to be accepted or whether they are allowed to be rejected as part of the general "operational necessity" exception. Some have suggested that rejection of them is so common as an anti-spam measure that the use of such literals should be deprecated entirely in the specification, others that the are still useful and used and/or that, whatever is said about IP address literals within an SMTP session (e.g., in MAIL or RCPT commands), they should continue to be allowed (and required) in EHLO.

G.2. Repeated Use of EHLO

While the specification says that an SMTP client's sending EHLO again after it has been issued (starting an SMTP session and treats it as if RSET had been sent (closing the session) followed by EHLO, there are apparently applications, at least some of them involving setting up of secure connections, in which the second EHLO is required and does not imply RSET. Does the specification need to be adjusted to reflect or call out those cases?

G.3. Meaning of "MTA" and Related Terminology

A terminology issue has come up about what the term "MTA" actually refers to, a question that became at least slightly more complicated when we formalized RFC 6409 Submission Servers. Does the document need to be adjusted to be more clear about this topic? Note that the answer may interact with the question asked in Section 2 above. Possibly along the same lines, RFC 2821 changed the RFC 821 terminology from "sender-SMTP" and "receiver-SMTP" to "SMTP client" and "SMTP server" respectively. As things have evolved, it is possible that newer terminology is a source of confusion and that the terminology should be changed back, something that also needs discussion.

G.4. Originator, or Originating System, Authentication

Should RFC 5321bis address authentication and related issues or should Section 3.9 or other text be reshaped (in addition to or instead of the comment on that section) to lay a better foundation for such work, either in the context of mailing lists or more generally?

G.5. Remove or deprecate the work-around from code 552 to 452

The suggestion in Section 4.5.3.1.10 may have outlived its usefulness and/or be inconsistent with current practice. Should it be removed and/or explicitly deprecated?

G.6. Clarify where the protocol stands with respect to submission and TLS issues

1. submission on port 587

2. submission on port 465

3. TLS relay on a port different from 25 (whenever)

G.7. Probably-substantive Discussion Topics Identified in Other Ways

The following issues were identified as a group in the opening Note but called out specifically only in embedded CREF comments in earlier (-00 and -01) versions of this draft.

G.7.1. Issues with 521, 554, and 556 codes

See new Section 4.2.4.2. More text may be needed, there or elsewhere, about choices of codes in response to initial opening and to EHLO, especially to deal with selective policy rejections.

G.7.2. SMTP Model, terminology, and relationship to RFC 5598

CREF comment in Section 2 and also CREF comment in Section 2.3.10

G.7.3. Resolvable FQDNs and private domain names

Multiple CREF comments in Section 2.3.5

G.7.4. Possible clarification about mail transactions and transaction state

CREF comment in Section 3.3 and also reference in Section 4.1.4

G.7.5. Issues with mailing lists, aliases, and forwarding

CREF comment in Section 3.9. May also want to note forwarding as an email address portability issue.

G.7.6. Requirements for domain name and/or IP address in EHLO

CREF comment in Section 4.1.4

G.7.7. Does the 'first digit only' and/or non-listed reply code text need clarification?

CREF comments in Section 4.2 and Section 4.3.1

G.7.8. Size limits

CREF comment in Section 4.5.3

G.7.9. Discussion of 'blind' copies and RCPT

CREF comment in Section 7.2. May alto need to discussion whether that terminology is politically incorrect and suggest a replacement.

G.7.10. Further clarifications needed to source routes?

CREF comment in Appendix C

G.7.11. Should 1yz Be Revisited?

RFC 5421 depreciated the "positive preliminary reply" response code category with first digit "1", so that the first digit of valid SMTP response codes must be 2, 3, 4, or 5. It has been suggested (see mail from Hector Santos with Subject "SMTP Reply code 1yz Positive Preliminary reply", March 5, 2020 12:56 -0500, on the SMTP list) that these codes should be reinstated to deal with some situations that became more plausible after 5321 was published. Do we need to take this back up?

G.7.12. Review Timeout Specifications

RFC 5321 (and its predecessors going back to 821) specify minimum periods for client and server to wait before timing out. Are those intervals still appropriate in a world of faster processors and faster networks? Should they be updated and revised? Or should more qualifying language be added?

G.8. Enhanced Reply Codes and DSNs

Enhanced Mail System Status Codes [34] were added to SMTP before RFC 5321 was published and are now, together with a corresponding registry [46], widely deployed and in extensive use in the network. Similar, the structure and extensions options for Delivery Status Notifications [35] is implemented, deployed, and in wide use. Is it

time to fold all or part of those mature specifications into the SMTP spec or at least to mention and normatively reference them? And, as an aside do those specs need work or, if they are kept separate, is it time to move them to Internet Standard?

G.9. Revisiting Quoted Strings

Recent discussions both in and out of the IETF have highlighted instances of non-compliance with the specification of a Local-part consisting of a Quoted-string, whether any content of QcontentSMTP that actually requires special treatment consists of qtextSMTP, quoted-pairSMTP, or both. Section 4.1.2 (of RFC 5321, repeated above) ends with a few paragraphs of warnings (essentially a partial applicability statement), the first of which cautions against cleverness with either Quoted-string or case sensitivity as a threat to interoperability.

The Quoted-string portion of that discussion has apparently been widely not read or ignored. Do we need to do something else? If we do an Applicability Statement, would it be useful to either reference the discussion in this document from there or to move the discussion there and reference it (normatively?) from here?

G.10. Internationalization

RFC 5321 came long before work on internationalization of email addresses and headers (other than by use of encoded words in MINE) and specifically before the work of the EAI WG leading to the SMTPUTF8 specifications, specifically RFCs 6530ff. The second explanatory paragraph at the end of Section 4.1.2 ("Systems MUST NOT define mailboxes ...") is an extremely strong prohibition against the use of non-ASCII characters. Would it be appropriate to add something like "in the absence of relevant extensions" there? Also, given [mis]behavior seen in the wild, does that paragraph (or an A/S) need an explicit caution about SMTP servers or clients assuming they can apply the popular web convention of using %NN sequences as a way to encode non-ASCII characters (<pct-encoded> in RFC 3986) and assuming some later system will interpret it as they expect? Would it be appropriate to add an Internationalization Considerations section to the body of this document if only for the purpose of pointing people elsewhere?

G.11. SMTP Clients, Servers, Senders, and Receivers

RFC 821 used the terms "SMTP-sender" and "SMTP-receiver". In RFC 2821 (and hence in 5321), we switched that to "client" and "server" (See the discussion in Section 1.2). In part because a relay is a server and then a client (in some recent practice, even interleaving

the two functions by opening the connection to the next host in line and sending commands before the incoming transaction is complete), RFC 5321 continues to use the original terminology in some places. Should we revisit that usage, possibly even returning to consistent use of the original terminology?

Appendix H. Change log for RFC 5321bis

[[RFC Editor: Please remove this section before publication.]]

H.1. RFC 5321 Errata Summary

This document addresses the following errata filed against RFC 5321 since its publication in October 2008 [52] [[CREF31: [[Note in Draft: Items with comments below have not yet been resolved.]]]]

1683 ABNF error. Section 4.4

4198 Description error. Section 4.2

2578 Syntax description error. Section 4.1.2

1543 Wrong code in description Section 3.8

4315 ABNF - IPv6 Section 4.1.3. [[CREF32: [5321bis]The IPv6 syntax has been adjusted since 5321 was published. See the rewritten form and the comment in the section cited in the previous sentence. The editor awaits instructions. See https://www.rfc-editor.org/errata/eid4315]]

5414 ABNF for Quoted-string Section 4.1.2

1851 Location of text on unexpected close Section 4.1.1.5. [[CREF33: [5321bis]Matter of taste, editor seeks advice.]]

3447 Use of normative language (e.g., more "MUST"s), possible confusion in some sections Section 4.4. [[CREF34: [5321bis]As Barry notes in his verifier comments on the erratum (see https://www.rfc-editor.org/errata/eid3447), the comments and suggestions here raise a number of interesting (and difficult) issues. One of the issues is that the core of RFCs 5321 (and 2821) is text carried over from Jon Postel's RFC 821, a document that was not only written in a different style than the IETF uses today but that was written at a time when no one had dreamt of RFC 2119 or even the IETF itself. It appears to me that trying to patch that style might easily result in a document that is harder to read as well as being error prone. If we want to get the document entirely into contemporary style, we really should bite

the bullet and do a complete rewrite. To respond to a different point in Barry's discussion, I think an explicit statement that 5321/5322 and their predecessors differ in places and why would be helpful. Text, and suggestions about where to put it, are solicited. A list of differences might be a good idea too, but getting it right might be more work than there is available energy to do correctly.]]

5711 Missing leading spaces in example Appendix D.3. [[CREF35: [5321bis]Well, this is interesting because the XML is correct and the spaces are there, embedded in artwork. So either the XML2RFC processor at the time took those leading spaces out or the RFC Editor improved on the document and the change was not caught in AUTH48, perhaps because rfcdiff ignores white space. We just need to watch for future iterations.]]

[[CREF36: [5321bis]Note that rejected errata have _not_ been reviewed to see if they contain anything useful that should be discussed again with the possibility of rethinking and changing text. Volunteers sought.]]

H.2. Changes from RFC 5321 (published October 2008) to the initial (-00) version of this draft

- Acknowledgments section (Section 9) trimmed back for new document.
- Introductory paragraph to Appendix F extended to make it clear that these features were deprecated a long time ago and really should not be in use any more.
- Adjusted some language to clarify that source routes really, really, should not be used or depended upon.
- IPv6 address syntax replaced by a copy of the IPv6 URI syntax and a note added.
- Production index added as a first step in tying all productions to their sources. As part of the effort to make the document more easily navigable, table of contents entries have been created for the individual command descriptions.
- Clarified the relationship between the SMTP "letters, digits, and hyphens" and DNS "preferred name syntax" (Section 2.3.5).
- Revised the reply code sections to add new 521 and 556 codes, clarify relationships, and be explicit about the requirement for clients to rely on first digits rather than the sequences in Section 4.3.2.

- In conjunction with the above, explicitly obsolete RFCs 1846 and 7504.
- Incorporated a correction reflecting Errata ID 2578.
- Some small editorial changes made to eliminate redundant statements that were very close together. Other, equally small, editorial changes have been made to improve grammar or clarity.
- A few questions, marked "[[5321bis Editor's Note:", or "[[Note in Draft" have been added for the group to resolve. Other questions, especially those in the errata summary, are simply included in narrative comments in CREFs.
- Checked and rationalized "response" (to a command) and "reply code" terminology. One can talk about a "999 response" but only a "999 reply code". There is no such thing as a "response code".
- Added note about length limit on mailbox names ("email addresses").
- Added an "errata summary" subsection to this change log/ comparison to 5321 in this Appendix. The entire Appendix will, of course, disappear at the time of RFC publication unless someone wants to make a strong case for retaining it.
- Rationalized CREFs to 2821, 5321, 5321bis etc.; added note to readers below the Abstract.
- Temporarily added a "Note on Reading This Working Draft" after the Abstract.

H.3. Changes Among Versions of Rfc5321bis

H.3.1. Changes from draft-klensin-rfc5321bis-00 (posted 2012-12-02) to -01

Substantively, these two versions differ only by suppression of the CREF and other discussion associated with the evolution from RFC 2821 to RFC 5321. That change includes an update to the document's Note to Readers, the date, the file name, and the addition of this change log subsection.

H.3.2. Changes from draft-klensin-rfc5321bis-01 (20191203) to -02

- Minor clarifications to improve text, e.g., addition of NOOP to the list of non-mail transaction examples in Section 4.1.4.

- Added topics exposed in the ietf-smtp list and the IETF list "dogfood" discussion during December 2019 and an index listing of substantive issues identified only in CREFs in the prior draft as a new Appendix G..

H.3.3. Changes from draft-klensin-rfc5321bis-02 (2019-12-27) to -03

- Added more text to Appendix G.7.1 to specifically call out the session-opening policy issues surrounding these codes.
- Added discussion of "1yz" reinstatement in Appendix G.7.11.
- Added discussion of timeouts in Appendix G.7.12.
- Added subsection on Enhanced Status Codes and DSNs to the outstanding issues list Appendix G.8.
- Replaced reference to RFC 1652 (8BITMIME) with the Internet Standard version, RFC 6152.
- With help from cketti, clarified the ABNF productions whose terminals appear in other documents.
- Added discussions of Quoted-string, Internationalization, and client-server versus sender-receiver terminology to Appendix G.
- Added note to the Abstract.

Index

Author's Address

John C. Klensin
1770 Massachusetts Ave, Suite 322
Cambridge, MA 02140
USA

EMail: john-ietf@jck.com

Network Working Group
Internet-Draft
Obsoletes: 5322 (if approved)
Updates: 4021 (if approved)
Intended status: Standards Track
Expires: 31 December 2020

P. Resnick, Ed.
Episteme

29 June 2020

Internet Message Format
draft-resnick-rfc5322bis-01

Abstract

This document specifies the Internet Message Format (IMF), a syntax for text messages that are sent between computer users, within the framework of "electronic mail" messages. This specification is a revision of Request For Comments (RFC) 5322, itself a revision of Request For Comments (RFC) 2822, all of which supersede Request For Comments (RFC) 822, "Standard for the Format of ARPA Internet Text Messages", updating it to reflect current practice and incorporating incremental changes that were specified in other RFCs.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 31 December 2020.

Copyright Notice

Table of Contents

1. Introduction

1.1. Scope

This document specifies the Internet Message Format (IMF), a syntax for text messages that are sent between computer users, within the framework of "electronic mail" messages. This specification is an update to [RFC5322], itself a revision of [RFC2822], all of which supersede [RFC0822], updating it to reflect current practice and incorporating incremental changes that were specified in other RFCs such as [RFC1123].

This document specifies a syntax only for text messages. In particular, it makes no provision for the transmission of images, audio, or other sorts of structured data in electronic mail messages. There are several extensions published, such as the MIME document series ([RFC2045], [RFC2046], [RFC2049]), which describe mechanisms for the transmission of such data through electronic mail, either by extending the syntax provided here or by structuring such messages to conform to this syntax. Those mechanisms are outside of the scope of this specification.

In the context of electronic mail, messages are viewed as having an envelope and contents. The envelope contains whatever information is needed to accomplish transmission and delivery. (See [I-D.klensin-rfc5321bis] for a discussion of the envelope.) The contents comprise the object to be delivered to the recipient. This specification applies only to the format and some of the semantics of message contents. It contains no specification of the information in the envelope.

However, some message systems may use information from the contents to create the envelope. It is intended that this specification facilitate the acquisition of such information by programs.

This specification is intended as a definition of what message content format is to be passed between systems. Though some message systems locally store messages in this format (which eliminates the need for translation between formats) and others use formats that differ from the one specified in this specification, local storage is outside of the scope of this specification.

> Note: This specification is not intended to dictate the internal formats used by sites, the specific message system features that they are expected to support, or any of the characteristics of user interface programs that create or read messages. In addition, this document does not specify an encoding of the characters for either transport or storage; that is, it does not specify the number of bits used or how those bits are specifically transferred over the wire or stored on disk.

1.2. Notational Conventions

1.2.1. Requirements Notation

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 [BCP14] RFC2119 RFC8174 when, and only when, they appear in all capitals, as shown here.

1.2.2. Syntactic Notation

This specification uses the Augmented Backus-Naur Form (ABNF) [STD68] notation for the formal definitions of the syntax of messages. Characters will be specified either by a decimal value (e.g., the value %d65 for uppercase A and %d97 for lowercase A) or by a case-insensitive literal value enclosed in quotation marks (e.g., "A" for either uppercase or lowercase A).

1.2.3. Structure of This Document

This document is divided into several sections.

This section, section 1, is a short introduction to the document.

Section 2 lays out the general description of a message and its constituent parts. This is an overview to help the reader understand some of the general principles used in the later portions of this document. Any examples in this section MUST NOT be taken as specification of the formal syntax of any part of a message.

Section 3 specifies formal ABNF rules for the structure of each part of a message (the syntax) and describes the relationship between those parts and their meaning in the context of a message (the semantics). That is, it lays out the actual rules for the structure of each part of a message (the syntax) as well as a description of the parts and instructions for their interpretation (the semantics). This includes analysis of the syntax and semantics of subparts of

messages that have specific structure. The syntax included in section 3 represents messages as they MUST be created. There are also notes in section 3 to indicate if any of the options specified in the syntax SHOULD be used over any of the others.

Both sections 2 and 3 describe messages that are legal to generate for purposes of this specification.

Section 4 of this document specifies an "obsolete" syntax. There are references in section 3 to these obsolete syntactic elements. The rules of the obsolete syntax are elements that have appeared in earlier versions of this specification or have previously been widely used in Internet messages. As such, these elements MUST be interpreted by parsers of messages in order to be conformant to this specification. However, since items in this syntax have been determined to be non-interoperable or to cause significant problems for recipients of messages, they MUST NOT be generated by creators of conformant messages.

Section 5 details security considerations to take into account when implementing this specification.

Appendix A lists examples of different sorts of messages. These examples are not exhaustive of the types of messages that appear on the Internet, but give a broad overview of certain syntactic forms.

Appendix B lists the differences between this specification and earlier specifications for Internet messages.

Appendix C contains acknowledgements.

2. Lexical Analysis of Messages

2.1. General Description

At the most basic level, a message is a series of characters. A message that is conformant with this specification is composed of characters with values in the range of 1 through 127 and interpreted as US-ASCII [ANSI.X3-4.1986] characters. For brevity, this document sometimes refers to this range of characters as simply "US-ASCII characters".

> Note: This document specifies that messages are made up of characters in the US-ASCII range of 1 through 127. There are other documents, specifically the MIME document series ([RFC2045], [RFC2046], [RFC2047], [RFC2049], [BCP13]) and the Internationalized Email Headers specification ([RFC6532]), that extend this specification to allow for values outside of that range. Discussion of those mechanisms is not within the scope of this specification.

Messages are divided into lines of characters. A line is a series of characters that is delimited with the two characters carriage-return and line-feed; that is, the carriage return (CR) character (ASCII value 13) followed immediately by the line feed (LF) character (ASCII value 10). (The carriage return/line feed pair is usually written in this document as "CRLF".)

A message consists of header fields (collectively called "the header section of the message") followed, optionally, by a body. The header section is a sequence of lines of characters with special syntax as defined in this specification. The body is simply a sequence of characters that follows the header section and is separated from the header section by an empty line (i.e., a line with nothing preceding the CRLF).

> Note: Common parlance and earlier versions of this specification use the term "header" to either refer to the entire header section or to refer to an individual header field. To avoid ambiguity, this document does not use the terms "header" or "headers" in isolation, but instead always uses "header field" to refer to the individual field and "header section" to refer to the entire collection.

2.1.1. Line Length Limits

There are two limits that this specification places on the number of characters in a line. Each line of characters MUST be no more than 998 characters, and SHOULD be no more than 78 characters, excluding the CRLF.

The 998 character limit is due to limitations in many implementations that send, receive, or store IMF messages which simply cannot handle more than 998 characters on a line. Receiving implementations would do well to handle an arbitrarily large number of characters in a line for robustness sake. However, there are so many implementations that (in compliance with the transport requirements of [I-D.klensin-rfc5321bis]) do not accept messages containing more than 1000 characters including the CR and LF per line, it is important for implementations not to create such messages.

The more conservative 78 character recommendation is to accommodate the many implementations of user interfaces that display these messages which may truncate, or disastrously wrap, the display of more than 78 characters per line, in spite of the fact that such implementations are non-conformant to the intent of this specification (and that of [I-D.klensin-rfc5321bis] if they actually cause information to be lost). Again, even though this limitation is put on messages, it is incumbent upon implementations that display messages to handle an arbitrarily large number of characters in a line (certainly at least up to the 998 character limit) for the sake of robustness.

2.2. Header Fields

Header fields are lines beginning with a field name, followed by a colon (":"), followed by a field body, and terminated by CRLF. A field name MUST be composed of visible US-ASCII characters (i.e., characters that have values between 33 and 126, inclusive), except colon. A field body may be composed of visible US-ASCII characters as well as the space (SP, ASCII value 32) and horizontal tab (HTAB, ASCII value 9) characters (together known as the white space characters, WSP). A field body MUST NOT include CR and LF except when used in "folding" and "unfolding", as described in section 2.2.3. All field bodies MUST conform to the syntax described in sections 3 and 4 of this specification.

2.2.1. Unstructured Header Field Bodies

Some field bodies in this specification are defined simply as "unstructured" (which is specified in section 3.2.5 as any visible US-ASCII characters plus white space characters) with no further restrictions. These are referred to as unstructured field bodies. Semantically, unstructured field bodies are simply to be treated as a single line of characters with no further processing (except for "folding" and "unfolding" as described in section 2.2.3).

2.2.2. Structured Header Field Bodies

Some field bodies in this specification have a syntax that is more restrictive than the unstructured field bodies described above. These are referred to as "structured" field bodies. Structured field bodies are sequences of specific lexical tokens as described in sections 3 and 4 of this specification. Many of these tokens are allowed (according to their syntax) to be introduced or end with comments (as described in section 3.2.2) as well as the white space characters, and those white space characters are subject to "folding" and "unfolding" as described in section 2.2.3. Semantic analysis of structured field bodies is given along with their syntax.

2.2.3. Long Header Fields

Each header field is logically a single line of characters comprising the field name, the colon, and the field body. For convenience however, and to deal with the 998/78 character limitations per line, the field body portion of a header field can be split into a multiple-line representation; this is called "folding". The general rule is that wherever this specification allows for folding white space (not simply WSP characters), a CRLF may be inserted before any WSP.

For example, the header field:

Subject: This is a test

can be represented as:

Subject: This
 is a test

> Note: Though structured field bodies are defined in such a way that folding can take place between many of the lexical tokens (and even within some of the lexical tokens), folding SHOULD be limited to placing the CRLF at higher-level syntactic breaks. For instance, if a field body is defined as comma-separated values, it is recommended that folding occur after the comma separating the structured items in preference to other places where the field could be folded, even if it is allowed elsewhere.

The process of moving from this folded multiple-line representation of a header field to its single line representation is called "unfolding". Unfolding is accomplished by simply removing any CRLF that is immediately followed by WSP. Each header field should be treated in its unfolded form for further syntactic and semantic evaluation. An unfolded header field has no length restriction and therefore may be indeterminately long.

2.3. Body

The body of a message is simply lines of US-ASCII characters. The only two limitations on the body are as follows:

* CR and LF MUST only occur together as CRLF; they MUST NOT appear independently in the body.

* Lines of characters in the body MUST be limited to 998 characters, and SHOULD be limited to 78 characters, excluding the CRLF.

> Note: As was stated earlier, there are other documents, specifically the MIME documents ([RFC2045], [RFC2046], [RFC2049], [BCP13]), that extend (and limit) this specification to allow for different sorts of message bodies. Again, these mechanisms are beyond the scope of this document.

3. Syntax

3.1. Introduction

The syntax as given in this section defines the legal syntax of Internet messages. Messages that are conformant to this specification MUST conform to the syntax in this section. If there are options in this section where one option SHOULD be generated, that is indicated either in the prose or in a comment next to the syntax.

For the defined expressions, a short description of the syntax and use is given, followed by the syntax in ABNF, followed by a semantic analysis. The following primitive tokens that are used but otherwise unspecified are taken from the "Core Rules" of [STD68], Appendix B.1: CR, LF, CRLF, HTAB, SP, WSP, DQUOTE, DIGIT, ALPHA, and VCHAR.

In some of the definitions, there will be non-terminals whose names start with "obs-". These "obs-" elements refer to tokens defined in the obsolete syntax in section 4. In all cases, these productions are to be ignored for the purposes of generating legal Internet messages and MUST NOT be used as part of such a message. However, when interpreting messages, these tokens MUST be honored as part of the legal syntax. In this sense, section 3 defines a grammar for the generation of messages, with "obs-" elements that are to be ignored, while section 4 adds grammar for the interpretation of messages.

3.2. Lexical Tokens

The following rules are used to define an underlying lexical analyzer, which feeds tokens to the higher-level parsers. This section defines the tokens used in structured header field bodies.

> Note: Readers of this specification need to pay special attention to how these lexical tokens are used in both the lower-level and higher-level syntax later in the document. Particularly, the white space tokens and the comment tokens defined in section 3.2.2 get used in the lower-level tokens defined here, and those lower-level tokens are in turn used as parts of the higher-level tokens defined later. Therefore, white space and comments may be allowed in the higher-level tokens even though they may not explicitly appear in a particular definition.

3.2.1. Quoted characters

Some characters are reserved for special interpretation, such as delimiting lexical tokens. To permit use of these characters as uninterpreted data, a quoting mechanism is provided.

quoted-pair = ("\" (VCHAR / WSP)) / obs-qp

Where any quoted-pair appears, it is to be interpreted as the character alone. That is to say, the "\" character that appears as part of a quoted-pair is semantically "invisible".

> Note: The "\" character may appear in a message where it is not part of a quoted-pair. A "\" character that does not appear in a quoted-pair is not semantically invisible. The only places in this specification where quoted-pair currently appears are ccontent, qcontent, and in obs-dtext in section 4.

3.2.2. Folding White Space and Comments

White space characters, including white space used in folding (described in section 2.2.3), may appear between many elements in header field bodies. Also, strings of characters that are treated as comments may be included in structured field bodies as characters enclosed in parentheses. The following defines the folding white space (FWS) and comment constructs.

Strings of characters enclosed in parentheses are considered comments so long as they do not appear within a "quoted-string", as defined in section 3.2.4. Comments may nest.

There are several places in this specification where comments and FWS may be freely inserted. To accommodate that syntax, an additional token for "CFWS" is defined for places where comments and/or FWS can occur. However, where CFWS occurs in this specification, it MUST NOT be inserted in such a way that any line of a folded header field is made up entirely of WSP characters and nothing else.

```
FWS             =   ([*WSP CRLF] 1*WSP) /  obs-FWS
                                        ; Folding white space

ctext           =   %d33-39 /           ; Visible US-ASCII
                    %d42-91 /           ;  characters not including
                    %d93-126 /          ;  "(", ")", or "\"
                    obs-ctext

ccontent        =   ctext / quoted-pair / comment

comment         =   "(" *([FWS] ccontent) [FWS] ")"

CFWS            =   (1*([FWS] comment) [FWS]) / FWS
```

Throughout this specification, where FWS (the folding white space token) appears, it indicates a place where folding, as discussed in section 2.2.3, may take place. Wherever folding appears in a message (that is, a header field body containing a CRLF followed by any WSP), unfolding (removal of the CRLF) is performed before any further semantic analysis is performed on that header field according to this specification. That is to say, any CRLF that appears in FWS is semantically "invisible".

A comment is normally used in a structured field body to provide some human-readable informational text. Since a comment is allowed to contain FWS, folding is permitted within the comment. Also note that since quoted-pair is allowed in a comment, the parentheses and backslash characters may appear in a comment, so long as they appear as a quoted-pair. Semantically, the enclosing parentheses are not part of the comment; the comment is what is contained between the two parentheses. As stated earlier, the "\" in any quoted-pair and the CRLF in any FWS that appears within the comment are semantically "invisible" and therefore not part of the comment either.

Runs of FWS, comment, or CFWS that occur between lexical tokens in a structured header field are semantically interpreted as a single space character.

3.2.3. Atom

Several productions in structured header field bodies are simply strings of certain basic characters. Such productions are called atoms.

Some of the structured header field bodies also allow the period character (".", ASCII value 46) within runs of atext. An additional "dot-atom" token is defined for those purposes.

> Note: The "specials" token does not appear anywhere else in this specification. It is simply the visible (i.e., non-control, non-white space) characters that do not appear in atext. It is provided only because it is useful for implementers who use tools that lexically analyze messages. Each of the characters in specials can be used to indicate a tokenization point in lexical analysis.

```
atext           =   ALPHA / DIGIT /    ; Visible US-ASCII
                    "!" / "#" /        ;  characters not including
                    "$" / "%" /        ;  specials. Used for atoms.
                    "&" / "'" /
                    "*" / "+" /
                    "-" / "/" /
                    "=" / "?" /
                    "^" / "_" /
                    "`" / "{" /
                    "|" / "}" /
                    "~"

atom            =   [CFWS] 1*atext [CFWS]

dot-atom-text   =   1*atext *("." 1*atext)

dot-atom        =   [CFWS] dot-atom-text [CFWS]

specials        =   "(" / ")" /        ; Special characters that do
                    "<" / ">" /        ;  not appear in atext
                    "[" / "]" /
                    ":" / ";" /
                    "@" / "\" /
                    "," / "." /
                    DQUOTE
```

Both atom and dot-atom are interpreted as a single unit, comprising the string of characters that make it up. Semantically, the optional comments and FWS surrounding the rest of the characters are not part of the atom; the atom is only the run of atext characters in an atom, or the atext and "." characters in a dot-atom.

3.2.4. Quoted Strings

Strings of characters that include characters other than those allowed in atoms can be represented in a quoted string format, where the characters are surrounded by quote (DQUOTE, ASCII value 34) characters.

```
qtext           =   %d33 /              ; Visible US-ASCII
                    %d35-91 /           ;  characters not including
                    %d93-126 /          ;  "\" or the quote character
                    obs-qtext

qcontent        =   qtext / quoted-pair

quoted-string   =   [CFWS]
                    DQUOTE *([FWS] qcontent) [FWS] DQUOTE
                    [CFWS]
```

// Erratum 3135 (https://www.rfc-editor.org/errata/eid3135) wanted to
// disallow empty quoted strings. There doesn't appear to be
// consensus for that (e.g., see discussion starting at
// https://mailarchive.ietf.org/arch/msg/ietf-
// 822/9NByCGWq7_dOLRNhrPkZR24074g) and therefore this erratum
// probably should have been rejected.

A quoted-string is treated as a unit. That is, quoted-string is identical to atom, semantically. Since a quoted-string is allowed to contain FWS, folding is permitted. Also note that since quoted-pair is allowed in a quoted-string, the quote and backslash characters may appear in a quoted-string so long as they appear as a quoted-pair.

Semantically, neither the optional CFWS outside of the quote characters nor the quote characters themselves are part of the quoted-string; the quoted-string is what is contained between the two quote characters. As stated earlier, the "\" in any quoted-pair and the CRLF in any FWS/CFWS that appears within the quoted-string are semantically "invisible" and therefore not part of the quoted-string either.

3.2.5. Miscellaneous Tokens

Three additional tokens are defined: word and phrase for combinations of atoms and/or quoted-strings, and unstructured for use in unstructured header fields and in some places within structured header fields.

```
word            =   atom / quoted-string

phrase          =   1*word / obs-phrase

unstructured    =   (*([FWS] VCHAR) *WSP) / obs-unstruct
```

3.3. Date and Time Specification

Date and time values occur in several header fields. This section specifies the syntax for a full date and time specification. Though folding white space is permitted throughout the date-time specification, it is RECOMMENDED that a single space be used in each place that FWS appears (whether it is required or optional); some older implementations will not interpret longer sequences of folding white space correctly.

```
date-time       =   [ day-of-week "," ] date time [CFWS]

day-of-week     =   ([FWS] day-name) / obs-day-of-week

day-name        =   "Mon" / "Tue" / "Wed" / "Thu" /
                    "Fri" / "Sat" / "Sun"

date            =   day month year

day             =   ([FWS] 1*2DIGIT FWS) / obs-day

month           =   "Jan" / "Feb" / "Mar" / "Apr" /
                    "May" / "Jun" / "Jul" / "Aug" /
                    "Sep" / "Oct" / "Nov" / "Dec"

year            =   (FWS 4*DIGIT FWS) / obs-year

time            =   time-of-day zone

time-of-day     =   hour ":" minute [ ":" second ]

hour            =   2DIGIT / obs-hour

minute          =   2DIGIT / obs-minute

second          =   2DIGIT / obs-second

zone            =   (FWS ( "+" / "-" ) 4DIGIT) / obs-zone
```

The day is the numeric day of the month. The year is any numeric year 1900 or later.

The time-of-day specifies the number of hours, minutes, and optionally seconds since midnight of the date indicated (at the offset specified by the zone).

The date and time-of-day SHOULD express local time.

The zone specifies the offset from Coordinated Universal Time (UTC) that the date and time-of-day represent. The "+" or "-" indicates whether the time-of-day is ahead of (i.e., east of) or behind (i.e., west of) Universal Time. The first two digits indicate the number of hours difference from Universal Time, and the last two digits indicate the number of additional minutes difference from Universal Time. (Hence, +hhmm means +(hh * 60 + mm) minutes, and -hhmm means -(hh * 60 + mm) minutes). The form "+0000" SHOULD be used to indicate a time zone at Universal Time. Though "-0000" also indicates Universal Time, it is used to indicate that the time was generated on a system that may be in a local time zone other than Universal Time and that the date-time contains no information about the local time zone.

A date-time specification MUST be semantically valid. That is, the day-of-week (if included) MUST be the day implied by the date, the numeric day-of-month MUST be between 1 and the number of days allowed for the specified month (in the specified year), the time-of-day MUST be in the range 00:00:00 through 23:59:60 (the number of seconds allowing for a leap second; see [RFC3339]), and the last two digits of the zone MUST be within the range 00 through 59.

3.4. Address Specification

Addresses occur in several message header fields to indicate senders and recipients of messages. An address may either be an individual mailbox, or a group of mailboxes.

```
address         =   mailbox / group

mailbox         =   name-addr / addr-spec

name-addr       =   [display-name] angle-addr

angle-addr      =   [CFWS] "<" addr-spec ">" [CFWS] /
                    obs-angle-addr

group           =   display-name ":" [group-list] ";" [CFWS]

display-name    =   phrase

mailbox-list    =   (mailbox *("," mailbox)) / obs-mbox-list

address-list    =   (address *("," address)) / obs-addr-list

group-list      =   mailbox-list / CFWS / obs-group-list
```

A mailbox receives mail. It is a conceptual entity that does not necessarily pertain to file storage. For example, some sites may choose to print mail on a printer and deliver the output to the addressee's desk.

Normally, a mailbox is composed of two parts: (1) an optional display name that indicates the name of the recipient (which can be a person or a system) that could be displayed to the user of a mail application, and (2) an addr-spec address enclosed in angle brackets ("<" and ">"). There is an alternate simple form of a mailbox where the addr-spec address appears alone, without the recipient's name or the angle brackets. The Internet addr-spec address is described in section 3.4.1.

> Note: Some legacy implementations used the simple form where the addr-spec appears without the angle brackets, but included the name of the recipient in parentheses as a comment following the addr-spec. Since the meaning of the information in a comment is unspecified, implementations SHOULD use the full name-addr form of the mailbox, instead of the legacy form, to specify the display name associated with a mailbox. Also, because some legacy implementations interpret the comment, comments generally SHOULD NOT be used in address fields to avoid confusing such implementations.

When it is desirable to treat several mailboxes as a single unit (i.e., in a distribution list), the group construct can be used. The group construct allows the sender to indicate a named group of recipients. This is done by giving a display name for the group, followed by a colon, followed by a comma-separated list of any number of mailboxes (including zero and one), and ending with a semicolon. Because the list of mailboxes can be empty, using the group construct is also a simple way to communicate to recipients that the message was sent to one or more named sets of recipients, without actually providing the individual mailbox address for any of those recipients.

3.4.1. Addr-Spec Specification

An addr-spec is a specific Internet identifier that contains a locally interpreted string followed by the at-sign character ("@", ASCII value 64) followed by an Internet domain. The locally interpreted string is either a quoted-string or a dot-atom. If the string can be represented as a dot-atom (that is, it contains no characters other than atext characters or one or more of "." surrounded by atext characters), then the dot-atom form SHOULD be used and the quoted-string form SHOULD NOT be used. Comments and folding white space SHOULD NOT be used around the "@" in the addr-spec.

> Note: A liberal syntax for the domain portion of addr-spec is given here. However, the domain portion contains addressing information specified by and used in other protocols (e.g., [STD13], [RFC1123], [I-D.klensin-rfc5321bis]). It is therefore incumbent upon implementations to conform to the syntax of addresses for the context in which they are used.

```
addr-spec       =   local-part "@" domain

local-part      =   dot-atom / quoted-string / obs-local-part

domain          =   dot-atom / domain-literal / obs-domain

domain-literal  =   [CFWS] "[" *([FWS] dtext) [FWS] "]" [CFWS]

dtext           =   %d33-90 /          ; Visible US-ASCII
                    %d94-126 /         ;  characters not including
                    obs-dtext          ;  "[", "]", or "\"
```

The domain portion identifies the point to which the mail is delivered. In the dot-atom form, this is interpreted as an Internet domain name (either a host name or a mail exchanger name) as described in [STD13] and [RFC1123]. In the domain-literal form, the domain is interpreted as the literal Internet address of the particular host. In both cases, how addressing is used and how messages are transported to a particular host is covered in separate documents, such as [I-D.klensin-rfc5321bis]. These mechanisms are outside of the scope of this document.

The local-part portion is a domain-dependent string. In addresses, it is simply interpreted on the particular host as a name of a particular mailbox.

3.5. Overall Message Syntax

A message consists of header fields, optionally followed by a message body. Lines in a message MUST be a maximum of 998 characters excluding the CRLF, but it is RECOMMENDED that lines be limited to 78 characters excluding the CRLF. (See section 2.1.1 for explanation.) In a message body, though all of the characters listed in the text rule MAY be used, the use of US-ASCII control characters (values 1 through 8, 11, 12, and 14 through 31) is discouraged since their interpretation by receivers for display is not guaranteed.

```
message         =   (fields / obs-fields)
                    [CRLF body]

body            =   (*(*998text CRLF) *998text) / obs-body

text            =   %d1-9 /             ; Characters excluding CR
                    %d11 /              ;  and LF
                    %d12 /
                    %d14-127
```

The header fields carry most of the semantic information and are defined in section 3.6. The body is simply a series of lines of text that are uninterpreted for the purposes of this specification.

3.6. Field Definitions

The header fields of a message are defined here. All header fields have the same general syntactic structure: a field name, followed by a colon, followed by the field body. The specific syntax for each header field is defined in the subsequent sections.

> Note: In the ABNF syntax for each field in subsequent sections, each field name is followed by the required colon. However, for brevity, sometimes the colon is not referred to in the textual description of the syntax. It is, nonetheless, required.

It is important to note that the header fields are not guaranteed to be in a particular order. They may appear in any order, and they have been known to be reordered occasionally when transported over the Internet. However, for the purposes of this specification, header fields SHOULD NOT be reordered when a message is transported or transformed. More importantly, the trace header fields and resent header fields MUST NOT be reordered, and SHOULD be kept in blocks prepended to the message. See sections 3.6.6 and 3.6.7 for more information.

The only required header fields are the origination date field and the originator address field(s). All other header fields are syntactically optional. More information is contained in the table following this definition.

```
fields          =   *(trace
                      *optional-field /
                      *(resent-date /
                       resent-from /
                       resent-sender /
                       resent-to /
                       resent-cc /
                       resent-bcc /
                       resent-msg-id))
                    *(orig-date /
                    from /
                    sender /
                    reply-to /
                    to /
                    cc /
                    bcc /
                    message-id /
                    in-reply-to /
                    references /
                    subject /
                    comments /
                    keywords /
                    optional-field)
```

// Should there be a 1 in front of the resent fields as per erratum
// 2950 (https://www.rfc-editor.org/errata/eid2950)?

The following table indicates limits on the number of times each field may occur in the header section of a message as well as any special limitations on the use of those fields. An asterisk ("*") next to a value in the minimum or maximum column indicates that a special restriction appears in the Notes column.

Field	Min number	Max number	Notes
trace	0	unlimited	Block prepended - see 3.6.7
resent-date	0*	unlimited*	One per block, required if other resent fields are present - see 3.6.6
resent-from	0	unlimited*	One per block - see 3.6.6

resent-sender	0*	unlimited*	One per block, MUST occur with multi-address resent-from - see 3.6.6
resent-to	0	unlimited*	One per block - see 3.6.6
resent-cc	0	unlimited*	One per block - see 3.6.6
resent-bcc	0	unlimited*	One per block - see 3.6.6
resent-msg-id	0	unlimited*	One per block - see 3.6.6
orig-date	1	1	
from	1	1	See sender and 3.6.2
sender	0*	1	MUST occur with multi-address from - see 3.6.2
reply-to	0	1	
to	0	1	
cc	0	1	
bcc	0	1	
message-id	0*	1	SHOULD be present - see 3.6.4
in-reply-to	0*	1	SHOULD occur in some replies - see 3.6.4
references	0*	1	SHOULD occur in some replies - see 3.6.4
subject	0	1	
comments	0	unlimited	
keywords	0	unlimited	
optional-field	0	unlimited	

Table 1

The exact interpretation of each field is described in subsequent sections.

3.6.1. The Origination Date Field

The origination date field consists of the field name "Date" followed by a date-time specification.

```
orig-date       =   "Date:" date-time CRLF
```

The origination date specifies the date and time at which the creator of the message indicated that the message was complete and ready to enter the mail delivery system. For instance, this might be the time that a user pushes the "send" or "submit" button in an application program. In any case, it is specifically not intended to convey the time that the message is actually transported, but rather the time at which the human or other creator of the message has put the message into its final form, ready for transport. (For example, a portable computer user who is not connected to a network might queue a message for delivery. The origination date is intended to contain the date and time that the user queued the message, not the time when the user connected to the network to send the message.)

3.6.2. Originator Fields

The originator fields of a message consist of the from field, the sender field (when applicable), and optionally the reply-to field. The from field consists of the field name "From" and a comma-separated list of one or more mailbox specifications. If the from field contains more than one mailbox specification in the mailbox-list, then the sender field, containing the field name "Sender" and a single mailbox specification, MUST appear in the message. In either case, an optional reply-to field MAY also be included, which contains the field name "Reply-To" and a comma-separated list of one or more addresses.

```
from            =   "From:" mailbox-list CRLF

sender          =   "Sender:" mailbox CRLF

reply-to        =   "Reply-To:" address-list CRLF
```

The originator fields indicate the mailbox(es) of the source of the message. The "From:" field specifies the author(s) of the message, that is, the mailbox(es) of the person(s) or system(s) responsible for the writing of the message. The "Sender:" field specifies the

mailbox of the agent responsible for the actual transmission of the message. For example, if a secretary were to send a message for another person, the mailbox of the secretary would appear in the "Sender:" field and the mailbox of the actual author would appear in the "From:" field. If the originator of the message can be indicated by a single mailbox and the author and transmitter are identical, the "Sender:" field SHOULD NOT be used. Otherwise, both fields SHOULD appear.

> Note: The transmitter information is always present. The absence of the "Sender:" field is sometimes mistakenly taken to mean that the agent responsible for transmission of the message has not been specified. This absence merely means that the transmitter is identical to the author and is therefore not redundantly placed into the "Sender:" field.

The originator fields also provide the information required when replying to a message. When the "Reply-To:" field is present, it indicates the address(es) to which the author of the message suggests that replies be sent. In the absence of the "Reply-To:" field, replies SHOULD by default be sent to the mailbox(es) specified in the "From:" field unless otherwise specified by the person composing the reply.

In all cases, the "From:" field SHOULD NOT contain any mailbox that does not belong to the author(s) of the message. See also section 3.6.3 for more information on forming the destination addresses for a reply.

3.6.3. Destination Address Fields

The destination fields of a message consist of three possible fields, each of the same form: the field name, which is either "To", "Cc", or "Bcc", followed by a comma-separated list of one or more addresses (either mailbox or group syntax).

```
to              =   "To:" address-list CRLF

cc              =   "Cc:" address-list CRLF

bcc             =   "Bcc:" [address-list / CFWS] CRLF
```

The destination fields specify the recipients of the message. Each destination field may have one or more addresses, and the addresses indicate the intended recipients of the message. The only difference between the three fields is how each is used.

The "To:" field contains the address(es) of the primary recipient(s) of the message.

The "Cc:" field (where the "Cc" means "Carbon Copy" in the sense of making a copy on a typewriter using carbon paper) contains the addresses of others who are to receive the message, though the content of the message may not be directed at them.

The "Bcc:" field (where the "Bcc" means "Blind Carbon Copy") contains addresses of recipients of the message whose addresses are not to be revealed to other recipients of the message. There are three ways in which the "Bcc:" field is used. In the first case, when a message containing a "Bcc:" field is prepared to be sent, the "Bcc:" line is removed even though all of the recipients (including those specified in the "Bcc:" field) are sent a copy of the message. In the second case, recipients specified in the "To:" and "Cc:" lines each are sent a copy of the message with the "Bcc:" line removed as above, but the recipients on the "Bcc:" line get a separate copy of the message containing a "Bcc:" line. (When there are multiple recipient addresses in the "Bcc:" field, some implementations actually send a separate copy of the message to each recipient with a "Bcc:" containing only the address of that particular recipient.) Finally, since a "Bcc:" field may contain no addresses, a "Bcc:" field can be used without any addresses indicating to the recipients that blind copies were sent to someone. Which method to use with "Bcc:" fields is implementation dependent, but refer to the "Security Considerations" section of this document for a discussion of each.

When a message is a reply to another message, the mailboxes of the authors of the original message (the mailboxes in the "From:" field) or mailboxes specified in the "Reply-To:" field (if it exists) MAY appear in the "To:" field of the reply since these would normally be the primary recipients of the reply. If a reply is sent to a message that has destination fields, it is often desirable to send a copy of the reply to all of the recipients of the message, in addition to the author. When such a reply is formed, addresses in the "To:" and "Cc:" fields of the original message MAY appear in the "Cc:" field of the reply, since these are normally secondary recipients of the reply. If a "Bcc:" field is present in the original message, addresses in that field MAY appear in the "Bcc:" field of the reply, but they SHOULD NOT appear in the "To:" or "Cc:" fields.

> Note: Some mail applications have automatic reply commands that include the destination addresses of the original message in the destination addresses of the reply. How those reply commands behave is implementation dependent and is beyond the scope of this document. In particular, whether or not to include the original destination addresses when the original message had a "Reply-To:" field is not addressed here.

3.6.4. Identification Fields

Though listed as optional in the table (Table 1) in section 3.6, every message SHOULD have a "Message-ID:" field. Furthermore, reply messages SHOULD have "In-Reply-To:" and "References:" fields as appropriate and as described below.

The "Message-ID:" field contains a single unique message identifier. The "References:" and "In-Reply-To:" fields each contain one or more unique message identifiers, optionally separated by CFWS.

The message identifier (msg-id) syntax is a limited version of the addr-spec construct enclosed in the angle bracket characters, "<" and ">". Unlike addr-spec, this syntax only permits the dot-atom-text form on the left-hand side of the "@" and does not have internal CFWS anywhere in the message identifier.

> Note: As with addr-spec, a liberal syntax is given for the right-hand side of the "@" in a msg-id. However, later in this section, the use of a domain for the right-hand side of the "@" is RECOMMENDED. Again, the syntax of domain constructs is specified by and used in other protocols (e.g., [STD13], [RFC1123], [I-D.klensin-rfc5321bis]). It is therefore incumbent upon implementations to conform to the syntax of addresses for the context in which they are used.

```
message-id      =   "Message-ID:" msg-id CRLF

in-reply-to     =   "In-Reply-To:" 1*msg-id CRLF

references      =   "References:" 1*msg-id CRLF

msg-id          =   [CFWS] "<" msg-id-internal ">" [CFWS]

msg-id-internal =   id-left "@" id-right

id-left         =   dot-atom-text / obs-id-left

id-right        =   dot-atom-text / no-fold-literal / obs-id-right

no-fold-literal =   "[" *dtext "]"
```

The "Message-ID:" field provides a unique message identifier that refers to a particular version of a particular message. The uniqueness of the message identifier is guaranteed by the host that generates it (see below). This message identifier is intended to be machine readable and not necessarily meaningful to humans. A message identifier pertains to exactly one version of a particular message; subsequent revisions to the message each receive new message identifiers.

> Note: There are many instances when messages are "changed", but those changes do not constitute a new instantiation of that message, and therefore the message would not get a new message identifier. For example, when messages are introduced into the transport system, they are often prepended with additional header fields such as trace fields (described in section 3.6.7) and resent fields (described in section 3.6.6). The addition of such header fields does not change the identity of the message and therefore the original "Message-ID:" field is retained. In all cases, it is the meaning that the sender of the message wishes to convey (i.e., whether this is the same message or a different message) that determines whether or not the "Message-ID:" field changes, not any particular syntactic difference that appears (or does not appear) in the message.

The "In-Reply-To:" and "References:" fields are used when creating a reply to a message. They hold the message identifier of the original message and the message identifiers of other messages (for example, in the case of a reply to a message that was itself a reply). The "In-Reply-To:" field may be used to identify the message (or messages) to which the new message is a reply, while the "References:" field may be used to identify a "thread" of conversation.

When creating a reply to a message, the "In-Reply-To:" and "References:" fields of the resultant message are constructed as follows:

The "In-Reply-To:" field will contain the contents of the "Message-ID:" field of the message to which this one is a reply (the "parent message"). If there is more than one parent message, then the "In-Reply-To:" field will contain the contents of all of the parents' "Message-ID:" fields. If there is no "Message-ID:" field in any of the parent messages, then the new message will have no "In-Reply-To:" field.

The "References:" field will contain the contents of the parent's "References:" field (if any) followed by the contents of the parent's "Message-ID:" field (if any). If the parent message does not contain a "References:" field but does have an "In-Reply-To:" field containing a single message identifier, then the "References:" field will contain the contents of the parent's "In-Reply-To:" field followed by the contents of the parent's "Message-ID:" field (if any). If the parent has none of the "References:", "In-Reply-To:", or "Message-ID:" fields, then the new message will have no "References:" field.

> Note: Some implementations parse the "References:" field to display the "thread of the discussion". These implementations assume that each new message is a reply to a single parent and hence that they can walk backwards through the "References:" field to find the parent of each message listed there. Therefore, trying to form a "References:" field for a reply that has multiple parents is discouraged; how to do so is not defined in this document.

The message identifier (msg-id) itself MUST be a globally unique identifier for a message. The generator of the message identifier MUST guarantee that the msg-id is unique. There are several algorithms that can be used to accomplish this. Since the msg-id has a similar syntax to addr-spec (identical except that quoted strings, comments, and folding white space are not allowed), a good method is to put the domain name (or a domain literal IP address) of the host on which the message identifier was created on the right-hand side of the "@" (since domain names and IP addresses are normally unique), and put a combination of the current absolute date and time along with some other currently unique (perhaps sequential) identifier available on the system (for example, a process id number) on the left-hand side. Though other algorithms will work, it is RECOMMENDED that the right-hand side contain some domain identifier (either of the host itself or otherwise) such that the generator of the message identifier can guarantee the uniqueness of the left-hand side within the scope of that domain.

Semantically, the angle bracket characters are not part of the msg-id; the msg-id is what is contained between the two angle bracket characters.

3.6.5. Informational Fields

The informational fields are all optional. The "Subject:" and "Comments:" fields are unstructured fields as defined in section 2.2.1, and therefore may contain text or folding white space. The "Keywords:" field contains a comma-separated list of one or more words or quoted-strings.

```
subject         =   "Subject:" unstructured CRLF

comments        =   "Comments:" unstructured CRLF

keywords        =   "Keywords:" phrase *("," phrase) CRLF
```

These three fields are intended to have only human-readable content with information about the message. The "Subject:" field is the most common and contains a short string identifying the topic of the message. When used in a reply, the field body MAY start with the string "Re: " (an abbreviation of the Latin "in re", meaning "in the matter of") followed by the contents of the "Subject:" field body of the original message. If this is done, only one instance of the literal string "Re: " ought to be used since use of other strings or more than one instance can lead to undesirable consequences. The "Comments:" field contains any additional comments on the text of the body of the message. The "Keywords:" field contains a comma-separated list of important words and phrases that might be useful for the recipient.

3.6.6. Resent Fields

Resent fields SHOULD be added to any message that is reintroduced by a user into the transport system. A separate set of resent fields SHOULD be added each time this is done. All of the resent fields corresponding to a particular resending of the message SHOULD be grouped together. Each new set of resent fields is prepended to the message; that is, the most recent set of resent fields appears earlier in the message. No other fields in the message are changed when resent fields are added.

Each of the resent fields corresponds to a particular field elsewhere in the syntax. For instance, the "Resent-Date:" field corresponds to the "Date:" field and the "Resent-To:" field corresponds to the "To:" field. In each case, the syntax for the field body is identical to the syntax given previously for the corresponding field.

When resent fields are used, the "Resent-From:" and "Resent-Date:" fields MUST be present. The "Resent-Message-ID:" field SHOULD be present. "Resent-Sender:" SHOULD NOT be used if "Resent-Sender:" would be identical to "Resent-From:".

```
resent-date     =   "Resent-Date:" date-time CRLF

resent-from     =   "Resent-From:" mailbox-list CRLF

resent-sender   =   "Resent-Sender:" mailbox CRLF

resent-to       =   "Resent-To:" address-list CRLF

resent-cc       =   "Resent-Cc:" address-list CRLF

resent-bcc      =   "Resent-Bcc:" [address-list / CFWS] CRLF

resent-msg-id   =   "Resent-Message-ID:" msg-id CRLF
```

Resent fields are used to identify a message as having been reintroduced into the transport system by a user. The purpose of using resent fields is to have the message appear to the final recipient as if it were sent directly by the original sender, with all of the original fields remaining the same. Each set of resent fields correspond to a particular resending event. That is, if a message is resent multiple times, each set of resent fields gives identifying information for each individual time. Resent fields are strictly informational. They MUST NOT be used in the normal processing of replies or other such automatic actions on messages.

> Note: Reintroducing a message into the transport system and using resent fields is a different operation from "forwarding". "Forwarding" has two meanings: One sense of forwarding is that a mail reading program can be told by a user to forward a copy of a message to another person, making the forwarded message the body of the new message. A forwarded message in this sense does not appear to have come from the original sender, but is an entirely new message from the forwarder of the message. Forwarding may also mean that a mail transport program gets a message and forwards it on to a different destination for final delivery. Resent header fields are not intended for use with either type of forwarding.

The resent originator fields indicate the mailbox of the person(s) or system(s) that resent the message. As with the regular originator fields, there are two forms: a simple "Resent-From:" form, which contains the mailbox of the individual doing the resending, and the more complex form, when one individual (identified in the "Resent-Sender:" field) resends a message on behalf of one or more others (identified in the "Resent-From:" field).

> Note: When replying to a resent message, replies behave just as they would with any other message, using the original "From:", "Reply-To:", "Message-ID:", and other fields. The resent fields are only informational and MUST NOT be used in the normal processing of replies.

The "Resent-Date:" indicates the date and time at which the resent message is dispatched by the resender of the message. Like the "Date:" field, it is not the date and time that the message was actually transported.

The "Resent-To:", "Resent-Cc:", and "Resent-Bcc:" fields function identically to the "To:", "Cc:", and "Bcc:" fields, respectively, except that they indicate the recipients of the resent message, not the recipients of the original message.

The "Resent-Message-ID:" field provides a unique identifier for the resent message.

3.6.7. Trace Fields

The trace fields are a group of header fields consisting of an optional "Return-Path:" field, and one or more "Received:" fields. The "Return-Path:" header field contains a pair of angle brackets that enclose an optional addr-spec. The "Received:" field contains a (possibly empty) list of tokens followed by a semicolon and a date-time specification. Each token must be a word, angle-addr, addr-spec, or a domain. Further restrictions are applied to the syntax of the trace fields by specifications that provide for their use, such as [I-D.klensin-rfc5321bis].

```
trace           =   [return]
                    1*received

return          =   "Return-Path:" path CRLF

path            =   angle-addr / ([CFWS] "<" [CFWS] ">" [CFWS])

received        =   "Received:"
                    [1*received-token / CFWS] ";" date-time CRLF

received-token  =   word / angle-addr / addr-spec / domain
```

A full discussion of the Internet mail use of trace fields is contained in [I-D.klensin-rfc5321bis]. For the purposes of this specification, the trace fields are strictly informational, and any formal interpretation of them is outside of the scope of this document.

3.6.8. Optional Fields

Fields may appear in messages that are otherwise unspecified in this document. They MUST conform to the syntax of an optional-field. This is a field name, made up of the visible US-ASCII characters except colon, followed by a colon, followed by any text that conforms to the unstructured syntax.

The field names of any optional field MUST NOT be identical to any field name specified elsewhere in this document.

```
optional-field  =   field-name ":" unstructured CRLF

field-name      =   1*ftext


ftext           =   %d33-57 /          ; Visible US-ASCII
                    %d59-126           ;  characters not including
                                       ;  ":".
```

// Erratum 5918 (https://www.rfc-editor.org/errata/eid5918) basically
// suggests changing field-name to 1*77ftext (leaving room for the
// colon and folding white space). That's probably what was
// intended, but it probably also requires an obs-field-name, and
// there's no indication that the current text has ever caused a
// problem. The editor is ambivalent.

For the purposes of this specification, any optional field is uninterpreted.

4. Obsolete Syntax

Earlier versions of this specification allowed for different (usually more liberal) syntax than is allowed in this version. Also, there have been syntactic elements used in messages on the Internet whose interpretations have never been documented. Though these syntactic forms MUST NOT be generated according to the grammar in section 3, they MUST be accepted and parsed by a conformant receiver. This section documents many of these syntactic elements. Taking the grammar in section 3 and adding the definitions presented in this section will result in the grammar to use for the interpretation of messages.

> Note: This section identifies syntactic forms that any implementation MUST reasonably interpret. However, there are certainly Internet messages that do not conform to even the additional syntax given in this section. The fact that a

> particular form does not appear in any section of this document is not justification for computer programs to crash or for malformed data to be irretrievably lost by any implementation. It is up to the implementation to deal with messages robustly.

One important difference between the obsolete (interpreting) and the current (generating) syntax is that in structured header field bodies (i.e., between the colon and the CRLF of any structured header field), white space characters, including folding white space, and comments could be freely inserted between any syntactic tokens. This allowed many complex forms that have proven difficult for some implementations to parse.

Another key difference between the obsolete and the current syntax is that the rule in section 3.2.2 regarding lines composed entirely of white space in comments and folding white space does not apply. See the discussion of folding white space in section 4.2 below.

Finally, certain characters that were formerly allowed in messages appear in this section. The NUL character (ASCII value 0) was once allowed, but is no longer for compatibility reasons. Similarly, US-ASCII control characters other than CR, LF, SP, and HTAB (ASCII values 1 through 8, 11, 12, 14 through 31, and 127) were allowed to appear in header field bodies. CR and LF were allowed to appear in messages other than as CRLF; this use is also shown here.

Other differences in syntax and semantics are noted in the following sections.

4.1. Miscellaneous Obsolete Tokens

These syntactic elements are used elsewhere in the obsolete syntax or in the main syntax. Bare CR, bare LF, and NUL are added to obs-qp, obs-body, and obs-unstruct. US-ASCII control characters are added to obs-qp, obs-unstruct, obs-ctext, and obs-qtext. The period character is added to obs-phrase. The obs-phrase-list provides for a (potentially empty) comma-separated list of phrases that may include "null" elements. That is, there could be two or more commas in such a list with nothing in between them, or commas at the beginning or end of the list.

> Note: The "period" (or "full stop") character (".") in obs-phrase is not a form that was allowed in earlier versions of this or any other specification. Period (nor any other character from specials) was not allowed in phrase because it introduced a parsing difficulty distinguishing between phrases and portions of an addr-spec (see section 4.4). It appears here because the period character is currently used in many

> messages in the display-name portion of addresses, especially for initials in names, and therefore must be interpreted properly.

```
obs-NO-WS-CTL   =   %d1-8 /            ; US-ASCII control
                    %d11 /             ;  characters that do not
                    %d12 /             ;  include the carriage
                    %d14-31 /          ;  return, line feed, and
                    %d127              ;  white space characters

obs-ctext       =   obs-NO-WS-CTL

obs-qtext       =   obs-NO-WS-CTL

obs-utext       =   %d0 / obs-NO-WS-CTL / VCHAR

obs-qp          =   "\" (%d0 / obs-NO-WS-CTL / LF / CR)

obs-body        =   *(%d0 / LF / CR / text)

obs-unstruct    =   *((*CR 1*(obs-utext / FWS)) / 1*LF) *CR

obs-phrase      =   word *(word / "." / CFWS)

obs-phrase-list =   [phrase / CFWS] *("," [phrase / CFWS])
```

Bare CR and bare LF appear in messages with two different meanings. In many cases, bare CR or bare LF are used improperly instead of CRLF to indicate line separators. In other cases, bare CR and bare LF are used simply as US-ASCII control characters with their traditional ASCII meanings.

4.2. Obsolete Folding White Space

In the obsolete syntax, any amount of folding white space MAY be inserted where the obs-FWS rule is allowed. This creates the possibility of having two consecutive "folds" in a line, and therefore the possibility that a line which makes up a folded header field could be composed entirely of white space.

```
obs-FWS         =   1*([CRLF] WSP)
```

4.3. Obsolete Date and Time

The syntax for the obsolete date format allows a 2 digit year in the date field and allows for a list of alphabetic time zone specifiers that were used in earlier versions of this specification. It also permits comments and folding white space between many of the tokens.

```
obs-day-of-week =   [CFWS] day-name [CFWS]

obs-day         =   [CFWS] 1*2DIGIT [CFWS]

obs-year        =   [CFWS] 2*DIGIT [CFWS]

obs-hour        =   [CFWS] 2DIGIT [CFWS]

obs-minute      =   [CFWS] 2DIGIT [CFWS]

obs-second      =   [CFWS] 2DIGIT [CFWS]

obs-zone        =   "UT" / "GMT" /     ; Universal Time
                                       ; North American UT
                                       ; offsets
                    "EST" / "EDT" /    ; Eastern:  - 5/ - 4
                    "CST" / "CDT" /    ; Central:  - 6/ - 5
                    "MST" / "MDT" /    ; Mountain: - 7/ - 6
                    "PST" / "PDT" /    ; Pacific:  - 8/ - 7
                                       ;
                    %d65-73 /          ; Military zones - "A"
                    %d75-90 /          ; through "I" and "K"
                    %d97-105 /         ; through "Z", both
                    %d107-122          ; upper and lower case
```

Where a two or three digit year occurs in a date, the year is to be interpreted as follows: If a two digit year is encountered whose value is between 00 and 49, the year is interpreted by adding 2000, ending up with a value between 2000 and 2049. If a two digit year is encountered with a value between 50 and 99, or any three digit year is encountered, the year is interpreted by adding 1900.

In the obsolete time zone, "UT" and "GMT" are indications of "Universal Time" and "Greenwich Mean Time", respectively, and are both semantically identical to "+0000".

The remaining three character zones are the US time zones. The first letter, "E", "C", "M", or "P" stands for "Eastern", "Central", "Mountain", and "Pacific". The second letter is either "S" for "Standard" time, or "D" for "Daylight" (daylight saving or summer) time. Their interpretations are as follows:

EDT is semantically equivalent to -0400
EST is semantically equivalent to -0500
CDT is semantically equivalent to -0500
CST is semantically equivalent to -0600
MDT is semantically equivalent to -0600
MST is semantically equivalent to -0700

```
   PDT is semantically equivalent to -0700
   PST is semantically equivalent to -0800
```

The 1 character military time zones were defined in a non-standard way in [RFC0822] and are therefore unpredictable in their meaning. The original definitions of the military zones "A" through "I" are equivalent to "+0100" through "+0900", respectively; "K", "L", and "M" are equivalent to "+1000", "+1100", and "+1200", respectively; "N" through "Y" are equivalent to "-0100" through "-1200". respectively; and "Z" is equivalent to "+0000". However, because of the error in [RFC0822], they SHOULD all be considered equivalent to "-0000" unless there is out-of-band information confirming their meaning.

Other multi-character (usually between 3 and 5) alphabetic time zones have been used in Internet messages. Any such time zone whose meaning is not known SHOULD be considered equivalent to "-0000" unless there is out-of-band information confirming their meaning.

4.4. Obsolete Addressing

There are four primary differences in addressing. First, mailbox addresses were allowed to have a route portion before the addr-spec when enclosed in "<" and ">". The route is simply a comma-separated list of domain names, each preceded by "@", and the list terminated by a colon. Second, CFWS were allowed between the period-separated elements of local-part and domain (i.e., dot-atom was not used). In addition, local-part is allowed to contain quoted-string in addition to just atom. Third, mailbox-list and address-list were allowed to have "null" members. That is, there could be two or more commas in such a list with nothing in between them, or commas at the beginning or end of the list. Finally, US-ASCII control characters and quoted-pairs were allowed in domain literals and are added here.

```
obs-angle-addr  =   [CFWS] "<" obs-route addr-spec ">" [CFWS]

obs-route       =   obs-domain-list ":"

obs-domain-list =   *(CFWS / ",") "@" domain
                    *("," [CFWS] ["@" domain])

obs-mbox-list   =   *([CFWS] ",") mailbox *("," [mailbox / CFWS])

obs-addr-list   =   *([CFWS] ",") address *("," [address / CFWS])

obs-group-list  =   1*([CFWS] ",") [CFWS]

obs-local-part  =   word *("." word)

obs-domain      =   atom *("." atom)

obs-dtext       =   obs-NO-WS-CTL / quoted-pair
```

When interpreting addresses, the route portion SHOULD be ignored.

4.5. Obsolete Header Fields

Syntactically, the primary difference in the obsolete field syntax is that it allows multiple occurrences of any of the fields and they may occur in any order. Also, any amount of white space is allowed before the ":" at the end of the field name.

```
obs-fields      =   *(obs-return /
                      obs-received /
                      obs-orig-date /
                      obs-from /
                      obs-sender /
                      obs-reply-to /
                      obs-to /
                      obs-cc /
                      obs-bcc /
                      obs-message-id /
                      obs-in-reply-to /
                      obs-references /
                      obs-subject /
                      obs-comments /
                      obs-keywords /
                      obs-resent-date /
                      obs-resent-from /
                      obs-resent-send /
                      obs-resent-rply /
                      obs-resent-to /
                      obs-resent-cc /
                      obs-resent-bcc /
                      obs-resent-mid /
                      obs-optional)
```

Except for destination address fields (described in section 4.5.3), the interpretation of multiple occurrences of fields is unspecified. Also, the interpretation of trace fields and resent fields that do not occur in blocks prepended to the message is unspecified as well. Unless otherwise noted in the following sections, interpretation of other fields is identical to the interpretation of their non-obsolete counterparts in section 3.

4.5.1. Obsolete Origination Date Field

```
obs-orig-date   =   "Date" *WSP ":" date-time CRLF
```

4.5.2. Obsolete Originator Fields

```
obs-from        =   "From" *WSP ":" mailbox-list CRLF

obs-sender      =   "Sender" *WSP ":" mailbox CRLF

obs-reply-to    =   "Reply-To" *WSP ":" address-list CRLF
```

4.5.3. Obsolete Destination Address Fields

```
obs-to          =   "To" *WSP ":" address-list CRLF

obs-cc          =   "Cc" *WSP ":" address-list CRLF

obs-bcc         =   "Bcc" *WSP ":"
                    (address-list / (*([CFWS] ",") [CFWS])) CRLF
```

When multiple occurrences of destination address fields occur in a message, they SHOULD be treated as if the address list in the first occurrence of the field is combined with the address lists of the subsequent occurrences by adding a comma and concatenating.

4.5.4. Obsolete Identification Fields

The obsolete "In-Reply-To:" and "References:" fields differ from the current syntax in that they allow phrase (words or quoted strings) to appear. The obsolete forms of the left and right sides of msg-id allow interspersed CFWS, making them syntactically identical to local-part and domain, respectively.

```
obs-message-id  =   "Message-ID" *WSP ":" msg-id CRLF

obs-in-reply-to =   "In-Reply-To" *WSP ":" *(phrase / msg-id) CRLF

obs-references  =   "References" *WSP ":" *(phrase / msg-id) CRLF

obs-id-left     =   local-part

obs-id-right    =   domain
```

For purposes of interpretation, the phrases in the "In-Reply-To:" and "References:" fields are ignored.

Semantically, none of the optional CFWS in the local-part and the domain is part of the obs-id-left and obs-id-right, respectively.

4.5.5. Obsolete Informational Fields

```
obs-subject     =   "Subject" *WSP ":" unstructured CRLF

obs-comments    =   "Comments" *WSP ":" unstructured CRLF

obs-keywords    =   "Keywords" *WSP ":" obs-phrase-list CRLF
```

4.5.6. Obsolete Resent Fields

The obsolete syntax adds a "Resent-Reply-To:" field, which consists of the field name, the optional comments and folding white space, the colon, and a comma separated list of addresses.

```
obs-resent-from =   "Resent-From" *WSP ":" mailbox-list CRLF

obs-resent-send =   "Resent-Sender" *WSP ":" mailbox CRLF

obs-resent-date =   "Resent-Date" *WSP ":" date-time CRLF

obs-resent-to   =   "Resent-To" *WSP ":" address-list CRLF

obs-resent-cc   =   "Resent-Cc" *WSP ":" address-list CRLF

obs-resent-bcc  =   "Resent-Bcc" *WSP ":"
                    (address-list / (*([CFWS] ",") [CFWS])) CRLF

obs-resent-mid  =   "Resent-Message-ID" *WSP ":" msg-id CRLF

obs-resent-rply =   "Resent-Reply-To" *WSP ":" address-list CRLF
```

As with other resent fields, the "Resent-Reply-To:" field is to be treated as trace information only.

4.5.7. Obsolete Trace Fields

The obs-return and obs-received are again given here as template definitions, just as return and received are in section 3. Their full syntax is given in [I-D.klensin-rfc5321bis].

```
obs-return      =   "Return-Path" *WSP ":" path CRLF

obs-received    =   "Received" *WSP ":"
                    [1*received-token / CFWS] [ ";" date-time CRLF ]
```

4.5.8. Obsolete optional fields

```
obs-optional    =   field-name *WSP ":" unstructured CRLF
```

5. Security Considerations

Care needs to be taken when displaying messages on a terminal or terminal emulator. Powerful terminals may act on escape sequences and other combinations of US-ASCII control characters with a variety of consequences. They can remap the keyboard or permit other modifications to the terminal that could lead to denial of service or even damaged data. They can trigger (sometimes programmable) answerback messages that can allow a message to cause commands to be issued on the recipient's behalf. They can also affect the operation of terminal attached devices such as printers. Message viewers may wish to strip potentially dangerous terminal escape sequences from the message prior to display. However, other escape sequences appear in messages for useful purposes (cf. [ISO.2022.1994], [RFC2045], [RFC2046], [RFC2047], [RFC2049], [BCP13]) and therefore should not be stripped indiscriminately.

Transmission of non-text objects in messages raises additional security issues. These issues are discussed in [RFC2045], [RFC2046], [RFC2047], [RFC2049], [BCP13].

Many implementations use the "Bcc:" (blind carbon copy) field, described in section 3.6.3, to facilitate sending messages to recipients without revealing the addresses of one or more of the addressees to the other recipients. Mishandling this use of "Bcc:" may disclose confidential information that could eventually lead to security problems through knowledge of even the existence of a particular mail address. For example, if using the first method described in section 3.6.3, where the "Bcc:" line is removed from the message, blind recipients have no explicit indication that they have been sent a blind copy, except insofar as their address does not appear in the header section of a message. Because of this, one of the blind addressees could potentially send a reply to all of the shown recipients and accidentally reveal that the message went to the blind recipient. When the second method from section 3.6.3 is used, the blind recipient's address appears in the "Bcc:" field of a separate copy of the message. If the "Bcc:" field contains all of the blind addressees, all of the "Bcc:" recipients will be seen by each "Bcc:" recipient. Even if a separate message is sent to each "Bcc:" recipient with only the individual's address, implementations still need to be careful to process replies to the message as per section 3.6.3 so as not to accidentally reveal the blind recipient to other recipients.

6. IANA Considerations

This document updates the registrations that appeared in [RFC4021] that referred to the definitions in [RFC2822]. IANA is requested to update the Permanent Message Header Field Repository with the following header fields, in accordance with the procedures set out in [RFC3864].

Header field name Date
Applicable protocol Mail
Status standard
Author/Change controller IETF
Specification document(s) This document (section 3.6.1)

Header field name From
Applicable protocol Mail
Status standard
Author/Change controller IETF
Specification document(s) This document (section 3.6.2)

Header field name Sender
Applicable protocol Mail
Status standard
Author/Change controller IETF
Specification document(s) This document (section 3.6.2)

Header field name Reply-To
Applicable protocol Mail
Status standard
Author/Change controller IETF
Specification document(s) This document (section 3.6.2)

Header field name To
Applicable protocol Mail
Status standard
Author/Change controller IETF
Specification document(s) This document (section 3.6.3)

Header field name Cc
Applicable protocol Mail
Status standard
Author/Change controller IETF
Specification document(s) This document (section 3.6.3)

Header field name Bcc
Applicable protocol Mail
Status standard
Author/Change controller IETF

Specification document(s) This document (section 3.6.3)

Header field name Message-ID
Applicable protocol Mail
Status standard
Author/Change controller IETF
Specification document(s) This document (section 3.6.4)

Header field name In-Reply-To
Applicable protocol Mail
Status standard
Author/Change controller IETF
Specification document(s) This document (section 3.6.4)

Header field name References
Applicable protocol Mail
Status standard
Author/Change controller IETF
Specification document(s) This document (section 3.6.4)

Header field name Subject
Applicable protocol Mail
Status standard
Author/Change controller IETF
Specification document(s) This document (section 3.6.5)

Header field name Comments
Applicable protocol Mail
Status standard
Author/Change controller IETF
Specification document(s) This document (section 3.6.5)

Header field name Keywords
Applicable protocol Mail
Status standard
Author/Change controller IETF
Specification document(s) This document (section 3.6.5)

Header field name Resent-Date
Applicable protocol Mail
Status standard
Author/Change controller IETF
Specification document(s) This document (section 3.6.6)

Header field name Resent-From
Applicable protocol Mail
Status standard
Author/Change controller IETF

Specification document(s) This document (section 3.6.6)

Header field name Resent-Sender
Applicable protocol Mail
Status standard
Author/Change controller IETF
Specification document(s) This document (section 3.6.6)

Header field name Resent-To
Applicable protocol Mail
Status standard
Author/Change controller IETF
Specification document(s) This document (section 3.6.6)

Header field name Resent-Cc
Applicable protocol Mail
Status standard
Author/Change controller IETF
Specification document(s) This document (section 3.6.6)

Header field name Resent-Bcc
Applicable protocol Mail
Status standard
Author/Change controller IETF
Specification document(s) This document (section 3.6.6)

Header field name Resent-Reply-To
Applicable protocol Mail
Status obsolete
Author/Change controller IETF
Specification document(s) This document (section 4.5.6)

Header field name Resent-Message-ID
Applicable protocol Mail
Status standard
Author/Change controller IETF
Specification document(s) This document (section 3.6.6)

Header field name Return-Path
Applicable protocol Mail
Status standard
Author/Change controller IETF
Specification document(s) This document (section 3.6.7)

Header field name Received
Applicable protocol Mail
Status standard
Author/Change controller IETF

Specification document(s) This document (section 3.6.7)
Related information [I-D.klensin-rfc5321bis]

7. References

7.1. Normative References

[ANSI.X3-4.1986]
American National Standards Institute, "Coded Character Set - 7-bit American Standard Code for Information Interchange", ANSI X3.4, 1986.

[BCP14] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017.

<https://www.rfc-editor.org/info/bcp14>

[RFC1123] 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>.

[STD13] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

<https://www.rfc-editor.org/info/std13>

[STD68] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

<https://www.rfc-editor.org/info/std68>

7.2. Informative References

[BCP13] Freed, N. and J. Klensin, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", BCP 13, RFC 4289, December 2005.

Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, January 2013.

<https://www.rfc-editor.org/info/bcp13>

[ISO.2022.1994]
International Organization for Standardization, "Information technology - Character code structure and extension techniques", ISO Standard 2022, 1994.

[RFC0822] 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>.

[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>.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/info/rfc2046>.

[RFC2047] 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>.

[RFC2049] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples", RFC 2049, DOI 10.17487/RFC2049, November 1996, <https://www.rfc-editor.org/info/rfc2049>.

[RFC2822] Resnick, P., Ed., "Internet Message Format", RFC 2822, DOI 10.17487/RFC2822, April 2001, <https://www.rfc-editor.org/info/rfc2822>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, DOI 10.17487/RFC3864, September 2004, <https://www.rfc-editor.org/info/rfc3864>.

[RFC4021] Klyne, G. and J. Palme, "Registration of Mail and MIME Header Fields", RFC 4021, DOI 10.17487/RFC4021, March 2005, <https://www.rfc-editor.org/info/rfc4021>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/info/rfc5322>.

[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>.

[I-D.klensin-rfc5321bis] Klensin, J., "Simple Mail Transfer Protocol", Work in Progress, Internet-Draft, draft-klensin-rfc5321bis-02, 27 December 2019, <https://tools.ietf.org/html/draft-klensin-rfc5321bis-02>.

Appendix A. Example Messages

This section presents a selection of messages. These are intended to assist in the implementation of this specification, but should not be taken as normative; that is to say, although the examples in this section were carefully reviewed, if there happens to be a conflict between these examples and the syntax described in sections 3 and 4 of this document, the syntax in those sections is to be taken as correct.

In the text version of this document, messages in this section are delimited between lines of "----". The "----" lines are not part of the message itself.

A.1. Addressing Examples

The following are examples of messages that might be sent between two individuals.

A.1.1. A Message from One Person to Another with Simple Addressing

This could be called a canonical message. It has a single author, John Doe, a single recipient, Mary Smith, a subject, the date, a message identifier, and a textual message in the body.

```
From: John Doe <jdoe@machine.example>
To: Mary Smith <mary@example.net>
Subject: Saying Hello
Date: Fri, 21 Nov 1997 09:55:06 -0600
Message-ID: <1234@local.machine.example>

This is a message just to say hello.
So, "Hello".
```

If John's secretary Michael actually sent the message, even though John was the author and replies to this message should go back to him, the sender field would be used:

```
From: John Doe <jdoe@machine.example>
Sender: Michael Jones <mjones@machine.example>
To: Mary Smith <mary@example.net>
Subject: Saying Hello
Date: Fri, 21 Nov 1997 09:55:06 -0600
Message-ID: <1234@local.machine.example>

This is a message just to say hello.
So, "Hello".
```

A.1.2. Different Types of Mailboxes

This message includes multiple addresses in the destination fields and also uses several different forms of addresses.

```
From: "Joe Q. Public" <john.q.public@example.com>
To: Mary Smith <mary@x.test>, jdoe@example.org, Who? <one@y.test>
Cc: <boss@nil.test>, "Giant; \"Big\" Box" <sysservices@example.net>
Date: Tue, 1 Jul 2003 10:52:37 +0200
Message-ID: <5678.21-Nov-1997@example.com>

Hi everyone.
```

Note that the display names for Joe Q. Public and Giant; "Big" Box needed to be enclosed in double-quotes because the former contains the period and the latter contains both semicolon and double-quote characters (the double-quote characters appearing as quoted-pair constructs). Conversely, the display name for Who? could appear without them because the question mark is legal in an atom. Notice also that jdoe@example.org and boss@nil.test have no display names associated with them at all, and jdoe@example.org uses the simpler address form without the angle brackets.

A.1.3. Group Addresses

```
From: Pete <pete@silly.example>
To: A Group:Ed Jones <c@a.test>,joe@where.test,John <jdoe@one.test>;
Cc: Undisclosed recipients:;
Date: Thu, 13 Feb 1969 23:32:54 -0330
Message-ID: <testabcd.1234@silly.example>

Testing.
```

In this message, the "To:" field has a single group recipient named "A Group", which contains 3 addresses, and a "Cc:" field with an empty group recipient named Undisclosed recipients.

A.2. Reply Messages

The following is a series of three messages that make up a conversation thread between John and Mary. John first sends a message to Mary, Mary then replies to John's message, and then John replies to Mary's reply message.

Note especially the "Message-ID:", "References:", and "In-Reply-To:" fields in each message.

```
From: John Doe <jdoe@machine.example>
To: Mary Smith <mary@example.net>
Subject: Saying Hello
Date: Fri, 21 Nov 1997 09:55:06 -0600
Message-ID: <1234@local.machine.example>

This is a message just to say hello.
So, "Hello".
```

When sending replies, the Subject field is often retained, though prepended with "Re: " as described in section 3.6.5.

```
From: Mary Smith <mary@example.net>
To: John Doe <jdoe@machine.example>
Reply-To: "Mary Smith: Personal Account" <smith@home.example>
Subject: Re: Saying Hello
Date: Fri, 21 Nov 1997 10:01:10 -0600
Message-ID: <3456@example.net>
In-Reply-To: <1234@local.machine.example>
References: <1234@local.machine.example>

This is a reply to your hello.
```

Note the "Reply-To:" field in the above message. When John replies to Mary's message above, the reply should go to the address in the "Reply-To:" field instead of the address in the "From:" field.

```
To: "Mary Smith: Personal Account" <smith@home.example>
From: John Doe <jdoe@machine.example>
Subject: Re: Saying Hello
Date: Fri, 21 Nov 1997 11:00:00 -0600
Message-ID: <abcd.1234@local.machine.test>
In-Reply-To: <3456@example.net>
References: <1234@local.machine.example> <3456@example.net>

This is a reply to your reply.
```

A.3. Resent Messages

Start with the message that has been used as an example several times:

```
From: John Doe <jdoe@machine.example>
To: Mary Smith <mary@example.net>
Subject: Saying Hello
Date: Fri, 21 Nov 1997 09:55:06 -0600
Message-ID: <1234@local.machine.example>

This is a message just to say hello.
So, "Hello".
```

Say that Mary, upon receiving this message, wishes to send a copy of the message to Jane such that (a) the message would appear to have come straight from John; (b) if Jane replies to the message, the reply should go back to John; and (c) all of the original information, like the date the message was originally sent to Mary, the message identifier, and the original addressee, is preserved. In this case, resent fields are prepended to the message:

```
Resent-From: Mary Smith <mary@example.net>
Resent-To: Jane Brown <j-brown@other.example>
Resent-Date: Mon, 24 Nov 1997 14:22:01 -0800
Resent-Message-ID: <78910@example.net>
From: John Doe <jdoe@machine.example>
To: Mary Smith <mary@example.net>
Subject: Saying Hello
Date: Fri, 21 Nov 1997 09:55:06 -0600
Message-ID: <1234@local.machine.example>

This is a message just to say hello.
So, "Hello".
```

If Jane, in turn, wished to resend this message to another person, she would prepend her own set of resent header fields to the above and send that. (Note that for brevity, trace fields are not shown.)

A.4. Messages with Trace Fields

As messages are sent through the transport system as described in [I-D.klensin-rfc5321bis], trace fields are prepended to the message. The following is an example of what those trace fields might look like. Note that there is some folding white space in the first one since these lines can be long.

```
Received: from x.y.test
   by example.net
   via TCP
   with ESMTP
   id ABC12345
   for <mary@example.net>;  21 Nov 1997 10:05:43 -0600
Received: from node.example by x.y.test; 21 Nov 1997 10:01:22 -0600
From: John Doe <jdoe@node.example>
To: Mary Smith <mary@example.net>
Subject: Saying Hello
Date: Fri, 21 Nov 1997 09:55:06 -0600
Message-ID: <1234@local.node.example>

This is a message just to say hello.
So, "Hello".
```

A.5. White Space, Comments, and Other Oddities

White space, including folding white space, and comments can be inserted between many of the tokens of fields. Taking the example from A.1.3, white space and comments can be inserted into all of the fields.

```
From: Pete(A nice \) chap) <pete@silly.test(his host is silly)>
To:A Group(Some people)
     :Chris Jones <c@public.example(.host of Chris)>,
         joe@example.org,
  John <jdoe@one.test> (my dear friend); (the end of the group)
Cc:(Empty list)(start)Hidden recipients  :(nobody(that I know))  ;
Date: Thu,
      13
        Feb
          1969
      23:32
               -0330 (Newfoundland Time)
Message-ID:             <testabcd.1234@silly.test>

Testing.
```

The above example is aesthetically displeasing, but perfectly legal. Note particularly (1) the comments in the "From:" field (including one that has a ")" character appearing as part of a quoted-pair); (2) the white space absent after the ":" in the "To:" field as well as the comment and folding white space after the group name, the special character (".") in the comment in Chris Jones's address, and the folding white space before and after "joe@example.org,"; (3) the multiple and nested comments in the "Cc:" field as well as the comment immediately following the ":" after "Cc"; (4) the folding white space (but no comments except at the end) and the missing seconds in the time of the date field; and (5) the white space before (but not within) the identifier in the "Message-ID:" field.

A.6. Obsoleted Forms

The following are examples of obsolete (that is, the "MUST NOT generate") syntactic elements described in section 4 of this document.

A.6.1. Obsolete Addressing

Note in the example below the lack of quotes around Joe Q. Public, the route that appears in the address for Mary Smith, the two commas that appear in the "To:" field, and the spaces that appear around the "." in the jdoe address.

```
From: Joe Q. Public <john.q.public@example.com>
To: Mary Smith <@node.test:mary@example.net>, , jdoe@test  . example
Date: Tue, 1 Jul 2003 10:52:37 +0200
Message-ID: <5678.21-Nov-1997@example.com>

Hi everyone.
```

A.6.2. Obsolete Dates

The following message uses an obsolete date format, including a non-numeric time zone and a two digit year. Note that although the day-of-week is missing, that is not specific to the obsolete syntax; it is optional in the current syntax as well.

```
From: John Doe <jdoe@machine.example>
To: Mary Smith <mary@example.net>
Subject: Saying Hello
Date: 21 Nov 97 09:55:06 GMT
Message-ID: <1234@local.machine.example>

This is a message just to say hello.
So, "Hello".
```

A.6.3. Obsolete White Space and Comments

White space and comments can appear between many more elements than in the current syntax. Also, folding lines that are made up entirely of white space are legal.

```
From  : John Doe <jdoe@machine(comment).  example>
To    : Mary Smith
__
          <mary@example.net>
Subject     : Saying Hello
Date  : Fri, 21 Nov 1997 09(comment):   55  :  06 -0600
Message-ID  : <1234   @   local(blah)  .machine .example>

This is a message just to say hello.
So, "Hello".
```

Note especially the second line of the "To:" field. It starts with two space characters. (Note that "__" represent blank spaces.) Therefore, it is considered part of the folding, as described in section 4.2. Also, the comments and white space throughout addresses, dates, and message identifiers are all part of the obsolete syntax.

Appendix B. Differences from Earlier Specifications

This appendix contains a list of changes that have been made in the Internet Message Format from earlier specifications, specifically [RFC0822], [RFC1123], [RFC2822], and [RFC5322]. Items marked with an asterisk (*) below are items which appear in section 4 of this document and therefore can no longer be generated.

The following are the changes made from [RFC0822] and [RFC1123] to [RFC2822] that remain in this document:

1. Period allowed in obsolete form of phrase.
2. ABNF moved out of document, now in [STD68].
3. Four or more digits allowed for year.
4. Header field ordering (and lack thereof) made explicit.
5. Encrypted header field removed.
6. Specifically allow and give meaning to "-0000" time zone.
7. Folding white space is not allowed between every token.
8. Requirement for destinations removed.
9. Forwarding and resending redefined.
10. Extension header fields no longer specifically called out.
11. ASCII 0 (null) removed.*
12. Folding continuation lines cannot contain only white space.*
13. Free insertion of comments not allowed in date.*

14. Non-numeric time zones not allowed.*
15. Two digit years not allowed.*
16. Three digit years interpreted, but not allowed for generation.*
17. Routes in addresses not allowed.*
18. CFWS within local-parts and domains not allowed.*
19. Empty members of address lists not allowed.*
20. Folding white space between field name and colon not allowed.*
21. Comments between field name and colon not allowed.
22. Tightened syntax of in-reply-to and references.*
23. CFWS within msg-id not allowed.*
24. Tightened semantics of resent fields as informational only.
25. Resent-Reply-To not allowed.*
26. No multiple occurrences of fields (except resent and received).*
27. Free CR and LF not allowed.*
28. Line length limits specified.
29. Bcc more clearly specified.

The following are changes from [RFC2822].

1. Assorted typographical/grammatical errors fixed and clarifications made.
2. Changed "standard" to "document" or "specification" throughout.
3. Made distinction between "header field" and "header section".
4. Removed NO-WS-CTL from ctext, qtext, dtext, and unstructured.*
5. Moved discussion of specials to the "Atom" section. Moved text to "Overall message syntax" section.
6. Simplified CFWS syntax.
7. Fixed unstructured syntax (erratum 373 (https://www.rfc-editor.org/errata/eid373)).
8. Changed date and time syntax to deal with white space in obsolete date syntax.
9. Removed quoted-pair from domain literals and message identifiers.*
10. Clarified that other specifications limit domain syntax.
11. Simplified "Bcc:" and "Resent-Bcc:" syntax.
12. Allowed optional-field to appear within trace information.
13. Removed no-fold-quote from msg-id. Clarified syntax limitations.
14. Generalized "Received:" syntax to fix bugs and move definition out of this document.
15. Simplified obs-qp. Fixed and simplified obs-utext (which now only appears in the obsolete syntax). Removed obs-text and obs-char, adding obs-body.
16. Fixed obsolete date syntax to allow for more (or less) comments and white space.
17. Fixed all obsolete list syntax (obs-domain-list, obs-mbox-list, obs-addr-list, obs-phrase-list, and the newly added obs-group-list).

18. Fixed obs-reply-to syntax.
19. Fixed obs-bcc and obs-resent-bcc to allow empty lists.
20. Removed obs-path.

The following are changes from [RFC5322].

1. Clarified addr-spec description (erratum 1766 (https://www.rfc-editor.org/errata/eid1766)).
2. Fixed obs-unstruct to be more limited (erratum 1905 (https://www.rfc-editor.org/errata/eid1905)).*
3. Simplified obs-body (erratum 1906 (https://www.rfc-editor.org/errata/eid1906)).*
4. Fixed obs-FWS to allow for a leading CRLF (erratum 1908 (https://www.rfc-editor.org/errata/eid1908)).*
5. Fixed comments within addresses in A.5 (errata 2515 (https://www.rfc-editor.org/errata/eid2515) and 2579 (https://www.rfc-editor.org/errata/eid2579)).
6. Fixed time zone description (erratum 2726 (https://www.rfc-editor.org/errata/eid2726)).
7. Removed inappropriate uses of "sent" in 3.6.3, 3.6.6, and 5 (erratum 3048 (https://www.rfc-editor.org/errata/eid3048)).
8. Allow for CFWS in otherwise empty list of "Received:" field tokens (erratum 3979 (https://www.rfc-editor.org/errata/eid3979)).
9. Changed "printable" to "visible" to clarify that it doesn't include the space character (erratum 4692 (https://www.rfc-editor.org/errata/eid4692)).
10. Clarify midnight in time-of-day (erratum 5905 (https://www.rfc-editor.org/errata/eid5905)).
11. Allow for date-time in obs-received (erratum 5867 (https://www.rfc-editor.org/errata/eid5867)).*
12. Separated out "msg-id-internal" in "msg-id".
13. Updated references to STD 13, STD 68, BCP 13, and BCP 14, and reference for leap seconds to RFC 3339.
14. Fixed typo in daylight saving time in description of obs-zone.*

There are also 3 errata that were "Held For Document Update" that have not been addressed. See the comments in the following document sections:

1. Erratum 2950: Section 3.6
2. Erratum 3135: Section 3.2.4
3. Erratum 5918: Section 3.6.8

Appendix C. Acknowledgements

Many people contributed to this document. They included folks who participated in the Detailed Revision and Update of Messaging Standards (DRUMS) Working Group of the Internet Engineering Task Force (IETF), the chair of DRUMS, the Area Directors of the IETF, reporters of errata on earlier versions of this document, and people who simply sent their comments in via email. The editor is deeply indebted to them all and thanks them sincerely. (The list of these people has been temporarily removed to try to bring it up to date.)

Author's Address

Peter W. Resnick (editor)
Episteme Technology Consulting LLC
503 West Indiana Avenue
Urbana, IL 61801-4941
United States of America

Phone: +1 217 337 1905
Email: resnick@episteme.net
URI: https://www.episteme.net/