Network Working Group
Request for Comments: 2821
Obsoletes: 821, 974, 1869
Updates: 1123
Category: Standards Track

J. Klensin, Editor
AT&T Laboratories

April 2001

Simple Mail Transfer Protocol

Status of this Memo

Copyright Notice

Abstract

This document is a self-contained specification of the basic protocol for the Internet electronic mail transport. It consolidates, updates and clarifies, but doesn’t add new or change existing functionality of the following:

- the original SMTP (Simple Mail Transfer Protocol) specification of RFC 821 [30],

- domain name system requirements and implications for mail transport from RFC 1035 [22] and RFC 974 [27],

- the clarifications and applicability statements in RFC 1123 [2], and

- material drawn from the SMTP Extension mechanisms [19].

It obsoletes RFC 821, RFC 974, and updates RFC 1123 (replaces 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 POP [3, 26] and IMAP [6]. Additional submission issues are discussed in RFC 2476 [15].

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 [32] discusses message headers, message bodies and formats and structures for them, and their relationship.

Table of Contents

1. Introduction 4
2. The SMTP Model 5
2.1 Basic Structure 5
2.2 The Extension Model 7
2.2.1 Background 7
2.2.2 Definition and Registration of Extensions 8
2.3 Terminology 9
2.3.1 Mail Objects 10
2.3.2 Senders and Receivers 10
2.3.3 Mail Agents and Message Stores 10
2.3.4 Host 11
2.3.5 Domain 11
2.3.6 Buffer and State Table 11
2.3.7 Lines 12
2.3.8 Originator, Delivery, Relay, and Gateway Systems 12
2.3.9 Message Content and Mail Data 13
2.3.10 Mailbox and Address 13
2.3.11 Reply 13
2.4 General Syntax Principles and Transaction Model 13
3. The SMTP Procedures: An Overview 15
3.1 Session Initiation 15
3.2 Client Initiation 16
3.3 Mail Transactions 16
3.4 Forwarding for Address Correction or Updating 19

3.5 Commands for Debugging Addresses 20
3.5.1 Overview 20
3.5.2 VRFY Normal Response 22
3.5.3 Meaning of VRFY or EXPN Success Response 22
3.5.4 Semantics and Applications of EXPN 23
3.6 Domains 23
3.7 Relaying 24
3.8 Mail Gatewaying 25
3.8.1 Header Fields in Gatewaying 26
3.8.2 Received Lines in Gatewaying 26
3.8.3 Addresses in Gatewaying 26
3.8.4 Other Header Fields in Gatewaying 27
3.8.5 Envelopes in Gatewaying 27
3.9 Terminating Sessions and Connections 27
3.10 Mailing Lists and Aliases 28
3.10.1 Alias 28
3.10.2 List 28
4. The SMTP Specifications 29
4.1 SMTP Commands 29
4.1.1 Command Semantics and Syntax 29
4.1.1.1 Extended HELLO (EHLO) or HELLO (HELO) 29
4.1.1.2 MAIL (MAIL) 31
4.1.1.3 RECIPIENT (RCPT) 31
4.1.1.4 DATA (DATA) 33
4.1.1.5 RESET (RSET) 34
4.1.1.6 VERIFY (VRFY) 35
4.1.1.7 EXPAND (EXPN) 35
4.1.1.8 HELP (HELP) 35
4.1.1.9 NOOP (NOOP) 35
4.1.1.10 QUIT (QUIT) 36
4.1.2 Command Argument Syntax 36
4.1.3 Address Literals 38
4.1.4 Order of Commands 39
4.1.5 Private-use Commands 40
4.2 SMTP Replies 40
4.2.1 Reply Code Severities and Theory 42
4.2.2 Reply Codes by Function Groups 44
4.2.3 Reply Codes in Numeric Order 45
4.2.4 Reply Code 502 46
4.2.5 Reply Codes After DATA and the Subsequent <CRLF>.<CRLF> 46
4.3 Sequencing of Commands and Replies 47
4.3.1 Sequencing Overview 47
4.3.2 Command-Reply Sequences 48
4.4 Trace Information 49
4.5 Additional Implementation Issues 53
4.5.1 Minimum Implementation 53
4.5.2 Transparency 53
4.5.3 Sizes and Timeouts 54

4.5.3.1 Size limits and minimums 54
4.5.3.2 Timeouts 56
4.5.4 Retry Strategies 57
4.5.4.1 Sending Strategy 58
4.5.4.2 Receiving Strategy 59
4.5.5 Messages with a null reverse-path 59
5. Address Resolution and Mail Handling 60
6. Problem Detection and Handling 62
6.1 Reliable Delivery and Replies by Email 62
6.2 Loop Detection 63
6.3 Compensating for Irregularities 63
7. Security Considerations 64
7.1 Mail Security and Spoofing 64
7.2 "Blind" Copies 65
7.3 VRFY, EXPN, and Security 65
7.4 Information Disclosure in Announcements 66
7.5 Information Disclosure in Trace Fields 66
7.6 Information Disclosure in Message Forwarding 67
7.7 Scope of Operation of SMTP Servers 67
8. IANA Considerations 67
9. References 68
10. Editor's Address 70
11. Acknowledgments 70
Appendices 71
A. TCP Transport Service 71
B. Generating SMTP Commands from RFC 822 Headers 71
C. Source Routes 72
D. Scenarios 73
E. Other Gateway Issues 76
F. Deprecated Features of RFC 821 76
Full Copyright Statement 79

1. Introduction

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 describe some of them.

An important feature of SMTP is its capability to transport mail across networks, usually referred to as "SMTP mail relaying" (see section 3.8). 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

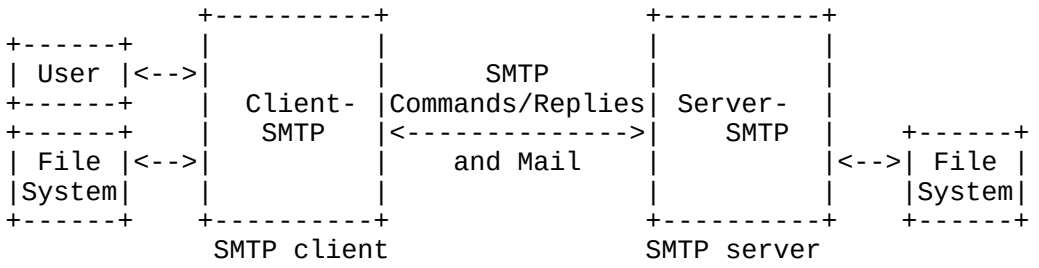
SMTP, a process can transfer mail to another process on the same network or to some other network via a relay or gateway process accessible to both networks.

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 [22, 27] (and section 5 of this document) are used to identify the appropriate next-hop destination for a message being transported.

2. The SMTP Model

2.1 Basic Structure

The SMTP design can be pictured as:



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

The means by which a mail message is presented to an SMTP client, and how that client determines the domain name(s) to which mail messages are to be transferred is a local matter, and is not addressed by this document. In some cases, the domain name(s) transferred to, or 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 [3, 26] or IMAP [6] protocols, or when the SMTP client is inside an isolated transport service environment, the domain name 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 domain names 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.

The means by which an SMTP client, once it has determined a target domain name, determines the identity of an SMTP server to which a copy of a message is to be transferred, and then performs that transfer, is 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, a formal handoff of responsibility for the message occurs: the protocol requires that a server accept responsibility for either delivering a message or properly reporting the failure to do so.

Once the transmission channel is established and initial handshaking 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 headers 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 extension requests such as command pipelining [13].

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. This transmission normally occurs 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. 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.

Usually, intermediate hosts are determined via the DNS MX record, not by explicit "source" routing (see section 5 and appendices C and F.2).

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 [33].

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;

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

- 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.

2.3 Terminology

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 below.

1. MUST This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.

2. MUST NOT This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.

3. SHOULD This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

4. SHOULD NOT This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

5. MAY This word, or the adjective "OPTIONAL", mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which

does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

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 recipient address specification command (RCPT TO) could be used to specify alternate delivery modes, such as immediate display; those variations have now been deprecated (see appendix F, section F.6).

The SMTP content is sent in the SMTP DATA protocol unit and has two parts: the headers and the body. If the content conforms to other contemporary standards, the headers form a collection of field/value pairs structured as in the message format specification [32]; the body, if structured, is defined according to MIME [12]. The content is textual in nature, expressed using the US-ASCII repertoire [1]. Although SMTP extensions (such as "8BITMIME" [20]) may relax this restriction for the content body, the content headers are always encoded using the US-ASCII repertoire. A MIME extension [23] defines 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 "domain"); identifying them by numerical address is discouraged.

2.3.5 Domain

A domain (or domain name) consists of one or more dot-separated components. These components ("labels" in DNS terminology [22]) are restricted for SMTP purposes to consist of a sequence of letters, digits, and hyphens drawn from the ASCII character set [1]. 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 [22] and section 5 of this specification.

The domain name, as described in this document and in [22], 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.

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 which 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 Lines

SMTP commands and, unless altered by a service extension, message data, are transmitted in "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 4.5.3).

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.

2.3.8 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 which 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 [11]).

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 message headers and the possibly-structured message body. The MIME specification [12] provides the standard mechanisms for structured message bodies.

2.3.10 Mailbox and Address

As used in this specification, an "address" is a character string that identifies a user to whom mail will be sent or a location into which mail will be deposited. The term "mailbox" refers to that depository. The two terms are typically used interchangeably unless the distinction between the location in which mail is placed (the mailbox) and a reference to it (the address) is important. An address normally consists of user and domain specifications. The standard mailbox naming convention is defined to be "local-part@domain": contemporary usage permits a much broader set of applications than simple "user names". Consequently, and due to a long history of problems when intermediate hosts have attempted to optimize transport by modifying them, the local-part MUST be interpreted and assigned semantics only by the host specified in the domain part of the address.

2.3.11 Reply

An SMTP reply is an acknowledgment (positive or negative) sent 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. Recent work [34] has specified further structuring of the reply strings, including the use of supplemental and more specific completion codes.

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 MUST follow 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. This is NOT true of a mailbox local-part. 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. Mailbox domains are not case sensitive. 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.

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 field 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 [1]. 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 seven bit transport only. An originating SMTP client which has not successfully negotiated an appropriate extension with a particular server 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 ultimate delivery of a severely garbled message to the recipient. Delivery SMTP systems MAY reject ("bounce") such messages rather than deliver them. No sending SMTP system is 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.

Eight-bit message content transmission MAY be requested of the server by a client using extended SMTP facilities, notably the "8BITMIME" extension [20]. 8BITMIME SHOULD be supported by SMTP servers. However, it MUST not be construed as authorization to transmit unrestricted eight bit material. 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 [8]. Metalanguage terms used in running text are surrounded by pointed brackets (e.g., <CRLF>) for clarity.

3. The SMTP Procedures: An Overview

This section contains descriptions of the procedures used in SMTP: session initiation, the mail transaction, forwarding mail, verifying mailbox names and expanding mailing lists, and the 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 4.5.1).

The SMTP protocol allows a server to formally reject a transaction 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 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 which are unable to support service extensions and contemporary clients which 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 which 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> can 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.

```
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). This step of the procedure can be repeated any number of times.

The <forward-path> can contain more than just a mailbox. Historically, the <forward-path> can be 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 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).

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), or 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. 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 RFC 822 format [7, 32] is being used, the mail data include the memo header items such as Date, Subject, To, Cc, From. Server SMTP systems SHOULD NOT reject messages based on perceived defects in the RFC 822 or MIME [12] message header or message body. In particular,

they MUST NOT reject messages in which the numbers of Resent-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 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.

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. But, 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 or bounce messages when they are not deliverable when 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. But, 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 are strongly encouraged to provide configuration mechanisms so that sites which 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 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 response 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 [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 normally includes the mailbox name, i.e., "<local-part@domain>", where "domain" is a fully qualified domain name, MUST appear in the syntax. 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. 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, they MUST be listed as service extensions in an EHLO response, if they are supported.

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 discussed 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 bounced 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 Domains

Only resolvable, fully-qualified, domain names (FQDNs) are permitted when domain names are used in SMTP. In other words, names that can be resolved to MX RRs or A RRs (as discussed in section 5) are permitted, as are CNAME RRs whose targets can be resolved, in turn, to MX or A RRs. 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 A RR) or, if the host has no name, an address literal as described in section 4.1.1.1.

- 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.

3.7 Relaying

In general, the availability of Mail eXchanger records in the domain name system [22, 27] makes the use of explicit source routes in the Internet mail system unnecessary. Many historical problems with their interpretation 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 are also permitted to 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.

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.

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.

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, and work is underway on standardized mail submission protocols that might eventually supercede the current practices. 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 which act as gateways into other environments, not just SMTP relays and final delivery systems; see sections 3.8 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, [24, 25]) 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 2.4.1, a relay SMTP has no need to inspect or act upon the headers or body of the message data and MUST NOT do so except to add its own "Received:" header (section 4.4) and, optionally, to attempt to detect looping in the mail system (see section 6.2).

3.8 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 2.3.8, 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.

3.8.1 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 2.4.1.

Other mail systems gatewayed to the Internet often use a subset of RFC 822 headers 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. 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).

3.8.2 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.

"Received:" 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 fields arising in non-SMTP environments, receiving systems MUST NOT reject mail based on the format of a trace field and SHOULD be extremely robust in the light of unexpected information or formats in those fields.

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

3.8.3 Addresses in Gatewaying

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

3.8.4 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., fields MUST be transformed (if necessary) to satisfy RFC 822 syntax, 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 return path from the SMTP envelope, not to the sender listed in the "From:" field (or other fields) of the RFC 822 message.

3.8.5 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.

3.9 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 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 response code. This response 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).

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 which is forcibly shut down via external means SHOULD attempt to send a line containing a 421 response code to the SMTP client before exiting. The SMTP client will normally read the 421 response 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 451 response had been received and act accordingly.

3.10 Mailing Lists and Aliases

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 [32] MUST be left unchanged; in particular, the "From" field of the message header 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.

3.10.1 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.

3.10.2 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 all of the expanded

addresses. The return 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.

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 are encouraged to 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 which 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 which 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 field 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), optionally followed by information that will help to identify the client system. y 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 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 keyworks 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 [8], is:

```
ehlo-ok-rsp  =    ( "250"    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*(%d33-127)
               ; any CHAR excluding <SP> and all
               ; control characters (US-ASCII 0-31 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 2.4.1.

4.1.1.2 MAIL (MAIL)

This command is used to initiate a mail transaction in which the mail data is delivered to an SMTP server which may, in turn, deliver it to one or more mailboxes or pass it on to another system (possibly using SMTP). The argument field 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 nondelivery notifications), the reverse-path may be null (see section 3.7).

This command clears the reverse-path buffer, the forward-path buffer, and the mail data buffer; and inserts the reverse-path information from this command 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 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 use of this command. The argument field 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.

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>
```

Of course, 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 TO:" ("<Postmaster@" domain ">" / "<Postmaster>" / Forward-Path)
                [SP Rcpt-parameters] CRLF
```

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 is terminated by a line containing only a period, that is, the character sequence "<CRLF>.<CRLF>" (see section 4.5.2). This is the end of mail data indication. Note that 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 data, ends the DATA command itself. 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 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" 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., if 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" 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" 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 may be issued at any time.

Syntax:
"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 may be issued at any time.

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

Syntax:
"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 an OK reply.

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

Syntax:
"NOOP" [SP String] CRLF

4.1.1.10 QUIT (QUIT)

This command specifies that the receiver MUST send an 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 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.

Syntax:
"QUIT" CRLF

4.1.2 Command Argument Syntax

The syntax of the argument fields of the above commands (using the syntax specified in [8] where applicable) is given below. Some of the productions given below are used only in conjunction with source routes as described in appendix C. Terminals not defined in this document, such as ALPHA, DIGIT, SP, CR, LF, CRLF, are as defined in the "core" syntax [8 (section 6)] or in the message format syntax [32].

```
Reverse-path = Path
Forward-path = Path
Path = "<" [ A-d-l ":" ] Mailbox ">"
A-d-l = At-domain *( "," A-d-l )
      ; 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 / %d62-127)
      ; any CHAR excluding "=", SP, and control characters
Keyword  = Ldh-str
Argument = Atom
Domain = (sub-domain 1*("." sub-domain)) / address-literal
sub-domain = Let-dig [Ldh-str]

address-literal = "[" IPv4-address-literal /
                      IPv6-address-literal /
                      General-address-literal "]"
      ; See section 4.1.3

Mailbox = Local-part "@" Domain

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

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

Atom = 1*atext

Quoted-string = DQUOTE *qcontent DQUOTE

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 field with the comma being the fourth character of the field.

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, RFC1035 [22]), characters outside the set of alphas, 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.

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 IPv6 standards [17].

Specifically:

```
IPv4-address-literal = Snum 3("." Snum)
IPv6-address-literal = "IPv6:" IPv6-addr
General-address-literal = Standardized-tag ":" 1*dcontent
Standardized-tag = Ldh-str
      ; MUST be specified in a standards-track RFC
      ; and registered with IANA

Snum = 1*3DIGIT  ; representing a decimal integer
      ; value in the range 0 through 255
Let-dig = ALPHA / DIGIT
Ldh-str = *( ALPHA / DIGIT / "-" ) Let-dig

IPv6-addr = IPv6-full / IPv6-comp / IPv6v4-full / IPv6v4-comp
IPv6-hex  = 1*4HEXDIG
IPv6-full = IPv6-hex 7(":" IPv6-hex)
IPv6-comp = [IPv6-hex *5(":" IPv6-hex)] "::" [IPv6-hex *5(":"
           IPv6-hex)]
      ; The "::" represents at least 2 16-bit groups of zeros
      ; No more than 6 groups in addition to the "::" may be
      ; present
IPv6v4-full = IPv6-hex 5(":" IPv6-hex) ":" IPv4-address-literal
IPv6v4-comp = [IPv6-hex *3(":" IPv6-hex)] "::"
```

```
                [IPv6-hex *3(":" IPv6-hex) ":"] IPv4-address-literal
          ; The "::" represents at least 2 16-bit groups of zeros
          ; No more than 4 groups in addition to the "::" and
          ; IPv4-address-literal may be present
```

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 or EXPN) without this initialization.

An EHLO command MAY be issued by a client later in the session. If it is issued after the session begins, 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, or 502 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 valid principal host name (not a CNAME or MX name) for its host. 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 and supplemental information provided that will assist in identifying the client.

An SMTP server MAY verify that the domain name parameter in the EHLO command actually corresponds to the IP address of the client. However, the server MUST NOT refuse to accept a message for this reason if the verification fails: the information about verification failure is for logging and tracing only.

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) 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 (or a new EHLO) 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 it the previous one successfully concluded with a successful DATA command, or if the previous one was aborted with a RSET.

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 cannot be used at any other time in a session, but 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 section 4.2.1). Since, in violation of this specification, the text is sometimes not sent, clients which 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 [ SP text ] CRLF
Reply-line = Reply-code [ SP text ] CRLF
```

where "Greeting" appears only in the 220 response that announces that the server is opening its part of the connection.

An SMTP server SHOULD send only the reply codes listed in this document. 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 receiver-SMTP SHOULD test the first digit (severity indication) of the reply code.

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 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.

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 five values for the first digit of the reply code:

1yz Positive Preliminary reply
The command has been accepted, but the requested action is being held in abeyance, pending confirmation of the information in this reply. The SMTP client should send another command specifying whether to continue or abort the action. Note: unextended SMTP does not have any commands that allow this type of reply, and so does not have continue or abort commands.

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 is encouraged to 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 is discouraged from repeating 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).

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:

```
123-First line
123-Second line
123-234 text beginning with numbers
123 The last line
```

In many cases the SMTP client then simply needs to search for a line beginning with the reply code followed by <SP> or <CRLF> and ignore all preceding lines. 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)
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)

      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)
      450 Requested mail action not taken: mailbox unavailable
         (e.g., mailbox busy)
      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")
```

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)
451 Requested action aborted: local error in processing
452 Requested action not taken: insufficient system storage
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)
503 Bad sequence of commands
504 Command parameter not implemented
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")

4.2.4 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.

4.2.5 Reply Codes After DATA and the Subsequent <CRLF>.<CRLF>

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

- delivering the message (if the recipient mailbox exists), or
- if attempts to deliver the message fail due to transient conditions, retrying delivery some reasonable number of times at intervals as specified in section 4.5.4.

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

When an SMTP server returns a permanent error status (5yz) code after the DATA command is completed with <CRLF>.<CRLF>, it MUST NOT make any subsequent attempt to deliver that message. The SMTP client retains responsibility for 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. I.e., 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 completely 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 and intervention of the message.

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 4.1.3 for a discussion of alternatives in these situations.

For example,

```
   220 ISIF.USC.EDU Service ready
or
   220 mail.foo.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 meaning and action implied by the code numbers and by the specific command reply sequence cannot be altered.

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.

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.

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: 554
EHLO or HELO
   S: 250
   E: 504, 550
MAIL
   S: 250
   E: 552, 451, 452, 550, 553, 503
RCPT
   S: 250, 251 (but see section 3.4 for discussion of 251 and 551)
   E: 550, 551, 552, 553, 450, 451, 452, 503, 550
DATA
   I: 354 -> data -> S: 250
                     E: 552, 554, 451, 452
   E: 451, 554, 503
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 ("time stamp" or "Received") information at the beginning of the message content, as discussed in section 4.1.1.4.

This line MUST be structured as follows:

- The FROM field, 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 field MAY contain an "@" as suggested in RFC 822, but this is not required.

- The FOR field MAY contain a list of <path> entries when multiple RCPT commands have been given. This may raise some security issues and is usually not desirable; see section 7.2.

An Internet mail program MUST NOT change a Received: line that was previously added to the message header. 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 fields is important for detecting problems, especially slow relays. SMTP servers that create Received fields SHOULD use explicit offsets in the dates (e.g., -0800), rather than time zone names of any type. Local time (with an offset) is preferred to 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 mail data headers and body [32].

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. SMTP servers performing a relay function MUST NOT inspect the message data, and especially not to the extent needed to determine if Return-path headers are present. SMTP servers making final delivery MAY remove Return-path headers 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 (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, 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 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 is preferred when possible. All undeliverable mail notification messages are sent using the MAIL command (even if they result from processing the obsolete SEND, SOML, or SAML commands) and use a null return path as discussed in section 3.7.

The time stamp line and the return path line are formally defined as follows:

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

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

Stamp = From-domain By-domain Opt-info ";"  FWS date-time

      ; where "date-time" is as defined in [32]
      ; but the "obs-" forms, especially two-digit
      ; years, are prohibited in SMTP and MUST NOT be used.

From-domain = "FROM" FWS Extended-Domain CFWS

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

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]

Via = "VIA" FWS Link CFWS

With = "WITH" FWS Protocol CFWS

ID = "ID" FWS String / msg-id CFWS

For = "FOR" FWS 1*( Path / Mailbox ) CFWS

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
Attdl-Protocol = Atom
      ; Additional standard names for protocols are registered with the
         ; Internet Assigned Numbers Authority (IANA).  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 is required for 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 which 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 which 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 3.7 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 [16] 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 which impose no limits on the length of these objects should be used.

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

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

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

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

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

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

message content
The maximum total length of a message content (including any message headers as well as the message body) MUST BE at least 64K octets. Since the introduction of Internet standards for multimedia mail [12], 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 [18], and SMTP client systems that will send large messages SHOULD utilize it when possible.

recipients 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 servers MUST NOT, and delivery SMTP servers SHOULD NOT, perform validation tests on message headers suggests that rejecting a message based on the total number of recipients shown in header fields is to be discouraged. A server which imposes a limit on the number of recipients MUST behave in an orderly fashion, such as to reject 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.

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.

RFC 821 [30] 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 recipients 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 which received 452 responses, at least 100 of these will be able to fit in the SMTP server's recipients buffer. Each retransmission attempt which 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 5XX response code. This would be most appropriate if the policy limitation was intended to apply if the total recipient count for a particular message body were enforced even if that message body was sent in multiple mail transactions.

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:

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.

MAIL Command: 5 minutes

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.

DATA Initiation: 2 minutes
This is while awaiting the "354 Start Input" reply to a DATA command.

Data Block: 3 minutes
This is while awaiting the completion of each TCP SEND call transmitting a chunk of data.

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.

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. 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 [9].

The strategy may be further modified as a result of multiple addresses per host (see below) to optimize delivery time vs. 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. Implementation of this efficiency feature is strongly encouraged.

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 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 which 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) [24], and also Message Disposition Notifications (MDNs) [10]. 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 with a valid, non-null reverse-path.

Implementors of automated email processors should be careful to make sure that the various kinds of messages with null reverse-path are handled correctly, in particular such systems SHOULD NOT reply to messages with null reverse-path.

5. Address Resolution and Mail Handling

Once an SMTP client lexically identifies a domain to which mail will be delivered for processing (as described in sections 3.6 and 3.7), a DNS lookup MUST be performed to resolve the domain name [22]. 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 and, due to a history of problems, are generally discouraged. The lookup first attempts to locate an MX record associated with the name. If a CNAME record is found instead, the resulting name is processed as if it were the initial name. If no MX records are found, but an A RR is found, the A RR is treated as if it was associated with an implicit MX RR, with a preference of 0, pointing to that host. If one or more MX RRs are found for a given name, SMTP systems MUST NOT utilize any A 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 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 is used to rank the host addresses: multiple MX records, and multihomed hosts.

Multiple MX records contain a preference indication that MUST be used in sorting (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 SMTP 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.

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.

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 [28] for a discussion of this problem.

6.2 Loop Detection

Simple counting of the number of "Received:" headers 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.

6.3 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 [3, 26, 5, 21]. These protocols have encouraged the use of SMTP as a posting 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 [7]). 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.

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 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 of actions performed by the servers (in trace fields and/or header comments) is strongly encouraged.

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 which use digital signatures (see [14] and, e.g., PGP [4] or S/MIME [31]).

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, unless they are accompanied by careful handoffs of responsibility in a carefully-designed trust environment, they remain inherently weaker than end-to-end mechanisms which 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 or in which error (or normal) replies should be directed to a special address. (Systems that provide convenient ways for users to alter these fields on a per-message basis should attempt to establish a primary and permanent mailbox address for the user so that Sender 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 an ignorant user who is trying to fake mail.

7.2 "Blind" Copies

Addresses that do not appear in the message headers 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 headers, either as part of trace headers or as informational or private-extension headers. 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 headers. Receiving systems SHOULD NOT attempt to deduce such relationships and use them to alter the headers of the message for delivery. The popular "Apparently-to" header 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. 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.

Within the last few years, 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. Implementations SHOULD still provide support for EXPN, but sites SHOULD carefully evaluate the tradeoffs. As authentication mechanisms are introduced into SMTP, some sites may choose to make EXPN available only to authenticated requestors.

7.4 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 are strongly encouraged to minimally provide for making type and version information available in some way to other network hosts.

7.5 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") 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.

7.6 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.

7.7 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, MAIL, or RCPT as appropriate.

8. IANA Considerations

IANA will maintain three registries in support of this specification. The first 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 consists of "tags" that identify forms of domain literals other than those for IPv4 addresses (specified in RFC 821 and in this document) and IPv6 addresses (specified in this document). Additional literal types require standardization before being used; none are anticipated at this time.

The third, 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")

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.

9. References

[1] American National Standards Institute (formerly United States of America Standards Institute), X3.4, 1968, "USA Code for Information Interchange". ANSI X3.4-1968 has been replaced by newer versions with slight modifications, but the 1968 version remains definitive for the Internet.

[2] Braden, R., "Requirements for Internet hosts - application and support", STD 3, RFC 1123, October 1989.

[3] Butler, M., Chase, D., Goldberger, J., Postel, J. and J. Reynolds, "Post Office Protocol - version 2", RFC 937, February 1985.

[4] Callas, J., Donnerhacke, L., Finney, H. and R. Thayer, "OpenPGP Message Format", RFC 2440, November 1998.

[5] Crispin, M., "Interactive Mail Access Protocol - Version 2", RFC 1176, August 1990.

[6] Crispin, M., "Internet Message Access Protocol - Version 4", RFC 2060, December 1996.

[7] Crocker, D., "Standard for the Format of ARPA Internet Text Messages", RFC 822, August 1982.

[8] Crocker, D. and P. Overell, Eds., "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[9] De Winter, J., "SMTP Service Extension for Remote Message Queue Starting", RFC 1985, August 1996.

[10] Fajman, R., "An Extensible Message Format for Message Disposition Notifications", RFC 2298, March 1998.

[11] Freed, N, "Behavior of and Requirements for Internet Firewalls", RFC 2979, October 2000.

[12] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, December 1996.

[13] Freed, N., "SMTP Service Extension for Command Pipelining", RFC 2920, September 2000.

[14] Galvin, J., Murphy, S., Crocker, S. and N. Freed, "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted", RFC 1847, October 1995.

[15] Gellens, R. and J. Klensin, "Message Submission", RFC 2476, December 1998.

[16] Kille, S., "Mapping between X.400 and RFC822/MIME", RFC 2156, January 1998.

[17] Hinden, R and S. Deering, Eds. "IP Version 6 Addressing Architecture", RFC 2373, July 1998.

[18] Klensin, J., Freed, N. and K. Moore, "SMTP Service Extension for Message Size Declaration", STD 10, RFC 1870, November 1995.

[19] Klensin, J., Freed, N., Rose, M., Stefferud, E. and D. Crocker, "SMTP Service Extensions", STD 10, RFC 1869, November 1995.

[20] Klensin, J., Freed, N., Rose, M., Stefferud, E. and D. Crocker, "SMTP Service Extension for 8bit-MIMEtransport", RFC 1652, July 1994.

[21] Lambert, M., "PCMAIL: A distributed mail system for personal computers", RFC 1056, July 1988.

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

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

[23] Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, December 1996.

[24] Moore, K., "SMTP Service Extension for Delivery Status Notifications", RFC 1891, January 1996.

[25] Moore, K., and G. Vaudreuil, "An Extensible Message Format for Delivery Status Notifications", RFC 1894, January 1996.

[26] Myers, J. and M. Rose, "Post Office Protocol - Version 3", STD 53, RFC 1939, May 1996.

[27] Partridge, C., "Mail routing and the domain system", RFC 974, January 1986.

[28] Partridge, C., "Duplicate messages and SMTP", RFC 1047, February 1988.

[29] Postel, J., ed., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", STD 7, RFC 793, September 1981.

[30] Postel, J., "Simple Mail Transfer Protocol", RFC 821, August 1982.

[31] Ramsdell, B., Ed., "S/MIME Version 3 Message Specification", RFC 2633, June 1999.

[32] Resnick, P., Ed., "Internet Message Format", RFC 2822, April 2001.

[33] Vaudreuil, G., "SMTP Service Extensions for Transmission of Large and Binary MIME Messages", RFC 1830, August 1995.

[34] Vaudreuil, G., "Enhanced Mail System Status Codes", RFC 1893, January 1996.

10. Editor’s Address

John C. Klensin
AT&T Laboratories
99 Bedford St
Boston, MA 02111 USA

Phone: 617-574-3076
EMail: klensin@research.att.com

11. Acknowledgments

Many people worked long and hard on the many iterations of this document. There was wide-ranging debate in the IETF DRUMS Working Group, both on its mailing list and in face to face discussions, about many technical issues and the role of a revised standard for Internet mail transport, and many contributors helped form the wording in this specification. The hundreds of participants in the many discussions since RFC 821 was produced are too numerous to mention, but they all helped this document become what it is.

APPENDICES

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, but not in SMTP commands or responses.

B. Generating SMTP Commands from RFC 822 Headers

Some systems use RFC 822 headers (only) in a mail submission protocol, or otherwise generate SMTP commands from RFC 822 headers 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 a mail envelopes is not separated early in processing from header 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 fields SHOULD then be removed from the headers. Once this process is completed, the remaining headers SHOULD be checked to verify that at least one To:, Cc:, or Bcc: header remains. If none do, then a bcc: header with no additional information SHOULD be inserted as specified in [32].

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 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 headers 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-only remailer, loops back to the Internet environment (and the mailing list) are almost inevitable.

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> SHOULD be the host sending the MAIL command. 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; while servers MUST be prepared to receive and handle them as discussed in section 3.3 and F.2, clients SHOULD NOT transmit them and this section was included only to provide context.

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

If source routes are used, RFC 821 and the text below should be consulted for the mechanisms for constructing and updating the forward- and reverse-paths.

The SMTP server transforms the command arguments by moving its own identifier (its domain name or that of any domain for which it is acting as a mail exchanger), if it appears, from the forward-path to the beginning of the reverse-path.

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. Conversely, SMTP servers MUST NOT derive final message delivery information from message header fields.

When the list of hosts is present, 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. As each 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).

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, 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

```
      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:<@foo.com: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

```
      S: 220 xyz.com Simple Mail Transfer Service Ready
      C: EHLO foo.com
      S: 250 xyz.com is on the air
      C: MAIL FROM:<@foo.com: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 05:33:22 -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
```

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: SEND 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
```

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 envelope information from one system to the message headers or body of another) have generally proven to be inadequate in important ways. Systems translating between environments that do not support both envelopes and headers and Internet mail must be written with the understanding that some information loss is almost inevitable.

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.

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 is strongly preferred to 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 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.

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.

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.