Network Working Group
Request for Comments: 2822
Obsoletes: 822
Category: Standards Track

P. Resnick, Editor
QUALCOMM Incorporated

April 2001

Internet Message Format

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2001). All Rights Reserved.

Abstract

This standard specifies a syntax for text messages that are sent between computer users, within the framework of "electronic mail" messages. This standard supersedes the one specified in Request For Comments (RFC) 822, "Standard for the Format of ARPA Internet Text Messages", updating it to reflect current practice and incorporating incremental changes that were specified in other RFCs.

Table of Contents

1. Introduction 3
1.1. Scope 3
1.2. Notational conventions 4
1.2.1. Requirements notation 4
1.2.2. Syntactic notation 4
1.3. Structure of this document 4
2. Lexical Analysis of Messages 5
2.1. General Description 5
2.1.1. Line Length Limits 6
2.2. Header Fields 7
2.2.1. Unstructured Header Field Bodies 7
2.2.2. Structured Header Field Bodies 7
2.2.3. Long Header Fields 7
2.3. Body 8
3. Syntax 9
3.1. Introduction 9
3.2. Lexical Tokens 9

LG Elecs. Ex. 1009
LG Elecs. v. Pantech Corp.
IPR2023-01271 Page 1

3.2.1. Primitive Tokens 9
3.2.2. Quoted characters 10
3.2.3. Folding white space and comments 11
3.2.4. Atom 12
3.2.5. Quoted strings 13
3.2.6. Miscellaneous tokens 13
3.3. Date and Time Specification 14
3.4. Address Specification 15
3.4.1. Addr-spec specification 16
3.5 Overall message syntax 17
3.6. Field definitions 18
3.6.1. The origination date field 20
3.6.2. Originator fields 21
3.6.3. Destination address fields 22
3.6.4. Identification fields 23
3.6.5. Informational fields 26
3.6.6. Resent fields 26
3.6.7. Trace fields 28
3.6.8. Optional fields 29
4. Obsolete Syntax 29
4.1. Miscellaneous obsolete tokens 30
4.2. Obsolete folding white space 31
4.3. Obsolete Date and Time 31
4.4. Obsolete Addressing 33
4.5. Obsolete header fields 33
4.5.1. Obsolete origination date field 34
4.5.2. Obsolete originator fields 34
4.5.3. Obsolete destination address fields 34
4.5.4. Obsolete identification fields 35
4.5.5. Obsolete informational fields 35
4.5.6. Obsolete resent fields 35
4.5.7. Obsolete trace fields 36
4.5.8. Obsolete optional fields 36
5. Security Considerations 36
6. Bibliography 37
7. Editor's Address 38
8. Acknowledgements 39
Appendix A. Example messages 41
A.1. Addressing examples 41
A.1.1. A message from one person to another with simple
addressing 41
A.1.2. Different types of mailboxes 42
A.1.3. Group addresses 43
A.2. Reply messages 43
A.3. Resent messages 44
A.4. Messages with trace fields 46
A.5. White space, comments, and other oddities 47
A.6. Obsoleted forms 47

A.6.1. Obsolete addressing 48
A.6.2. Obsolete dates 48
A.6.3. Obsolete white space and comments 48
Appendix B. Differences from earlier standards 49
Appendix C. Notices 50
Full Copyright Statement 51

1. Introduction

1.1. Scope

This standard specifies a syntax for text messages that are sent between computer users, within the framework of "electronic mail" messages. This standard supersedes the one specified in Request For Comments (RFC) 822, "Standard for the Format of ARPA Internet Text Messages" [RFC822], updating it to reflect current practice and incorporating incremental changes that were specified in other RFCs [STD3].

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

In the context of electronic mail, messages are viewed as having an envelope and contents. The envelope contains whatever information is needed to accomplish transmission and delivery. (See [RFC2821] for a discussion of the envelope.) The contents comprise the object to be delivered to the recipient. This standard applies only to the format and some of the semantics of message contents. It contains no specification of the information in the envelope.

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

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

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

1.2. Notational conventions

1.2.1. Requirements notation

This document occasionally uses terms that appear in capital letters. When the terms "MUST", "SHOULD", "RECOMMENDED", "MUST NOT", "SHOULD NOT", and "MAY" appear capitalized, they are being used to indicate particular requirements of this specification. A discussion of the meanings of these terms appears in [RFC2119].

1.2.2. Syntactic notation

This standard uses the Augmented Backus-Naur Form (ABNF) notation specified in [RFC2234] for the formal definitions of the syntax of messages. Characters will be specified either by a decimal value (e.g., the value %d65 for uppercase A and %d97 for lowercase A) or by a case-insensitive literal value enclosed in quotation marks (e.g., "A" for either uppercase or lowercase A). See [RFC2234] for the full description of the notation.

1.3. Structure of this document

This document is divided into several sections.

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

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

Section 3 specifies formal ABNF rules for the structure of each part of a message (the syntax) and describes the relationship between those parts and their meaning in the context of a message (the semantics). That is, it describes the actual rules for the structure of each part of a message (the syntax) as well as a description of the parts and instructions on how they ought to be interpreted (the semantics). This includes analysis of the syntax and semantics of

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

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

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

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

Section 6 is a bibliography of references in this document.

Section 7 contains the editor's address.

Section 8 contains acknowledgements.

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

Appendix B lists the differences between this standard and earlier standards for Internet messages.

Appendix C has copyright and intellectual property notices.

2. Lexical Analysis of Messages

2.1. General Description

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

Note: This standard specifies that messages are made up of characters in the US-ASCII range of 1 through 127. There are other documents, specifically the MIME document series [RFC2045, RFC2046, RFC2047, RFC2048, RFC2049], that extend this standard to allow for values outside of that range. Discussion of those mechanisms is not within the scope of this standard.

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

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

2.1.1. Line Length Limits

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

The 998 character limit is due to limitations in many implementations which send, receive, or store Internet Message Format messages that simply cannot handle more than 998 characters on a line. Receiving implementations would do well to handle an arbitrarily large number of characters in a line for robustness sake. However, there are so many implementations which (in compliance with the transport requirements of [RFC2821]) do not accept messages containing more than 1000 character including the CR and LF per line, it is important for implementations not to create such messages.

The more conservative 78 character recommendation is to accommodate the many implementations of user interfaces that display these messages which may truncate, or disastrously wrap, the display of more than 78 characters per line, in spite of the fact that such implementations are non-conformant to the intent of this specification (and that of [RFC2821] if they actually cause information to be lost). Again, even though this limitation is put on messages, it is encumbant upon implementations which display messages

to handle an arbitrarily large number of characters in a line (certainly at least up to the 998 character limit) for the sake of robustness.

2.2. Header Fields

Header fields are lines composed of a field name, followed by a colon (":"), followed by a field body, and terminated by CRLF. A field name MUST be composed of printable US-ASCII characters (i.e., characters that have values between 33 and 126, inclusive), except colon. A field body may be composed of any US-ASCII characters, except for CR and LF. However, a field body may contain CRLF when used in header "folding" and "unfolding" as described in section 2.2.3. All field bodies MUST conform to the syntax described in sections 3 and 4 of this standard.

2.2.1. Unstructured Header Field Bodies

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

2.2.2. Structured Header Field Bodies

Some field bodies in this standard have specific syntactical structure more restrictive than the unstructured field bodies described above. These are referred to as "structured" field bodies. Structured field bodies are sequences of specific lexical tokens as described in sections 3 and 4 of this standard. Many of these tokens are allowed (according to their syntax) to be introduced or end with comments (as described in section 3.2.3) as well as the space (SP, ASCII value 32) and horizontal tab (HTAB, ASCII value 9) characters (together known as the white space characters, WSP), and those WSP characters are subject to header "folding" and "unfolding" as described in section 2.2.3. Semantic analysis of structured field bodies is given along with their syntax.

2.2.3. Long Header Fields

Each header field is logically a single line of characters comprising the field name, the colon, and the field body. For convenience however, and to deal with the 998/78 character limitations per line, the field body portion of a header field can be split into a multiple line representation; this is called "folding". The general rule is

that wherever this standard allows for folding white space (not simply WSP characters), a CRLF may be inserted before any WSP. For example, the header field:

```
        Subject: This is a test
```

can be represented as:

```
        Subject: This
         is a test
```

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

The process of moving from this folded multiple-line representation of a header field to its single line representation is called "unfolding". Unfolding is accomplished by simply removing any CRLF that is immediately followed by WSP. Each header field should be treated in its unfolded form for further syntactic and semantic evaluation.

2.3. Body

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

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

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

Note: As was stated earlier, there are other standards documents, specifically the MIME documents [RFC2045, RFC2046, RFC2048, RFC2049] that extend this standard to allow for different sorts of message bodies. Again, these mechanisms are beyond the scope of this document.

3. Syntax

3.1. Introduction

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

For the defined expressions, a short description of the syntax and use is given, followed by the syntax in ABNF, followed by a semantic analysis. Primitive tokens that are used but otherwise unspecified come from [RFC2234].

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

3.2. Lexical Tokens

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

Note: Readers of this standard need to pay special attention to how these lexical tokens are used in both the lower-level and higher-level syntax later in the document. Particularly, the white space tokens and the comment tokens defined in section 3.2.3 get used in the lower-level tokens defined here, and those lower-level tokens are in turn used as parts of the higher-level tokens defined later. Therefore, the white space and comments may be allowed in the higher-level tokens even though they may not explicitly appear in a particular definition.

3.2.1. Primitive Tokens

The following are primitive tokens referred to elsewhere in this standard, but not otherwise defined in [RFC2234]. Some of them will not appear anywhere else in the syntax, but they are convenient to refer to in other parts of this document.

Note: The "specials" below are just such an example. Though the specials token does not appear anywhere else in this standard, it is useful for implementers who use tools that lexically analyze messages. Each of the characters in specials can be used to indicate a tokenization point in lexical analysis.

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

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

specials        =       "(" / ")" /     ; Special characters used in
                        "<" / ">" /     ;  other parts of the syntax
                        "[" / "]" /
                        ":" / ";" /
                        "@" / "\" /
                        "," / "." /
                        DQUOTE
```

No special semantics are attached to these tokens. They are simply single characters.

3.2.2. Quoted characters

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

```
quoted-pair     =       ("\" text) / obs-qp
```

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

Note: The "\" character may appear in a message where it is not part of a quoted-pair. A "\" character that does not appear in a quoted-pair is not semantically invisible. The only places in this standard where quoted-pair currently appears are ccontent, qcontent, dcontent, no-fold-quote, and no-fold-literal.

3.2.3. Folding white space and comments

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

Strings of characters enclosed in parentheses are considered comments so long as they do not appear within a "quoted-string", as defined in section 3.2.5. Comments may nest.

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

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

ctext           =       NO-WS-CTL /     ; Non white space controls

                        %d33-39 /       ; The rest of the US-ASCII
                        %d42-91 /       ;  characters not including "(",
                        %d93-126        ;  ")", or "\"

ccontent        =       ctext / quoted-pair / comment

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

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

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

A comment is normally used in a structured field body to provide some human readable informational text. Since a comment is allowed to contain FWS, folding is permitted within the comment. Also note that since quoted-pair is allowed in a comment, the parentheses and

backslash characters may appear in a comment so long as they appear as a quoted-pair. Semantically, the enclosing parentheses are not part of the comment; the comment is what is contained between the two parentheses. As stated earlier, the "\" in any quoted-pair and the CRLF in any FWS that appears within the comment are semantically "invisible" and therefore not part of the comment either.

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

3.2.4. Atom

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

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

```
atext           =       ALPHA / DIGIT / ; Any character except controls,
                        "!" / "#" /     ;  SP, and specials.
                        "$" / "%" /     ;  Used for atoms
                        "&" / "'" /
                        "*" / "+" /
                        "-" / "/" /
                        "=" / "?" /
                        "^" / "_" /
                        "`" / "{" /
                        "|" / "}" /
                        "~"

atom            =       [CFWS] 1*atext [CFWS]

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

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

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

3.2.5. Quoted strings

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

```
qtext           =       NO-WS-CTL /     ; Non white space controls

                        %d33 /          ; The rest of the US-ASCII
                        %d35-91 /       ;  characters not including "\"
                        %d93-126        ;  or the quote character

qcontent        =       qtext / quoted-pair

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

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

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

3.2.6. Miscellaneous tokens

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

```
word            =       atom / quoted-string

phrase          =       1*word / obs-phrase
```

```
utext           =       NO-WS-CTL /     ; Non white space controls
                        %d33-126 /      ; The rest of US-ASCII
                        obs-utext

unstructured    =       *([FWS] utext) [FWS]
```

3.3. Date and Time Specification

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

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

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

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

date            =       day month year

year            =       4*DIGIT / obs-year

month           =       (FWS month-name FWS) / obs-month

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

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

time            =       time-of-day FWS zone

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

hour            =       2DIGIT / obs-hour

minute          =       2DIGIT / obs-minute

second          =       2DIGIT / obs-second

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

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

The time-of-day specifies the number of hours, minutes, and optionally seconds since midnight of the date indicated.

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

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

A date-time specification MUST be semantically valid. That is, the day-of-the-week (if included) MUST be the day implied by the date, the numeric day-of-month MUST be between 1 and the number of days allowed for the specified month (in the specified year), the time-of-day MUST be in the range 00:00:00 through 23:59:60 (the number of seconds allowing for a leap second; see [STD12]), and the zone MUST be within the range -9959 through +9959.

3.4. Address Specification

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

```
address         =       mailbox / group

mailbox         =       name-addr / addr-spec

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

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

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

```
display-name    =       phrase

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

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

A mailbox receives mail. It is a conceptual entity which does not necessarily pertain to file storage. For example, some sites may choose to print mail on a printer and deliver the output to the addressee's desk. Normally, a mailbox is comprised of two parts: (1) an optional display name that indicates the name of the recipient (which could be a person or a system) that could be displayed to the user of a mail application, and (2) an addr-spec address enclosed in angle brackets ("<" and ">"). There is also an alternate simple form of a mailbox where the addr-spec address appears alone, without the recipient's name or the angle brackets. The Internet addr-spec address is described in section 3.4.1.

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

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

3.4.1. Addr-spec specification

An addr-spec is a specific Internet identifier that contains a locally interpreted string followed by the at-sign character ("@", ASCII value 64) followed by an Internet domain. The locally interpreted string is either a quoted-string or a dot-atom. If the string can be represented as a dot-atom (that is, it contains no characters other than atext characters or "." surrounded by atext

characters), then the dot-atom form SHOULD be used and the quoted-string form SHOULD NOT be used. Comments and folding white space SHOULD NOT be used around the "@" in the addr-spec.

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

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

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

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

dcontent        =       dtext / quoted-pair

dtext           =       NO-WS-CTL /     ; Non white space controls

                        %d33-90 /       ; The rest of the US-ASCII
                        %d94-126        ;  characters not including "[",
                                        ;  "]", or "\"
```

The domain portion identifies the point to which the mail is delivered. In the dot-atom form, this is interpreted as an Internet domain name (either a host name or a mail exchanger name) as described in [STD3, STD13, STD14]. In the domain-literal form, the domain is interpreted as the literal Internet address of the particular host. In both cases, how addressing is used and how messages are transported to a particular host is covered in the mail transport document [RFC2821]. These mechanisms are outside of the scope of this document.

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

3.5 Overall message syntax

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

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

body            =       *(*998text CRLF) *998text
```

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

3.6. Field definitions

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

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

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

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

```
fields          =       *(trace
                          *(resent-date /
                           resent-from /
                           resent-sender /
                           resent-to /
                           resent-cc /
                           resent-bcc /
                           resent-msg-id))
                        *(orig-date /
                        from /
                        sender /
                        reply-to /
```

```
                        to /
                        cc /
                        bcc /
                        message-id /
                        in-reply-to /
                        references /
                        subject /
                        comments /
                        keywords /
                        optional-field)
```

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

Field	Min number	Max number	Notes
trace	0	unlimited	Block prepended - see 3.6.7
resent-date	0*	unlimited*	One per block, required if other resent fields present - see 3.6.6
resent-from	0	unlimited*	One per block - see 3.6.6
resent-sender	0*	unlimited*	One per block, MUST occur with multi-address resent-from - see 3.6.6
resent-to	0	unlimited*	One per block - see 3.6.6
resent-cc	0	unlimited*	One per block - see 3.6.6
resent-bcc	0	unlimited*	One per block - see 3.6.6
resent-msg-id	0	unlimited*	One per block - see 3.6.6
orig-date	1	1	
from	1	1	See sender and 3.6.2

sender	0*	1	MUST occur with multi-address from - see 3.6.2
reply-to	0	1	
to	0	1	
cc	0	1	
bcc	0	1	
message-id	0*	1	SHOULD be present - see 3.6.4
in-reply-to	0*	1	SHOULD occur in some replies - see 3.6.4
references	0*	1	SHOULD occur in some replies - see 3.6.4
subject	0	1	
comments	0	unlimited	
keywords	0	unlimited	
optional-field	0	unlimited	

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

3.6.1. The origination date field

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

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

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

for delivery. The origination date is intended to contain the date and time that the user queued the message, not the time when the user connected to the network to send the message.)

3.6.2. Originator fields

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

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

sender          =       "Sender:" mailbox CRLF

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

The originator fields indicate the mailbox(es) of the source of the message. The "From:" field specifies the author(s) of the message, that is, the mailbox(es) of the person(s) or system(s) responsible for the writing of the message. The "Sender:" field specifies the mailbox of the agent responsible for the actual transmission of the message. For example, if a secretary were to send a message for another person, the mailbox of the secretary would appear in the "Sender:" field and the mailbox of the actual author would appear in the "From:" field. If the originator of the message can be indicated by a single mailbox and the author and transmitter are identical, the "Sender:" field SHOULD NOT be used. Otherwise, both fields SHOULD appear.

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

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

3.6.3. Destination address fields

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

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

cc              =       "Cc:" address-list CRLF

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

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

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

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

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

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

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

3.6.4. Identification fields

Though optional, every message SHOULD have a "Message-ID:" field. Furthermore, reply messages SHOULD have "In-Reply-To:" and "References:" fields as appropriate, as described below.

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

The message identifier (msg-id) is similar in syntax to an angle-addr construct without the internal CFWS.

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

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

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

msg-id          =       [CFWS] "<" id-left "@" id-right ">" [CFWS]

id-left         =       dot-atom-text / no-fold-quote / obs-id-left

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

no-fold-quote   =       DQUOTE *(qtext / quoted-pair) DQUOTE
```

```
no-fold-literal =       "[" *(dtext / quoted-pair) "]"
```

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

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

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

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

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

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

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

The message identifier (msg-id) itself MUST be a globally unique identifier for a message. The generator of the message identifier MUST guarantee that the msg-id is unique. There are several algorithms that can be used to accomplish this. Since the msg-id has a similar syntax to angle-addr (identical except that comments and folding white space are not allowed), a good method is to put the domain name (or a domain literal IP address) of the host on which the message identifier was created on the right hand side of the "@", and put a combination of the current absolute date and time along with some other currently unique (perhaps sequential) identifier available on the system (for example, a process id number) on the left hand side. Using a date on the left hand side and a domain name or domain literal on the right hand side makes it possible to guarantee uniqueness since no two hosts use the same domain name or IP address at the same time. Though other algorithms will work, it is RECOMMENDED that the right hand side contain some domain identifier (either of the host itself or otherwise) such that the generator of the message identifier can guarantee the uniqueness of the left hand side within the scope of that domain.

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

3.6.5. Informational fields

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

```
subject         =       "Subject:" unstructured CRLF

comments        =       "Comments:" unstructured CRLF

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

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

3.6.6. Resent fields

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

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

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

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

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

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

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

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

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

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

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

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

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

Note: When replying to a resent message, replies behave just as they would with any other message, using the original "From:",

"Reply-To:", "Message-ID:", and other fields. The resent fields are only informational and MUST NOT be used in the normal processing of replies.

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

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

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

3.6.7. Trace fields

The trace fields are a group of header fields consisting of an optional "Return-Path:" field, and one or more "Received:" fields. The "Return-Path:" header field contains a pair of angle brackets that enclose an optional addr-spec. The "Received:" field contains a (possibly empty) list of name/value pairs followed by a semicolon and a date-time specification. The first item of the name/value pair is defined by item-name, and the second item is either an addr-spec, an atom, a domain, or a msg-id. Further restrictions may be applied to the syntax of the trace fields by standards that provide for their use, such as [RFC2821].

```
trace           =       [return]
                        1*received

return          =       "Return-Path:" path CRLF

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

received        =       "Received:" name-val-list ";" date-time CRLF

name-val-list   =       [CFWS] [name-val-pair *(CFWS name-val-pair)]

name-val-pair   =       item-name CFWS item-value

item-name       =       ALPHA *(["-"] (ALPHA / DIGIT))

item-value      =       1*angle-addr / addr-spec /
                         atom / domain / msg-id
```

A full discussion of the Internet mail use of trace fields is contained in [RFC2821]. For the purposes of this standard, the trace fields are strictly informational, and any formal interpretation of them is outside of the scope of this document.

3.6.8. Optional fields

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

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

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

field-name      =       1*ftext

ftext           =       %d33-57 /               ; Any character except
                        %d59-126                ;  controls, SP, and
                                                ;  ":".
```

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

4. Obsolete Syntax

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

Note: This section identifies syntactic forms that any implementation MUST reasonably interpret. However, there are certainly Internet messages which do not conform to even the additional syntax given in this section. The fact that a particular form does not appear in any section of this document is not justification for computer programs to crash or for malformed data to be irretrievably lost by any implementation. To repeat an example, though this document requires lines in messages to be no longer than 998 characters, silently

discarding the 999th and subsequent characters in a line without warning would still be bad behavior for an implementation. It is up to the implementation to deal with messages robustly.

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

Another key difference between the obsolete and the current syntax is that the rule in section 3.2.3 regarding lines composed entirely of white space in comments and folding white space does not apply. See the discussion of folding white space in section 4.2 below.

Finally, certain characters that were formerly allowed in messages appear in this section. The NUL character (ASCII value 0) was once allowed, but is no longer for compatibility reasons. CR and LF were allowed to appear in messages other than as CRLF; this use is also shown here.

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

4.1. Miscellaneous obsolete tokens

These syntactic elements are used elsewhere in the obsolete syntax or in the main syntax. The obs-char and obs-qp elements each add ASCII value 0. Bare CR and bare LF are added to obs-text and obs-utext. The period character is added to obs-phrase. The obs-phrase-list provides for "empty" elements in a comma-separated list of phrases.

Note: The "period" (or "full stop") character (".") in obs-phrase is not a form that was allowed in earlier versions of this or any other standard. Period (nor any other character from specials) was not allowed in phrase because it introduced a parsing difficulty distinguishing between phrases and portions of an addr-spec (see section 4.4). It appears here because the period character is currently used in many messages in the display-name portion of addresses, especially for initials in names, and therefore must be interpreted properly. In the future, period may appear in the regular syntax of phrase.

```
obs-qp          =       "\" (%d0-127)

obs-text        =       *LF *CR *(obs-char *LF *CR)
```

```
obs-char        =       %d0-9 / %d11 /          ; %d0-127 except CR and
                        %d12 / %d14-127         ;  LF

obs-utext       =       obs-text

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

obs-phrase-list =       phrase / 1*([phrase] [CFWS] "," [CFWS]) [phrase]
```

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

4.2. Obsolete folding white space

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

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

4.3. Obsolete Date and Time

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

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

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

obs-month       =       CFWS month-name CFWS

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

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

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

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

obs-zone        =       "UT" / "GMT" /          ; Universal Time
```

```
                                        ; North American UT
                                        ; offsets
                        "EST" / "EDT" /         ; Eastern:  - 5/ - 4
                        "CST" / "CDT" /         ; Central:  - 6/ - 5
                        "MST" / "MDT" /         ; Mountain: - 7/ - 6
                        "PST" / "PDT" /         ; Pacific:  - 8/ - 7

                        %d65-73 /               ; Military zones - "A"
                        %d75-90 /               ; through "I" and "K"
                        %d97-105 /              ; through "Z", both
                        %d107-122               ; upper and lower case
```

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

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

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

EDT is semantically equivalent to -0400
EST is semantically equivalent to -0500
CDT is semantically equivalent to -0500
CST is semantically equivalent to -0600
MDT is semantically equivalent to -0600
MST is semantically equivalent to -0700
PDT is semantically equivalent to -0700
PST is semantically equivalent to -0800

The 1 character military time zones were defined in a non-standard way in [RFC822] and are therefore unpredictable in their meaning. The original definitions of the military zones "A" through "I" are equivalent to "+0100" through "+0900" respectively; "K", "L", and "M" are equivalent to "+1000", "+1100", and "+1200" respectively; "N" through "Y" are equivalent to "-0100" through "-1200" respectively; and "Z" is equivalent to "+0000". However, because of the error in [RFC822], they SHOULD all be considered equivalent to "-0000" unless there is out-of-band information confirming their meaning.

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

4.4. Obsolete Addressing

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

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

obs-route       =       [CFWS] obs-domain-list ":" [CFWS]

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

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

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

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

obs-addr-list   =       1*([address] [CFWS] "," [CFWS]) [address]
```

When interpreting addresses, the route portion SHOULD be ignored.

4.5. Obsolete header fields

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

```
obs-fields      =       *(obs-return /
                        obs-received /
                        obs-orig-date /
                        obs-from /
                        obs-sender /
                        obs-reply-to /
                        obs-to /
```

```
                        obs-cc /
                        obs-bcc /
                        obs-message-id /
                        obs-in-reply-to /
                        obs-references /
                        obs-subject /
                        obs-comments /
                        obs-keywords /
                        obs-resent-date /
                        obs-resent-from /
                        obs-resent-send /
                        obs-resent-rply /
                        obs-resent-to /
                        obs-resent-cc /
                        obs-resent-bcc /
                        obs-resent-mid /
                        obs-optional)
```

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

4.5.1. Obsolete origination date field

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

4.5.2. Obsolete originator fields

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

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

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

4.5.3. Obsolete destination address fields

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

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

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

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

4.5.4. Obsolete identification fields

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

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

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

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

obs-id-left     =       local-part

obs-id-right    =       domain
```

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

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

4.5.5. Obsolete informational fields

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

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

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

4.5.6. Obsolete resent fields

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

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

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

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

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

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

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

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

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

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

4.5.7. Obsolete trace fields

The obs-return and obs-received are again given here as template definitions, just as return and received are in section 3. Their full syntax is given in [RFC2821].

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

obs-received    =       "Received" *WSP ":" name-val-list CRLF

obs-path        =       obs-angle-addr
```

4.5.8. Obsolete optional fields

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

5. Security Considerations

Care needs to be taken when displaying messages on a terminal or terminal emulator. Powerful terminals may act on escape sequences and other combinations of ASCII control characters with a variety of consequences. They can remap the keyboard or permit other modifications to the terminal which could lead to denial of service or even damaged data. They can trigger (sometimes programmable) answerback messages which can allow a message to cause commands to be issued on the recipient's behalf. They can also effect the operation of terminal attached devices such as printers. Message viewers may wish to strip potentially dangerous terminal escape sequences from the message prior to display. However, other escape sequences appear in messages for useful purposes (cf. [RFC2045, RFC2046, RFC2047, RFC2048, RFC2049, ISO2022]) and therefore should not be stripped indiscriminately.

Transmission of non-text objects in messages raises additional security issues. These issues are discussed in [RFC2045, RFC2046, RFC2047, RFC2048, RFC2049].

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

6. Bibliography

[ASCII] American National Standards Institute (ANSI), Coded Character Set - 7-Bit American National Standard Code for Information Interchange, ANSI X3.4, 1986.

[ISO2022] International Organization for Standardization (ISO), Information processing - ISO 7-bit and 8-bit coded character sets - Code extension techniques, Third edition - 1986-05-01, ISO 2022, 1986.

[RFC822] Crocker, D., "Standard for the Format of ARPA Internet Text Messages", RFC 822, August 1982.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[RFC2047] Moore, K., "Multipurpose Internet Mail Extensions (MIME) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, November 1996.

[RFC2048] Freed, N., Klensin, J. and J. Postel, "Multipurpose Internet Mail Extensions (MIME) Part Four: Format of Internet Message Bodies", RFC 2048, November 1996.

[RFC2049] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples", RFC 2049, November 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2234] Crocker, D., Editor, and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[RFC2821] Klensin, J., Editor, "Simple Mail Transfer Protocol", RFC 2821, March 2001.

[STD3] Braden, R., "Host Requirements", STD 3, RFC 1122 and RFC 1123, October 1989.

[STD12] Mills, D., "Network Time Protocol", STD 12, RFC 1119, September 1989.

[STD13] Mockapetris, P., "Domain Name System", STD 13, RFC 1034 and RFC 1035, November 1987.

[STD14] Partridge, C., "Mail Routing and the Domain System", STD 14, RFC 974, January 1986.

7. Editor's Address

Peter W. Resnick
QUALCOMM Incorporated
5775 Morehouse Drive
San Diego, CA 92121-1714
USA

Phone: +1 858 651 4478
Fax: +1 858 651 1102
EMail: presnick@qualcomm.com

8. Acknowledgements

Many people contributed to this document. They included folks who participated in the Detailed Revision and Update of Messaging Standards (DRUMS) Working Group of the Internet Engineering Task Force (IETF), the chair of DRUMS, the Area Directors of the IETF, and people who simply sent their comments in via e-mail. The editor is deeply indebted to them all and thanks them sincerely. The below list includes everyone who sent e-mail concerning this document. Hopefully, everyone who contributed is named here:

Matti Aarnio
Tanaka Akira
Russ Allbery
Eric Allman
Harald Tveit Alvestrand
Ran Atkinson
Jos Backus
Bruce Balden
Dave Barr
Alan Barrett
John Beck
J. Robert von Behren
Jos den Bekker
D. J. Bernstein
James Berriman
Norbert Bollow
Raj Bose
Antony Bowesman
Scott Bradner
Randy Bush
Tom Byrer
Bruce Campbell
Larry Campbell
W. J. Carpenter
Michael Chapman
Richard Clayton
Maurizio Codogno
Jim Conklin
R. Kelley Cook
Steve Coya
Mark Crispin
Dave Crocker
Matt Curtin
Michael D'Errico
Cyrus Daboo
Jutta Degener
Mark Delany
Barry Finkel
Erik Forsberg
Chuck Foster
Paul Fox
Klaus M. Frank
Ned Freed
Jochen Friedrich
Randall C. Gellens
Sukvinder Singh Gill
Tim Goodwin
Philip Guenther
Tony Hansen
John Hawkinson
Philip Hazel
Kai Henningsen
Robert Herriot
Paul Hethmon
Jim Hill
Paul E. Hoffman
Steve Hole
Kari Hurtta
Marco S. Hyman
Ofer Inbar
Olle Jarnefors
Kevin Johnson
Sudish Joseph
Maynard Kang
Prabhat Keni
John C. Klensin
Graham Klyne
Brad Knowles
Shuhei Kobayashi
Peter Koch
Dan Kohn
Christian Kuhtz
Anand Kumria
Steen Larsen
Larry Masinter
Denis McKeon
William P McQuillan
Alexey Melnikov
Perry E. Metzger
Steven Miller
Keith Moore
John Gardiner Myers
Chris Newman
John W. Noerenberg
Eric Norman
Mike O'Dell
Larry Osterman
Paul Overell
Jacob Palme
Michael A. Patton
Uzi Paz
Michael A. Quinlan
Eric S. Raymond
Sam Roberts
Hugh Sasse
Bart Schaefer
Tom Scola
Wolfgang Segmuller
Nick Shelness
John Stanley
Einar Stefferud
Jeff Stephenson
Bernard Stern
Peter Sylvester
Mark Symons
Eric Thomas
Lee Thompson
Karel De Vriendt
Matthew Wall
Rolf Weber
Brent B. Welch

Steve Dorner	Eliot Lear	Dan Wing
Harold A. Driscoll	Barry Leiba	Jack De Winter
Michael Elkins	Jay Levitt	Gregory J. Woodhouse
Robert Elz	Lars-Johan Liman	Greg A. Woods
Johnny Eriksson	Charles Lindsey	Kazu Yamamoto
Erik E. Fair	Pete Loshin	Alain Zahm
Roger Fajman	Simon Lyall	Jamie Zawinski
Patrik Faltstrom	Bill Manning	Timothy S. Zurcher
Claus Andre Farber	John Martin	

Appendix A. Example messages

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

Messages are delimited in this section between lines of "----". The "----" lines are not part of the message itself.

A.1. Addressing examples

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

A.1.1. A message from one person to another with simple addressing

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

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

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

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

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

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

A.1.2. Different types of mailboxes

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

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

Hi everyone.
----
```

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

A.1.3. Group addresses

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

Testing.
----
```

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

A.2. Reply messages

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

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

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

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

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

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

This is a reply to your hello.
----
```

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

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

This is a reply to your reply.
----
```

A.3. Resent messages

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

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

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

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

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

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

If Jane, in turn, wished to resend this message to another person, she would prepend her own set of resent header fields to the above and send that.

A.4. Messages with trace fields

As messages are sent through the transport system as described in [RFC2821], trace fields are prepended to the message. The following is an example of what those trace fields might look like. Note that there is some folding white space in the first one since these lines can be long.

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

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

A.5. White space, comments, and other oddities

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

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

Testing.
----
```

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

A.6. Obsoleted forms

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

A.6.1. Obsolete addressing

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

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

Hi everyone.
----
```

A.6.2. Obsolete dates

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

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

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

A.6.3. Obsolete white space and comments

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

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

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

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

Appendix B. Differences from earlier standards

This appendix contains a list of changes that have been made in the Internet Message Format from earlier standards, specifically [RFC822] and [STD3]. Items marked with an asterisk (*) below are items which appear in section 4 of this document and therefore can no longer be generated.

1. Period allowed in obsolete form of phrase.
2. ABNF moved out of document to [RFC2234].
3. Four or more digits allowed for year.
4. Header field ordering (and lack thereof) made explicit.
5. Encrypted header field removed.
6. Received syntax loosened to allow any token/value pair.
7. Specifically allow and give meaning to "-0000" time zone.
8. Folding white space is not allowed between every token.
9. Requirement for destinations removed.
10. Forwarding and resending redefined.
11. Extension header fields no longer specifically called out.
12. ASCII 0 (null) removed.*
13. Folding continuation lines cannot contain only white space.*
14. Free insertion of comments not allowed in date.*
15. Non-numeric time zones not allowed.*
16. Two digit years not allowed.*
17. Three digit years interpreted, but not allowed for generation.
18. Routes in addresses not allowed.*
19. CFWS within local-parts and domains not allowed.*
20. Empty members of address lists not allowed.*

21. Folding white space between field name and colon not allowed.*
22. Comments between field name and colon not allowed.
23. Tightened syntax of in-reply-to and references.*
24. CFWS within msg-id not allowed.*
25. Tightened semantics of resent fields as informational only.
26. Resent-Reply-To not allowed.*
27. No multiple occurrences of fields (except resent and received).*
28. Free CR and LF not allowed.*
29. Routes in return path not allowed.*
30. Line length limits specified.
31. Bcc more clearly specified.

Appendix C. Notices

Intellectual Property

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

Full Copyright Statement

Copyright (C) The Internet Society (2001). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

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