Network Working Group
Request for Comments: 2045
Obsoletes: 1521, 1522, 1590
Category: Standards Track

N. Freed
Innosoft
N. Borenstein
First Virtual
November 1996

# Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies

## Status of this Memo



## Abstract

STD 11, RFC 822, defines a message representation protocol specifying considerable detail about US-ASCII message headers, and leaves the message content, or message body, as flat US-ASCII text. This set of documents, collectively called the Multipurpose Internet Mail Extensions, or MIME, redefines the format of messages to allow for

(1) textual message bodies in character sets other than US-ASCII,

(2) an extensible set of different formats for non-textual message bodies,

(3) multi-part message bodies, and

(4) textual header information in character sets other than US-ASCII.

These documents are based on earlier work documented in RFC 934, STD 11, and RFC 1049, but extends and revises them. Because RFC 822 said so little about message bodies, these documents are largely orthogonal to (rather than a revision of) RFC 822.

This initial document specifies the various headers used to describe the structure of MIME messages. The second document, RFC 2046, defines the general structure of the MIME media typing system and defines an initial set of media types. The third document, RFC 2047, describes extensions to RFC 822 to allow non-US-ASCII text data in

Internet mail header fields. The fourth document, RFC 2048, specifies various IANA registration procedures for MIME-related facilities. The fifth and final document, RFC 2049, describes MIME conformance criteria as well as providing some illustrative examples of MIME message formats, acknowledgements, and the bibliography.

These documents are revisions of RFCs 1521, 1522, and 1590, which themselves were revisions of RFCs 1341 and 1342. An appendix in RFC 2049 describes differences and changes from previous versions.

Table of Contents

## 1. Introduction

Since its publication in 1982, RFC 822 has defined the standard format of textual mail messages on the Internet. Its success has been such that the RFC 822 format has been adopted, wholly or partially, well beyond the confines of the Internet and the Internet SMTP transport defined by RFC 821. As the format has seen wider use, a number of limitations have proven increasingly restrictive for the user community.

RFC 822 was intended to specify a format for text messages. As such, non-text messages, such as multimedia messages that might include audio or images, are simply not mentioned. Even in the case of text, however, RFC 822 is inadequate for the needs of mail users whose languages require the use of character sets richer than US-ASCII. Since RFC 822 does not specify mechanisms for mail containing audio, video, Asian language text, or even text in most European languages, additional specifications are needed.

One of the notable limitations of RFC 821/822 based mail systems is the fact that they limit the contents of electronic mail messages to relatively short lines (e.g. 1000 characters or less [RFC-821]) of 7bit US-ASCII. This forces users to convert any non-textual data that they may wish to send into seven-bit bytes representable as printable US-ASCII characters before invoking a local mail UA (User Agent, a program with which human users send and receive mail). Examples of such encodings currently used in the Internet include pure hexadecimal, uuencode, the 3-in-4 base 64 scheme specified in RFC 1421, the Andrew Toolkit Representation [ATK], and many others.

The limitations of RFC 822 mail become even more apparent as gateways are designed to allow for the exchange of mail messages between RFC 822 hosts and X.400 hosts. X.400 [X400] specifies mechanisms for the inclusion of non-textual material within electronic mail messages. The current standards for the mapping of X.400 messages to RFC 822 messages specify either that X.400 non-textual material must be converted to (not encoded in) IA5Text format, or that they must be discarded, notifying the RFC 822 user that discarding has occurred. This is clearly undesirable, as information that a user may wish to receive is lost. Even though a user agent may not have the capability of dealing with the non-textual material, the user might have some mechanism external to the UA that can extract useful information from the material. Moreover, it does not allow for the fact that the message may eventually be gatewayed back into an X.400 message handling system (i.e., the X.400 message is "tunneled" through Internet mail), where the non-textual information would definitely become useful again.

This document describes several mechanisms that combine to solve most of these problems without introducing any serious incompatibilities with the existing world of RFC 822 mail. In particular, it describes:

(1) A MIME-Version header field, which uses a version number to declare a message to be conformant with MIME and allows mail processing agents to distinguish between such messages and those generated by older or non-conformant software, which are presumed to lack such a field.

(2) A Content-Type header field, generalized from RFC 1049, which can be used to specify the media type and subtype of data in the body of a message and to fully specify the native representation (canonical form) of such data.

(3) A Content-Transfer-Encoding header field, which can be used to specify both the encoding transformation that was applied to the body and the domain of the result. Encoding transformations other than the identity transformation are usually applied to data in order to allow it to pass through mail transport mechanisms which may have data or character set limitations.

(4) Two additional header fields that can be used to further describe the data in a body, the Content-ID and Content-Description header fields.

All of the header fields defined in this document are subject to the general syntactic rules for header fields specified in RFC 822. In particular, all of these header fields except for Content-Disposition can include RFC 822 comments, which have no semantic content and should be ignored during MIME processing.

Finally, to specify and promote interoperability, RFC 2049 provides a basic applicability statement for a subset of the above mechanisms that defines a minimal level of "conformance" with this document.

HISTORICAL NOTE: Several of the mechanisms described in this set of documents may seem somewhat strange or even baroque at first reading. It is important to note that compatibility with existing standards AND robustness across existing practice were two of the highest priorities of the working group that developed this set of documents. In particular, compatibility was always favored over elegance.

Please refer to the current edition of the "Internet Official Protocol Standards" for the standardization state and status of this protocol. RFC 822 and STD 3, RFC 1123 also provide essential background for MIME since no conforming implementation of MIME can violate them. In addition, several other informational RFC documents will be of interest to the MIME implementor, in particular RFC 1344, RFC 1345, and RFC 1524.

## 2. Definitions, Conventions, and Generic BNF Grammar

Although the mechanisms specified in this set of documents are all described in prose, most are also described formally in the augmented BNF notation of RFC 822. Implementors will need to be familiar with this notation in order to understand this set of documents, and are referred to RFC 822 for a complete explanation of the augmented BNF notation.

Some of the augmented BNF in this set of documents makes named references to syntax rules defined in RFC 822. A complete formal grammar, then, is obtained by combining the collected grammar appendices in each document in this set with the BNF of RFC 822 plus the modifications to RFC 822 defined in RFC 1123 (which specifically changes the syntax for `return', `date' and `mailbox').

All numeric and octet values are given in decimal notation in this set of documents. All media type values, subtype values, and parameter names as defined are case-insensitive. However, parameter values are case-sensitive unless otherwise specified for the specific parameter.

FORMATTING NOTE: Notes, such at this one, provide additional nonessential information which may be skipped by the reader without missing anything essential. The primary purpose of these non-essential notes is to convey information about the rationale of this set of documents, or to place these documents in the proper historical or evolutionary context. Such information may in particular be skipped by those who are focused entirely on building a conformant implementation, but may be of use to those who wish to understand why certain design choices were made.

### 2.1. CRLF

The term CRLF, in this set of documents, refers to the sequence of octets corresponding to the two US-ASCII characters CR (decimal value 13) and LF (decimal value 10) which, taken together, in this order, denote a line break in RFC 822 mail.

## 2.2. Character Set

The term "character set" is used in MIME to refer to a method of converting a sequence of octets into a sequence of characters. Note that unconditional and unambiguous conversion in the other direction is not required, in that not all characters may be representable by a given character set and a character set may provide more than one sequence of octets to represent a particular sequence of characters.

This definition is intended to allow various kinds of character encodings, from simple single-table mappings such as US-ASCII to complex table switching methods such as those that use ISO 2022's techniques, to be used as character sets. However, the definition associated with a MIME character set name must fully specify the mapping to be performed. In particular, use of external profiling information to determine the exact mapping is not permitted.

NOTE: The term "character set" was originally to describe such straightforward schemes as US-ASCII and ISO-8859-1 which have a simple one-to-one mapping from single octets to single characters. Multi-octet coded character sets and switching techniques make the situation more complex. For example, some communities use the term "character encoding" for what MIME calls a "character set", while using the phrase "coded character set" to denote an abstract mapping from integers (not octets) to characters.

## 2.3. Message

The term "message", when not further qualified, means either a (complete or "top-level") RFC 822 message being transferred on a network, or a message encapsulated in a body of type "message/rfc822" or "message/partial".

## 2.4. Entity

The term "entity", refers specifically to the MIME-defined header fields and contents of either a message or one of the parts in the body of a multipart entity. The specification of such entities is the essence of MIME. Since the contents of an entity are often called the "body", it makes sense to speak about the body of an entity. Any sort of field may be present in the header of an entity, but only those fields whose names begin with "content-" actually have any MIME-related meaning. Note that this does NOT imply thay they have no meaning at all -- an entity that is also a message has non-MIME header fields whose meanings are defined by RFC 822.

## 2.5. Body Part

The term "body part" refers to an entity inside of a multipart entity.

## 2.6. Body

The term "body", when not further qualified, means the body of an entity, that is, the body of either a message or of a body part.

NOTE: The previous four definitions are clearly circular. This is unavoidable, since the overall structure of a MIME message is indeed recursive.

## 2.7. 7bit Data

"7bit data" refers to data that is all represented as relatively short lines with 998 octets or less between CRLF line separation sequences [RFC-821]. No octets with decimal values greater than 127 are allowed and neither are NULs (octets with decimal value 0). CR (decimal value 13) and LF (decimal value 10) octets only occur as part of CRLF line separation sequences.

## 2.8. 8bit Data

"8bit data" refers to data that is all represented as relatively short lines with 998 octets or less between CRLF line separation sequences [RFC-821]), but octets with decimal values greater than 127 may be used. As with "7bit data" CR and LF octets only occur as part of CRLF line separation sequences and no NULs are allowed.

## 2.9. Binary Data

"Binary data" refers to data where any sequence of octets whatsoever is allowed.

## 2.10. Lines

"Lines" are defined as sequences of octets separated by a CRLF sequences. This is consistent with both RFC 821 and RFC 822. "Lines" only refers to a unit of data in a message, which may or may not correspond to something that is actually displayed by a user agent.

## 3. MIME Header Fields

MIME defines a number of new RFC 822 header fields that are used to describe the content of a MIME entity. These header fields occur in at least two contexts:

(1) As part of a regular RFC 822 message header.

(2) In a MIME body part header within a multipart construct.

The formal definition of these header fields is as follows:

```
entity-headers := [ content CRLF ]
                  [ encoding CRLF ]
                  [ id CRLF ]
                  [ description CRLF ]
                  *( MIME-extension-field CRLF )

MIME-message-headers := entity-headers
                        fields
                        version CRLF
                        ; The ordering of the header
                        ; fields implied by this BNF
                        ; definition should be ignored.

MIME-part-headers := entity-headers
                     [ fields ]
                     ; Any field not beginning with
                     ; "content-" can have no defined
                     ; meaning and may be ignored.
                     ; The ordering of the header
                     ; fields implied by this BNF
                     ; definition should be ignored.
```

The syntax of the various specific MIME header fields will be described in the following sections.

## 4. MIME-Version Header Field

Since RFC 822 was published in 1982, there has really been only one format standard for Internet messages, and there has been little perceived need to declare the format standard in use. This document is an independent specification that complements RFC 822. Although the extensions in this document have been defined in such a way as to be compatible with RFC 822, there are still circumstances in which it might be desirable for a mail-processing agent to know whether a message was composed with the new standard in mind.

Therefore, this document defines a new header field, "MIME-Version", which is to be used to declare the version of the Internet message body format standard in use.

Messages composed in accordance with this document MUST include such a header field, with the following verbatim text:

```
MIME-Version: 1.0
```

The presence of this header field is an assertion that the message has been composed in compliance with this document.

Since it is possible that a future document might extend the message format standard again, a formal BNF is given for the content of the MIME-Version field:

```
version := "MIME-Version" ":" 1*DIGIT "." 1*DIGIT
```

Thus, future format specifiers, which might replace or extend "1.0", are constrained to be two integer fields, separated by a period. If a message is received with a MIME-version value other than "1.0", it cannot be assumed to conform with this document.

Note that the MIME-Version header field is required at the top level of a message. It is not required for each body part of a multipart entity. It is required for the embedded headers of a body of type "message/rfc822" or "message/partial" if and only if the embedded message is itself claimed to be MIME-conformant.

It is not possible to fully specify how a mail reader that conforms with MIME as defined in this document should treat a message that might arrive in the future with some value of MIME-Version other than "1.0".

It is also worth noting that version control for specific media types is not accomplished using the MIME-Version mechanism. In particular, some formats (such as application/postscript) have version numbering conventions that are internal to the media format. Where such conventions exist, MIME does nothing to supersede them. Where no such conventions exist, a MIME media type might use a "version" parameter in the content-type field if necessary.

NOTE TO IMPLEMENTORS: When checking MIME-Version values any RFC 822 comment strings that are present must be ignored. In particular, the following four MIME-Version fields are equivalent:

MIME-Version: 1.0

MIME-Version: 1.0 (produced by MetaSend Vx.x)

MIME-Version: (produced by MetaSend Vx.x) 1.0

MIME-Version: 1.(produced by MetaSend Vx.x)0

In the absence of a MIME-Version field, a receiving mail user agent (whether conforming to MIME requirements or not) may optionally choose to interpret the body of the message according to local conventions. Many such conventions are currently in use and it should be noted that in practice non-MIME messages can contain just about anything.

It is impossible to be certain that a non-MIME mail message is actually plain text in the US-ASCII character set since it might well be a message that, using some set of nonstandard local conventions that predate MIME, includes text in another character set or non-textual data presented in a manner that cannot be automatically recognized (e.g., a uuencoded compressed UNIX tar file).

## 5. Content-Type Header Field

The purpose of the Content-Type field is to describe the data contained in the body fully enough that the receiving user agent can pick an appropriate agent or mechanism to present the data to the user, or otherwise deal with the data in an appropriate manner. The value in this field is called a media type.

HISTORICAL NOTE: The Content-Type header field was first defined in RFC 1049. RFC 1049 used a simpler and less powerful syntax, but one that is largely compatible with the mechanism given here.

The Content-Type header field specifies the nature of the data in the body of an entity by giving media type and subtype identifiers, and by providing auxiliary information that may be required for certain media types. After the media type and subtype names, the remainder of the header field is simply a set of parameters, specified in an attribute=value notation. The ordering of parameters is not significant.

In general, the top-level media type is used to declare the general type of data, while the subtype specifies a specific format for that type of data. Thus, a media type of "image/xyz" is enough to tell a user agent that the data is an image, even if the user agent has no knowledge of the specific image format "xyz". Such information can be used, for example, to decide whether or not to show a user the raw data from an unrecognized subtype -- such an action might be reasonable for unrecognized subtypes of text, but not for unrecognized subtypes of image or audio. For this reason, registered subtypes of text, image, audio, and video should not contain embedded information that is really of a different type. Such compound formats should be represented using the "multipart" or "application" types.

Parameters are modifiers of the media subtype, and as such do not fundamentally affect the nature of the content. The set of meaningful parameters depends on the media type and subtype. Most parameters are associated with a single specific subtype. However, a given top-level media type may define parameters which are applicable to any subtype of that type. Parameters may be required by their defining content type or subtype or they may be optional. MIME implementations must ignore any parameters whose names they do not recognize.

For example, the "charset" parameter is applicable to any subtype of "text", while the "boundary" parameter is required for any subtype of the "multipart" media type.

There are NO globally-meaningful parameters that apply to all media types. Truly global mechanisms are best addressed, in the MIME model, by the definition of additional Content-* header fields.

An initial set of seven top-level media types is defined in RFC 2046. Five of these are discrete types whose content is essentially opaque as far as MIME processing is concerned. The remaining two are composite types whose contents require additional handling by MIME processors.

This set of top-level media types is intended to be substantially complete. It is expected that additions to the larger set of supported types can generally be accomplished by the creation of new subtypes of these initial types. In the future, more top-level types may be defined only by a standards-track extension to this standard. If another top-level type is to be used for any reason, it must be

given a name starting with "X-" to indicate its non-standard status and to avoid a potential conflict with a future official name.

### 5.1. Syntax of the Content-Type Header Field

In the Augmented BNF notation of RFC 822, a Content-Type header field value is defined as follows:

```
content := "Content-Type" ":" type "/" subtype
           *(";" parameter)
           ; Matching of media type and subtype
           ; is ALWAYS case-insensitive.

type := discrete-type / composite-type

discrete-type := "text" / "image" / "audio" / "video" /
                 "application" / extension-token

composite-type := "message" / "multipart" / extension-token

extension-token := ietf-token / x-token

ietf-token := <An extension token defined by a
               standards-track RFC and registered
               with IANA.>

x-token := <The two characters "X-" or "x-" followed, with
            no intervening white space, by any token>

subtype := extension-token / iana-token

iana-token := <A publicly-defined extension token. Tokens
               of this form must be registered with IANA
               as specified in RFC 2048.>

parameter := attribute "=" value

attribute := token
             ; Matching of attributes
             ; is ALWAYS case-insensitive.

value := token / quoted-string

token := 1*<any (US-ASCII) CHAR except SPACE, CTLs,
            or tspecials>

tspecials :=  "(" / ")" / "<" / ">" / "@" /
              "," / ";" / ":" / "\" / <">
              "/" / "[" / "]" / "?" / "="
              ; Must be in quoted-string,
              ; to use within parameter values
```

Note that the definition of "tspecials" is the same as the RFC 822 definition of "specials" with the addition of the three characters "/", "?", and "=", and the removal of ".".

Note also that a subtype specification is MANDATORY -- it may not be omitted from a Content-Type header field. As such, there are no default subtypes.

The type, subtype, and parameter names are not case sensitive. For example, TEXT, Text, and TeXt are all equivalent top-level media types. Parameter values are normally case sensitive, but sometimes are interpreted in a case-insensitive fashion, depending on the intended use. (For example, multipart boundaries are case-sensitive, but the "access-type" parameter for message/External-body is not case-sensitive.)

Note that the value of a quoted string parameter does not include the quotes. That is, the quotation marks in a quoted-string are not a part of the value of the parameter, but are merely used to delimit that parameter value. In addition, comments are allowed in accordance with RFC 822 rules for structured header fields. Thus the following two forms

```
Content-type: text/plain; charset=us-ascii (Plain text)

Content-type: text/plain; charset="us-ascii"
```

are completely equivalent.

Beyond this syntax, the only syntactic constraint on the definition of subtype names is the desire that their uses must not conflict. That is, it would be undesirable to have two different communities using "Content-Type: application/foobar" to mean two different things. The process of defining new media subtypes, then, is not intended to be a mechanism for imposing restrictions, but simply a mechanism for publicizing their definition and usage. There are, therefore, two acceptable mechanisms for defining new media subtypes:

(1) Private values (starting with "X-") may be defined bilaterally between two cooperating agents without outside registration or standardization. Such values cannot be registered or standardized.

(2) New standard values should be registered with IANA as described in RFC 2048.

The second document in this set, RFC 2046, defines the initial set of

media types for MIME.

### 5.2. Content-Type Defaults

Default RFC 822 messages without a MIME Content-Type header are taken by this protocol to be plain text in the US-ASCII character set, which can be explicitly specified as:

```
Content-type: text/plain; charset=us-ascii
```

This default is assumed if no Content-Type header field is specified. It is also recommend that this default be assumed when a syntactically invalid Content-Type header field is encountered. In the presence of a MIME-Version header field and the absence of any Content-Type header field, a receiving User Agent can also assume that plain US-ASCII text was the sender's intent. Plain US-ASCII text may still be assumed in the absence of a MIME-Version or the presence of an syntactically invalid Content-Type header field, but the sender's intent might have been otherwise.

## 6. Content-Transfer-Encoding Header Field

Many media types which could be usefully transported via email are represented, in their "natural" format, as 8bit character or binary data. Such data cannot be transmitted over some transfer protocols. For example, RFC 821 (SMTP) restricts mail messages to 7bit US-ASCII data with lines no longer than 1000 characters including any trailing CRLF line separator.

It is necessary, therefore, to define a standard mechanism for encoding such data into a 7bit short line format. Proper labelling of unencoded material in less restrictive formats for direct use over less restrictive transports is also desireable. This document specifies that such encodings will be indicated by a new "Content-Transfer-Encoding" header field. This field has not been defined by any previous standard.

### 6.1. Content-Transfer-Encoding Syntax

The Content-Transfer-Encoding field's value is a single token specifying the type of encoding, as enumerated below. Formally:

```
encoding := "Content-Transfer-Encoding" ":" mechanism

mechanism := "7bit" / "8bit" / "binary" /
             "quoted-printable" / "base64" /
             ietf-token / x-token
```

These values are not case sensitive -- Base64 and BASE64 and bAsE64 are all equivalent. An encoding type of 7BIT requires that the body

is already in a 7bit mail-ready representation. This is the default value -- that is, "Content-Transfer-Encoding: 7BIT" is assumed if the Content-Transfer-Encoding header field is not present.

## 6.2. Content-Transfer-Encodings Semantics

This single Content-Transfer-Encoding token actually provides two pieces of information. It specifies what sort of encoding transformation the body was subjected to and hence what decoding operation must be used to restore it to its original form, and it specifies what the domain of the result is.

The transformation part of any Content-Transfer-Encodings specifies, either explicitly or implicitly, a single, well-defined decoding algorithm, which for any sequence of encoded octets either transforms it to the original sequence of octets which was encoded, or shows that it is illegal as an encoded sequence. Content-Transfer-Encodings transformations never depend on any additional external profile information for proper operation. Note that while decoders must produce a single, well-defined output for a valid encoding no such restrictions exist for encoders: Encoding a given sequence of octets to different, equivalent encoded sequences is perfectly legal.

Three transformations are currently defined: identity, the "quoted-printable" encoding, and the "base64" encoding. The domains are "binary", "8bit" and "7bit".

The Content-Transfer-Encoding values "7bit", "8bit", and "binary" all mean that the identity (i.e. NO) encoding transformation has been performed. As such, they serve simply as indicators of the domain of the body data, and provide useful information about the sort of encoding that might be needed for transmission in a given transport system. The terms "7bit data", "8bit data", and "binary data" are all defined in Section 2.

The quoted-printable and base64 encodings transform their input from an arbitrary domain into material in the "7bit" range, thus making it safe to carry over restricted transports. The specific definition of the transformations are given below.

The proper Content-Transfer-Encoding label must always be used. Labelling unencoded data containing 8bit characters as "7bit" is not allowed, nor is labelling unencoded non-line-oriented data as anything other than "binary" allowed.

Unlike media subtypes, a proliferation of Content-Transfer-Encoding values is both undesirable and unnecessary. However, establishing only a single transformation into the "7bit" domain does not seem

possible. There is a tradeoff between the desire for a compact and efficient encoding of largely- binary data and the desire for a somewhat readable encoding of data that is mostly, but not entirely, 7bit. For this reason, at least two encoding mechanisms are necessary: a more or less readable encoding (quoted-printable) and a "dense" or "uniform" encoding (base64).

Mail transport for unencoded 8bit data is defined in RFC 1652. As of the initial publication of this document, there are no standardized Internet mail transports for which it is legitimate to include unencoded binary data in mail bodies. Thus there are no circumstances in which the "binary" Content-Transfer-Encoding is actually valid in Internet mail. However, in the event that binary mail transport becomes a reality in Internet mail, or when MIME is used in conjunction with any other binary-capable mail transport mechanism, binary bodies must be labelled as such using this mechanism.

NOTE: The five values defined for the Content-Transfer-Encoding field imply nothing about the media type other than the algorithm by which it was encoded or the transport system requirements if unencoded.

### 6.3. New Content-Transfer-Encodings

Implementors may, if necessary, define private Content-Transfer-Encoding values, but must use an x-token, which is a name prefixed by "X-", to indicate its non-standard status, e.g., "Content-Transfer-Encoding: x-my-new-encoding". Additional standardized Content-Transfer-Encoding values must be specified by a standards-track RFC. The requirements such specifications must meet are given in RFC 2048.
As such, all content-transfer-encoding namespace except that beginning with "X-" is explicitly reserved to the IETF for future use.

Unlike media types and subtypes, the creation of new Content-Transfer-Encoding values is STRONGLY discouraged, as it seems likely to hinder interoperability with little potential benefit

### 6.4. Interpretation and Use

If a Content-Transfer-Encoding header field appears as part of a message header, it applies to the entire body of that message. If a Content-Transfer-Encoding header field appears as part of an entity's headers, it applies only to the body of that entity. If an entity is of type "multipart" the Content-Transfer-Encoding is not permitted to

have any value other than "7bit", "8bit" or "binary". Even more severe restrictions apply to some subtypes of the "message" type.

It should be noted that most media types are defined in terms of octets rather than bits, so that the mechanisms described here are mechanisms for encoding arbitrary octet streams, not bit streams. If a bit stream is to be encoded via one of these mechanisms, it must first be converted to an 8bit byte stream using the network standard bit order ("big-endian"), in which the earlier bits in a stream become the higher-order bits in a 8bit byte. A bit stream not ending at an 8bit boundary must be padded with zeroes. RFC 2046 provides a mechanism for noting the addition of such padding in the case of the application/octet-stream media type, which has a "padding" parameter.

The encoding mechanisms defined here explicitly encode all data in US-ASCII. Thus, for example, suppose an entity has header fields such as:

```
Content-Type: text/plain; charset=ISO-8859-1
Content-transfer-encoding: base64
```

This must be interpreted to mean that the body is a base64 US-ASCII encoding of data that was originally in ISO-8859-1, and will be in that character set again after decoding.

Certain Content-Transfer-Encoding values may only be used on certain media types. In particular, it is EXPRESSLY FORBIDDEN to use any encodings other than "7bit", "8bit", or "binary" with any composite media type, i.e. one that recursively includes other Content-Type fields. Currently the only composite media types are "multipart" and "message". All encodings that are desired for bodies of type multipart or message must be done at the innermost level, by encoding the actual body that needs to be encoded.

It should also be noted that, by definition, if a composite entity has a transfer-encoding value such as "7bit", but one of the enclosed entities has a less restrictive value such as "8bit", then either the outer "7bit" labelling is in error, because 8bit data are included, or the inner "8bit" labelling placed an unnecessarily high demand on the transport system because the actual included data were actually 7bit-safe.

NOTE ON ENCODING RESTRICTIONS: Though the prohibition against using content-transfer-encodings on composite body data may seem overly restrictive, it is necessary to prevent nested encodings, in which data are passed through an encoding algorithm multiple times, and must be decoded multiple times in order to be properly viewed. Nested encodings add considerable complexity to user agents: Aside

from the obvious efficiency problems with such multiple encodings, they can obscure the basic structure of a message. In particular, they can imply that several decoding operations are necessary simply

to find out what types of bodies a message contains. Banning nested encodings may complicate the job of certain mail gateways, but this seems less of a problem than the effect of nested encodings on user agents.

Any entity with an unrecognized Content-Transfer-Encoding must be treated as if it has a Content-Type of "application/octet-stream", regardless of what the Content-Type header field actually says.

NOTE ON THE RELATIONSHIP BETWEEN CONTENT-TYPE AND CONTENT-TRANSFER-ENCODING: It may seem that the Content-Transfer-Encoding could be inferred from the characteristics of the media that is to be encoded, or, at the very least, that certain Content-Transfer-Encodings could be mandated for use with specific media types. There are several reasons why this is not the case. First, given the varying types of transports used for mail, some encodings may be appropriate for some combinations of media types and transports but not for others. (For example, in an 8bit transport, no encoding would be required for text in certain character sets, while such encodings are clearly required for 7bit SMTP.)

Second, certain media types may require different types of transfer encoding under different circumstances. For example, many PostScript bodies might consist entirely of short lines of 7bit data and hence require no encoding at all. Other PostScript bodies (especially those using Level 2 PostScript's binary encoding mechanism) may only be reasonably represented using a binary transport encoding. Finally, since the Content-Type field is intended to be an open-ended specification mechanism, strict specification of an association between media types and encodings effectively couples the specification of an application protocol with a specific lower-level transport. This is not desirable since the developers of a media type should not have to be aware of all the transports in use and what their limitations are.

## 6.5. Translating Encodings

The quoted-printable and base64 encodings are designed so that conversion between them is possible. The only issue that arises in such a conversion is the handling of hard line breaks in quoted-printable encoding output. When converting from quoted-printable to base64 a hard line break in the quoted-printable form represents a CRLF sequence in the canonical form of the data. It must therefore be converted to a corresponding encoded CRLF in the base64 form of the data. Similarly, a CRLF sequence in the canonical form of the data obtained after base64 decoding must be converted to a quoted-printable hard line break, but ONLY when converting text data.

## 6.6. Canonical Encoding Model

There was some confusion, in the previous versions of this RFC, regarding the model for when email data was to be converted to canonical form and encoded, and in particular how this process would affect the treatment of CRLFs, given that the representation of newlines varies greatly from system to system, and the relationship between content-transfer-encodings and character sets. A canonical model for encoding is presented in RFC 2049 for this reason.

## 6.7. Quoted-Printable Content-Transfer-Encoding

The Quoted-Printable encoding is intended to represent data that largely consists of octets that correspond to printable characters in the US-ASCII character set. It encodes the data in such a way that the resulting octets are unlikely to be modified by mail transport. If the data being encoded are mostly US-ASCII text, the encoded form of the data remains largely recognizable by humans. A body which is entirely US-ASCII may also be encoded in Quoted-Printable to ensure the integrity of the data should the message pass through a character-translating, and/or line-wrapping gateway.

In this encoding, octets are to be represented as determined by the following rules:

(1) (General 8bit representation) Any octet, except a CR or LF that is part of a CRLF line break of the canonical (standard) form of the data being encoded, may be represented by an "=" followed by a two digit hexadecimal representation of the octet's value. The digits of the hexadecimal alphabet, for this purpose, are "0123456789ABCDEF". Uppercase letters must be used; lowercase letters are not allowed. Thus, for example, the decimal value 12 (US-ASCII form feed) can be represented by "=0C", and the decimal value 61 (US-ASCII EQUAL SIGN) can be represented by "=3D". This rule must be followed except when the following rules allow an alternative encoding.

(2) (Literal representation) Octets with decimal values of 33 through 60 inclusive, and 62 through 126, inclusive, MAY be represented as the US-ASCII characters which correspond to those octets (EXCLAMATION POINT through LESS THAN, and GREATER THAN through TILDE, respectively).

(3) (White Space) Octets with values of 9 and 32 MAY be represented as US-ASCII TAB (HT) and SPACE characters,

respectively, but MUST NOT be so represented at the end of an encoded line. Any TAB (HT) or SPACE characters on an encoded line MUST thus be followed on that line by a printable character. In particular, an "=" at the end of an encoded line, indicating a soft line break (see rule #5) may follow one or more TAB (HT) or SPACE characters. It follows that an octet with decimal value 9 or 32 appearing at the end of an encoded line must be represented according to Rule #1. This rule is necessary because some MTAs (Message Transport Agents, programs which transport messages from one user to another, or perform a portion of such transfers) are known to pad lines of text with SPACEs, and others are known to remove "white space" characters from the end of a line. Therefore, when decoding a Quoted-Printable body, any trailing white space on a line must be deleted, as it will necessarily have been added by intermediate transport agents.

(4) (Line Breaks) A line break in a text body, represented as a CRLF sequence in the text canonical form, must be represented by a (RFC 822) line break, which is also a CRLF sequence, in the Quoted-Printable encoding. Since the canonical representation of media types other than text do not generally include the representation of line breaks as CRLF sequences, no hard line breaks (i.e. line breaks that are intended to be meaningful and to be displayed to the user) can occur in the quoted-printable encoding of such types. Sequences like "=0D", "=0A", "=0A=0D" and "=0D=0A" will routinely appear in non-text data represented in quoted-printable, of course.

Note that many implementations may elect to encode the local representation of various content types directly rather than converting to canonical form first, encoding, and then converting back to local representation. In particular, this may apply to plain text material on systems that use newline conventions other than a CRLF terminator sequence. Such an implementation optimization is permissible, but only when the combined canonicalization-encoding step is equivalent to performing the three steps separately.

(5) (Soft Line Breaks) The Quoted-Printable encoding REQUIRES that encoded lines be no more than 76 characters long. If longer lines are to be encoded with the Quoted-Printable encoding, "soft" line breaks

must be used. An equal sign as the last character on a encoded line indicates such a non-significant ("soft") line break in the encoded text.

Thus if the "raw" form of the line is a single unencoded line that says:

Now's the time for all folk to come to the aid of their country.

This can be represented, in the Quoted-Printable encoding, as:

```
Now's the time =
for all folk to come=
 to the aid of their country.
```

This provides a mechanism with which long lines are encoded in such a way as to be restored by the user agent. The 76 character limit does not count the trailing CRLF, but counts all other characters, including any equal signs.

Since the hyphen character ("-") may be represented as itself in the Quoted-Printable encoding, care must be taken, when encapsulating a quoted-printable encoded body inside one or more multipart entities, to ensure that the boundary delimiter does not appear anywhere in the encoded body. (A good strategy is to choose a boundary that includes a character sequence such as "=_" which can never appear in a quoted-printable body. See the definition of multipart messages in RFC 2046.)

NOTE: The quoted-printable encoding represents something of a compromise between readability and reliability in transport. Bodies encoded with the quoted-printable encoding will work reliably over most mail gateways, but may not work perfectly over a few gateways, notably those involving translation into EBCDIC. A higher level of confidence is offered by the base64 Content-Transfer-Encoding. A way to get reasonably reliable transport through EBCDIC gateways is to also quote the US-ASCII characters

```
!"#$@[\]^`{|}~
```

according to rule #1.

Because quoted-printable data is generally assumed to be line-oriented, it is to be expected that the representation of the breaks between the lines of quoted-printable data may be altered in transport, in the same manner that plain text mail has always been altered in Internet mail when passing between systems with differing

newline conventions. If such alterations are likely to constitute a

corruption of the data, it is probably more sensible to use the base64 encoding rather than the quoted-printable encoding.

NOTE: Several kinds of substrings cannot be generated according to the encoding rules for the quoted-printable content-transfer-encoding, and hence are formally illegal if they appear in the output of a quoted-printable encoder. This note enumerates these cases and suggests ways to handle such illegal substrings if any are encountered in quoted-printable data that is to be decoded.

(1) An "=" followed by two hexadecimal digits, one or both of which are lowercase letters in "abcdef", is formally illegal. A robust implementation might choose to recognize them as the corresponding uppercase letters.

(2) An "=" followed by a character that is neither a hexadecimal digit (including "abcdef") nor the CR character of a CRLF pair is illegal. This case can be the result of US-ASCII text having been included in a quoted-printable part of a message without itself having been subjected to quoted-printable encoding. A reasonable approach by a robust implementation might be to include the "=" character and the following character in the decoded data without any transformation and, if possible, indicate to the user that proper decoding was not possible at this point in the data.

(3) An "=" cannot be the ultimate or penultimate character in an encoded object. This could be handled as in case (2) above.

(4) Control characters other than TAB, or CR and LF as parts of CRLF pairs, must not appear. The same is true for octets with decimal values greater than 126. If found in incoming quoted-printable data by a decoder, a robust implementation might exclude them from the decoded data and warn the user that illegal characters were discovered.

(5) Encoded lines must not be longer than 76 characters, not counting the trailing CRLF. If longer lines are found in incoming, encoded data, a robust implementation might nevertheless decode the lines, and might report the erroneous encoding to the user.

WARNING TO IMPLEMENTORS: If binary data is encoded in quoted-printable, care must be taken to encode CR and LF characters as "=0D" and "=0A", respectively. In particular, a CRLF sequence in binary data should be encoded as "=0D=0A". Otherwise, if CRLF were represented as a hard line break, it might be incorrectly decoded on platforms with different line break conventions.

For formalists, the syntax of quoted-printable data is described by the following grammar:

```
quoted-printable := qp-line *(CRLF qp-line)

qp-line := *(qp-segment transport-padding CRLF)
           qp-part transport-padding

qp-part := qp-section
           ; Maximum length of 76 characters

qp-segment := qp-section *(SPACE / TAB) "="
              ; Maximum length of 76 characters

qp-section := [*(ptext / SPACE / TAB) ptext]

ptext := hex-octet / safe-char

safe-char := <any octet with decimal value of 33 through
             60 inclusive, and 62 through 126>
             ; Characters not listed as "mail-safe" in
             ; RFC 2049 are also not recommended.

hex-octet := "=" 2(DIGIT / "A" / "B" / "C" / "D" / "E" / "F")
             ; Octet must be used for characters > 127, =,
             ; SPACEs or TABs at the ends of lines, and is
             ; recommended for any character not listed in
             ; RFC 2049 as "mail-safe".

transport-padding := *LWSP-char
                     ; Composers MUST NOT generate
                     ; non-zero length transport
                     ; padding, but receivers MUST
                     ; be able to handle padding
                     ; added by message transports.
```

IMPORTANT: The addition of LWSP between the elements shown in this BNF is NOT allowed since this BNF does not specify a structured header field.

## 6.8. Base64 Content-Transfer-Encoding

The Base64 Content-Transfer-Encoding is designed to represent arbitrary sequences of octets in a form that need not be humanly readable. The encoding and decoding algorithms are simple, but the encoded data are consistently only about 33 percent larger than the unencoded data. This encoding is virtually identical to the one used in Privacy Enhanced Mail (PEM) applications, as defined in RFC 1421.

A 65-character subset of US-ASCII is used, enabling 6 bits to be represented per printable character. (The extra 65th character, "=", is used to signify a special processing function.)

NOTE: This subset has the important property that it is represented identically in all versions of ISO 646, including US-ASCII, and all characters in the subset are also represented identically in all versions of EBCDIC. Other popular encodings, such as the encoding used by the uuencode utility, Macintosh binhex 4.0 [RFC-1741], and the base85 encoding specified as part of Level 2 PostScript, do not share these properties, and thus do not fulfill the portability requirements a binary transport encoding for mail must meet.

The encoding process represents 24-bit groups of input bits as output strings of 4 encoded characters. Proceeding from left to right, a 24-bit input group is formed by concatenating 3 8bit input groups. These 24 bits are then treated as 4 concatenated 6-bit groups, each of which is translated into a single digit in the base64 alphabet. When encoding a bit stream via the base64 encoding, the bit stream must be presumed to be ordered with the most-significant-bit first. That is, the first bit in the stream will be the high-order bit in the first 8bit byte, and the eighth bit will be the low-order bit in the first 8bit byte, and so on.

Each 6-bit group is used as an index into an array of 64 printable characters. The character referenced by the index is placed in the output string. These characters, identified in Table 1, below, are selected so as to be universally representable, and the set excludes characters with particular significance to SMTP (e.g., ".", CR, LF) and to the multipart boundary delimiters defined in RFC 2046 (e.g., "-").

Table 1: The Base64 Alphabet

| Value | Encoding | Value | Encoding | Value | Encoding | Value | Encoding |
|---|---|---|---|---|---|---|---|
| 0 | A | 17 | R | 34 | i | 51 | z |
| 1 | B | 18 | S | 35 | j | 52 | 0 |
| 2 | C | 19 | T | 36 | k | 53 | 1 |
| 3 | D | 20 | U | 37 | l | 54 | 2 |
| 4 | E | 21 | V | 38 | m | 55 | 3 |
| 5 | F | 22 | W | 39 | n | 56 | 4 |
| 6 | G | 23 | X | 40 | o | 57 | 5 |
| 7 | H | 24 | Y | 41 | p | 58 | 6 |
| 8 | I | 25 | Z | 42 | q | 59 | 7 |
| 9 | J | 26 | a | 43 | r | 60 | 8 |
| 10 | K | 27 | b | 44 | s | 61 | 9 |
| 11 | L | 28 | c | 45 | t | 62 | + |
| 12 | M | 29 | d | 46 | u | 63 | / |
| 13 | N | 30 | e | 47 | v | | |
| 14 | O | 31 | f | 48 | w | (pad) | = |
| 15 | P | 32 | g | 49 | x | | |
| 16 | Q | 33 | h | 50 | y | | |

The encoded output stream must be represented in lines of no more than 76 characters each. All line breaks or other characters not found in Table 1 must be ignored by decoding software. In base64 data, characters other than those in Table 1, line breaks, and other white space probably indicate a transmission error, about which a warning message or even a message rejection might be appropriate under some circumstances.

Special processing is performed if fewer than 24 bits are available at the end of the data being encoded. A full encoding quantum is always completed at the end of a body. When fewer than 24 input bits are available in an input group, zero bits are added (on the right) to form an integral number of 6-bit groups. Padding at the end of the data is performed using the "=" character. Since all base64 input is an integral number of octets, only the following cases can arise: (1) the final quantum of encoding input is an integral multiple of 24 bits; here, the final unit of encoded output will be an integral multiple of 4 characters with no "=" padding, (2) the final quantum of encoding input is exactly 8 bits; here, the final unit of encoded output will be two characters followed by two "=" padding characters, or (3) the final quantum of encoding input is exactly 16 bits; here, the final unit of encoded output will be three characters followed by one "=" padding character.

Because it is used only for padding at the end of the data, the occurrence of any "=" characters may be taken as evidence that the end of the data has been reached (without truncation in transit). No

such assurance is possible, however, when the number of octets transmitted was a multiple of three and no "=" characters are present.

Any characters outside of the base64 alphabet are to be ignored in base64-encoded data.

Care must be taken to use the proper octets for line breaks if base64 encoding is applied directly to text material that has not been converted to canonical form. In particular, text line breaks must be converted into CRLF sequences prior to base64 encoding. The important thing to note is that this may be done directly by the encoder rather than in a prior canonicalization step in some implementations.

NOTE: There is no need to worry about quoting potential boundary delimiters within base64-encoded bodies within multipart entities because no hyphen characters are used in the base64 encoding.

## 7. Content-ID Header Field

In constructing a high-level user agent, it may be desirable to allow one body to make reference to another. Accordingly, bodies may be labelled using the "Content-ID" header field, which is syntactically identical to the "Message-ID" header field:

```
id := "Content-ID" ":" msg-id
```

Like the Message-ID values, Content-ID values must be generated to be world-unique.

The Content-ID value may be used for uniquely identifying MIME entities in several contexts, particularly for caching data referenced by the message/external-body mechanism. Although the Content-ID header is generally optional, its use is MANDATORY in implementations which generate data of the optional MIME media type "message/external-body". That is, each message/external-body entity must have a Content-ID field to permit caching of such data.

It is also worth noting that the Content-ID value has special semantics in the case of the multipart/alternative media type. This is explained in the section of RFC 2046 dealing with multipart/alternative.

## 8. Content-Description Header Field

The ability to associate some descriptive information with a given body is often desirable. For example, it may be useful to mark an "image" body as "a picture of the Space Shuttle Endeavor." Such text may be placed in the Content-Description header field. This header field is always optional.

```
description := "Content-Description" ":" *text
```

The description is presumed to be given in the US-ASCII character set, although the mechanism specified in RFC 2047 may be used for non-US-ASCII Content-Description values.

## 9. Additional MIME Header Fields

Future documents may elect to define additional MIME header fields for various purposes. Any new header field that further describes the content of a message should begin with the string "Content-" to allow such fields which appear in a message header to be distinguished from ordinary RFC 822 message header fields.

```
MIME-extension-field := <Any RFC 822 header field which
                         begins with the string
                         "Content-">
```

## 10. Summary

Using the MIME-Version, Content-Type, and Content-Transfer-Encoding header fields, it is possible to include, in a standardized way, arbitrary types of data with RFC 822 conformant mail messages. No restrictions imposed by either RFC 821 or RFC 822 are violated, and care has been taken to avoid problems caused by additional restrictions imposed by the characteristics of some Internet mail transport mechanisms (see RFC 2049).

The next document in this set, RFC 2046, specifies the initial set of media types that can be labelled and transported using these headers.

## 11. Security Considerations

Security issues are discussed in the second document in this set, RFC 2046.

## 12. Authors' Addresses

For more information, the authors of this document are best contacted
via Internet mail:

Ned Freed
Innosoft International, Inc.
1050 East Garvey Avenue South
West Covina, CA 91790
USA

Phone: +1 818 919 3600
Fax: +1 818 919 3614
EMail: ned@innosoft.com

Nathaniel S. Borenstein
First Virtual Holdings
25 Washington Avenue
Morristown, NJ 07960
USA

Phone: +1 201 540 8967
Fax: +1 201 993 3032
EMail: nsb@nsb.fv.com

MIME is a result of the work of the Internet Engineering Task Force Working Group on RFC 822 Extensions. The chairman of that group, Greg Vaudreuil, may be reached at:

Gregory M. Vaudreuil
Octel Network Services
17080 Dallas Parkway
Dallas, TX 75248-1905
USA

EMail: Greg.Vaudreuil@Octel.Com

## Appendix A -- Collected Grammar

This appendix contains the complete BNF grammar for all the syntax specified by this document.

By itself, however, this grammar is incomplete. It refers by name to several syntax rules that are defined by RFC 822. Rather than reproduce those definitions here, and risk unintentional differences between the two, this document simply refers the reader to RFC 822 for the remaining definitions. Wherever a term is undefined, it refers to the RFC 822 definition.

```
attribute := token
             ; Matching of attributes
             ; is ALWAYS case-insensitive.

composite-type := "message" / "multipart" / extension-token

content := "Content-Type" ":" type "/" subtype
           *(";" parameter)
           ; Matching of media type and subtype
           ; is ALWAYS case-insensitive.

description := "Content-Description" ":" *text

discrete-type := "text" / "image" / "audio" / "video" /
                 "application" / extension-token

encoding := "Content-Transfer-Encoding" ":" mechanism

entity-headers := [ content CRLF ]
                  [ encoding CRLF ]
                  [ id CRLF ]
                  [ description CRLF ]
                  *( MIME-extension-field CRLF )

extension-token := ietf-token / x-token

hex-octet := "=" 2(DIGIT / "A" / "B" / "C" / "D" / "E" / "F")
             ; Octet must be used for characters > 127, =,
             ; SPACEs or TABs at the ends of lines, and is
             ; recommended for any character not listed in
             ; RFC 2049 as "mail-safe".

iana-token := <A publicly-defined extension token. Tokens
               of this form must be registered with IANA
               as specified in RFC 2048.>
```

```
ietf-token := <An extension token defined by a
               standards-track RFC and registered
               with IANA.>

id := "Content-ID" ":" msg-id

mechanism := "7bit" / "8bit" / "binary" /
             "quoted-printable" / "base64" /
             ietf-token / x-token

MIME-extension-field := <Any RFC 822 header field which
                         begins with the string
                         "Content-">

MIME-message-headers := entity-headers
                        fields
                        version CRLF
                        ; The ordering of the header
                        ; fields implied by this BNF
                        ; definition should be ignored.

MIME-part-headers := entity-headers
                     [fields]
                     ; Any field not beginning with
                     ; "content-" can have no defined
                     ; meaning and may be ignored.
                     ; The ordering of the header
                     ; fields implied by this BNF
                     ; definition should be ignored.

parameter := attribute "=" value

ptext := hex-octet / safe-char

qp-line := *(qp-segment transport-padding CRLF)
           qp-part transport-padding

qp-part := qp-section
           ; Maximum length of 76 characters

qp-section := [*(ptext / SPACE / TAB) ptext]

qp-segment := qp-section *(SPACE / TAB) "="
              ; Maximum length of 76 characters

quoted-printable := qp-line *(CRLF qp-line)
```

```
safe-char := <any octet with decimal value of 33 through
             60 inclusive, and 62 through 126>
             ; Characters not listed as "mail-safe" in
             ; RFC 2049 are also not recommended.

subtype := extension-token / iana-token

token := 1*<any (US-ASCII) CHAR except SPACE, CTLs,
            or tspecials>

transport-padding := *LWSP-char
                     ; Composers MUST NOT generate
                     ; non-zero length transport
                     ; padding, but receivers MUST
                     ; be able to handle padding
                     ; added by message transports.

tspecials :=  "(" / ")" / "<" / ">" / "@" /
              "," / ";" / ":" / "\" / <">
              "/" / "[" / "]" / "?" / "="
              ; Must be in quoted-string,
              ; to use within parameter values

type := discrete-type / composite-type

value := token / quoted-string

version := "MIME-Version" ":" 1*DIGIT "." 1*DIGIT

x-token := <The two characters "X-" or "x-" followed, with
            no  intervening white space, by any token>
```