Groupe de travail Réseau
Request for Comments : 2045
RFC rendues obsolètes : 1521, 1522, 1590
Catégorie : Sur la voie de la normalisation

N. Freed, Innosoft
N. Borenstein, First Virtual
novembre 1996
Traduction Claude Brière de L'Isle

Extensions multi-objets de messagerie Internet (MIME) partie une : format des corps de message Internet

Statut du présent mémoire
Le présent document spécifie un protocole de l'Internet en cours de normalisation pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Protocoles officiels de l'Internet" (STD 1) pour voir l'état de normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.
(La présente traduction incorpore les errata exitants 512, 2589, et 3442.)

Résumé
Le STD 11, RFC 822, définit un protocole de représentation de message qui spécifie un nombre considérable de détails sur les en-têtes de message US-ASCII, et laisse le contenu du message, ou corps de message, comme du texte US-ASCII plat. Le présent ensemble de documents, appelé collectivement "extensions multi objets de messagerie Internet (MIME, *Multipurpose Internet Mail Extensions*) redéfinit le format des messages pour permettre :
(1) des corps de messages textuels dans des jeux de caractères autres que l'US-ASCII,
(2) un ensemble extensible de différents formats pour des corps de message non textuels,
(3) des corps de mesage multi parties, et
(4) des informations d'en-tête textuelles dans des jeux de caractères autres que l'US-ASCII.

Ces documents se fondent sur des travaux antérieurs documentés dans la RFC 934, STD 11, et dans la RFC 1049, mais les étendent et les révisent. Comme la RFC 822 en dit si peu sur les corps de message, ces documents sont largement orthogonaux (plutôt qu'une révision) à la RFC 822.

Ce premier document spécifie les divers en-têtes utilisés pour décrire la structure des messages MIME. Le second document, RFC 2046, définit la structure générale du système de support MIME et définit un jeu initial de types de supports. Le troisième document, RFC 2047, décrit des extensions à la RFC 822 pour permettre des données de texte non US-ASCII dans les champs d'en-tête de messagerie Internet. Le quatrième document, RFC 2048, spécifie diverses procédures d'enregistrement par l'IANA pour les facilités relatives à MIME. Le cinquième et dernier document, RFC 2049, décrit les critères de conformité à MIMIE ainsi que des exemple illustrant les formats de message MIME, les remerciements et la bibliographie.

Ces documents sont des révisions des RFC 1521, 1522, et 1590, qui étaient eux-mêmes des révisions des RFC 1341 et 1342. Un appendice de la RFC 2049 décrit les différences et changements par rapport aux précédentes versions.

Table des matières

1. Introduction

Depuis sa publication en 1982, la RFC 822 a défini le format standard des messages éléctroniques textuels sur l'Internet. Son succès a été tel que le format de la RFC 822 a été adopté, en tout ou en partie, bien au delà des confins de l'Internet et du transport SMTP Internet défini par la RFC 821. Comme le format a eu une utilisation plus large, un certain nombre de limitations se sont révélées de plus en plus restrictives pour la communauté des utilisateurs.

La RFC 822 était destinée à spécifier un format pour les messages de texte. À ce titre, les messages non textuels, comme les messages multimédia qui pouvaient inclure de l'audio ou des images, ne sont simplement pas mentionnés. Même dans le cas de texte, cependant, la RFC 822 est inadéquate pour les besoins des utilisateurs de messagerie électronique dont les langages exigent l'utilisation de jeux de caractères plus riches que l'US-ASCII. Comme la RFC 822 ne spécifie pas de mécanismes pour la messagerie qui contient de l'audio, de la vidéo, du texte d'un langage asiatique, ou même du texte dans la plupart des langues européennes, des spécifications supplémentaires sont nécessaires.

Une des limitations notables des systèmes de messagerie fondés sur les RFC 821/822 est le fait qu'ils limitent les contenus des messages électroniques à des lignes relativement courtes (par exemple 1000 caractères ou moins [RFC0821]) de 7bit US-ASCII. Cela force les utilisateurs à convertir toutes les données non textuelles qu'ils peuvent souhaiter envoyer en caractères d'octets à sept bits représentables comme de l'US-ASCII imprimables avant d'invoquer un agent d'utilisateur (UA, *User Agent)* un programme avec lequel les utilisateurs humains envoient et reçoivent la messagerie) de messagerie local. Des exemples de tels codages actuellement utilisés dans l'Internet incluent du pur hexadécimal, uuencode, le schéma en base 64 3 dans 4 spécifié dans la RFC 1421, la représentation "Andrew Toolkit" [ATK], et de nombreux autres.

Les limitations de la messagerie de la RFC 822 deviennent encore plus apparentes lorsque des passerelles sont conçues pour permettre l'échange de messages électroniques entre des hôtes de la RFC 822 et des hôtes X.400. X.400 [X400] spécifie des mécanismes pour l'inclusion de matériel non textuel au sein des messages électroniques. Les normes actuelles pour la transposition de messages X.400 en messages de la RFC 822 spécifient soit que le matériel X.400 non textuel doit être converti en (en non codé en) format de texte IA5Text, soit qu'il doit être éliminé, en notifiant l'utilisateur RFC 822 que l'élimination s'est produite. Ceci est clairement indésirable, car les informations qu'un utilisateur peut souhaiter recevoir sont perdues. Même si un agent d'utilisateur peut n'avoir pas la capacité de traiter le matériel non textuel, l'utilisateur peut avoir un mécanisme externe à l'UA qui peut extraire des informations utiles du matériel. De plus, cela ne tient pas compte du fait que le message peut finalement revenir par une passerelle à un système de traitement de message X.400 (c'est-à-dire que le message X.400 est "tunnelé" à travers la messagerie Internet) et que les informations non textuelles redeviendraient tout à fait utiles.

Le présent document décrit plusieurs mécanismes qui se combinent pour résoudre la plupart de ces problèmes sans introduire de sérieuses incompatibilités avec le monde existant de la messagerie RFC 822. En particulier, il décrit :

(1) Un champ d'en-tête MIME-Version, qui utilise un numéro de version pour déclarer qu'un message est conforme à MIME et permet aux agents de traitement de messagerie de distinguer de tels messages de ceux générés par des logiciels plus anciens ou non conformes, qui sont supposés ne pas avoir un tel champ.

(2) Un champ d'en-tête Content-Type, généralisé à partir de la RFC 1049, qui peut être utilisé pour spécifier le type de support et les sous type des données dans le corps d'un message et de pleinement spécifier la représentation native (en

forme canonique) de telles données.

(3) Un champ d'en-tête Content-Transfer-Encoding, qui peut être utilisé pour spécifier la transformation de codage qui a été appliquée au corps et le domaine du résultat. Les transformations de codage autres que la transformation d'identité sont généralement appliquées aux données afin de leur permettre de passer à travers les mécanismes de transport de messagerie qui peuvent avoir des limitations de données ou de jeu de caractères.

(4) Deux champs d'en-tête supplémentaires qui peuvent être utilisés pour décrire plus en détails les données dans un corps, les champs d'en-tête Content-ID et Content-Description.

Tous les champs d'en-tête définis dans le présent document sont soumis aux règles de syntaxe générales pour les champs d'en-tête spécifiés dans la RFC 822. En particulier, tous ces champs d'en-tête peuvent inclure des commentaires de la RFC 822, qui n'ont pas de contenu sémantique et devraient être ignorés durant le traitement MIME.

Finalement, pour spécifier et promouvoir l'interopérabilité, la RFC 2049 fournit une déclaration d'applicabilité de base pour un sous ensemble des mécanismes ci-dessus qui définit un niveau minimal de "conformité" avec ce document.

Note historique : plusieurs des mécanismes décrits dans cet ensemble de documents peuvent sembler un peu étranges ou même baroques à première vue. Il est important de noter que la compatibilité avec les normes existantes ET la robustesse de la pratique existante ont été les deux priorités du groupe de travail qui a développé cet ensemble de documents. En particulier, la compatibilité a toujours pris le pas sur l'élégance.

Prière de se référer à l'édition actuelle des "Normes officielles des protocoles de l'Internet" pour l'état de normalisation et le statut de ce protocole. La RFC 822 et le STD 3, RFC 1123 fournissent aussi des bases essentielles pour MIME car aucune mise en œuvre conforme à MIME ne peut les violer. De plus, plusieurs autres documents de RFC pour information vont intéresser la mise en œuvre de MIME, en particulier les RFC1344, RFC1345, et RFC1524.

2. Définitions, conventions, et grammaire BNFgénérique

Bien que les mécanismes spécifiés dans cet ensemble de documents soient tous décrits en prose, la plupart sont aussi décrits de façon formelle dans la notation BNF augmenté de la RFC 822. Cette notation devra être familière au lecteur pour qu'il puisse comprendre cet ensemble de documents, et l'explication complète de cette notation BNF augmenté se trouve dans la RFC0822.

Un partie du BNF augmenté de cet ensemble de documents fait une référence explicite aux règles de syntaxe définies dans la RFC 822. Une grammaire formelle complète est obtenue en combinant les appendices de grammaire collectés dans chaque document de cet ensemble avec le BNF de la RFC 822 plus les modifications à la RFC 822 définies dans la RFC1123 (qui change spécifiquement la syntaxe de "return", "date" et "mailbox").

Toutes les valeurs numériques et d'octet sont données en notation décimale dans cet ensemble de documents. Toutes les valeurs de type de supports, valeurs de sous type, et noms de paramètres définis sont insensibles à la casse. Cependant, les valeurs de paramètres sont sensibles à la casse sauf spécification contraire pour le paramètre concerné.

Note de formatage : les notes, telles que celle-ci, fournissent des informations non essentielles supplémentaires qui peuvent être sautées par le lecteur sans rien maquer d'essentiel. Le principal objet de ces notes non essentielles est de porter des informations sur les raisons de cet ensemble de documents, ou de placer ces documents dans le contexte historique ou évolutif approprié. De telles informations peuvent en particulier être sautées par ceux qui sont entièrement concentrés sur la construction d'une mise en œuvre conforme, mais peuvent être utiles à ceux qui souhaitent comprendre les raisons de certains choix de conception.

2.1 CRLF

Le terme CRLF, dans cet ensemble de documents, se réfère à la séquence d'octets correspondant aux deux caractères US-ASCII CR *(retour chariot)* (valeur décimale 13) et LF *(saut à la ligne)* (valeur décimale 10) qui, pris ensemble, dans cet ordre, notent un saut à la ligne dans la messagerie de la RFC 822.

2.2 Jeu de caractères

Le terme "jeu de caractères" est utilisé dans MIME pour se référer à une méthode de conversion d'une séquence d'octets en une séquence de caractères. Noter que la conversion inconditionnelle et sans ambiguïté dans l'autre direction n'est pas exigée, en ce que tous les caractères ne peuvent pas être représentables par un certain jeu de caractères et un jeu de

caractères peut fournir plus d'une séquence d'octets pour représenter une séquence de caractères particulière.

Cette définition est destinée à permettre diverses sortes de codage de caractères, des simples transpositions d'un seul tableau comme l'US-ASCII à des méthodes complexes de commutation de tableaux comme celles qui utilisent les techniques de ISO 2022, d'être utilisés comme jeux de caractères. Cependant, la définition associée à un nom de jeu de caractères MIME doit spécifier pleinement la transposition à effectuer. En particulier, l'utilisation d'informations de profilage externes pour déterminer la transposition exacte n'est pas permise.

Note : le terme "jeu de caractères" décrivait à l'origine des schémas directs comme l'US-ASCII et l'ISO-8859-1 qui ont une simple transposition biunivoque d'un seul octet en un seul caractère. Les jeux de caractères codés sur plusieurs octets et les techniques de commutation rendent la situation plus complexe. Par exemple, certaines communautés utilisent le terme de "codage de caractère" pour ce que MIME appelle un "jeu de caractères", tout en utilisant la phrase "jeu de caractères codés " pour noter une transposition abstraite des entiers (pas des octets) en caractères.

2.3 Message

Le terme "message", sans autre qualifiication, signifie soit un message de la RFC0822 (complet ou "de niveau supérieur") transféré sur un réseau, soit un message encapsulé dans un corps de type "message/rfc822" ou "message/partiel".

2.4 Entité

Le terme "entité", se réfère spécifiquement aux champs d'en-tête définis par MIME et au contenu d'un message ou d'une des parties dans le corps d'une entité multi parties. La spécification de telles entités est l'essence de MIME. Comme le contenu d'une entité est souvent appelé le "corps", on peut parler du corps d'une entité. Toutes sortes de champs peuvent être présents dans l'en-tête d'une entité, mais seuls les champs dont les noms commencent par "content-" ont en fait une signification en rapport avec MIME. Noter que ceci N'implique PAS qu'il n'ont pas du tout de signification -- une entité qui est aussi un message a des champs d'en-tête non MIME dont la signification est définie par la RFC 822.

2.5 Partie de corps

Le terme "partie de corps" se réfère à une entité à l'intérieur d'une entité multi parties.

2.6 Corps

Le terme "corps", sans autre qualification, signifie le corps d'une entité, qui est le corps d'un message ou une partie de corps.

Note : Les quatre définitions précédentes sont clairement circulaires. Ceci est inévitable, car la structure globale d'un message MIME est bien sûr récurrente.

2.7 Données en 7bits

"données en 7bits" se réfère aux données qui sont toutes représentées comme des lignes relativement courtes de 998 octets ou moins entre les séquences CRLF de séparation de lignes [RFC0821]. Aucun octet d'une valeur décimale supérieure à 127 n'est admise et ni des octets "NUL" (octets d'une valeur décimale 0). Les octets CR (valeur décimale 13) et LF (valeur décimale 10) ne se présentent qu'au titre de séquences CRLF de séparation de lignes.

2.8 Données en 8bits

"données en 8bits" se réfère aux données qui sont toutes représentées comme des lignes relativement courtes de 998 octets ou moins entre les séqeunces CRLF de séparation de lignes [RFC0821], mais des octets avec des valeurs décimales supérieures à 127 peuvent être utilisés. Comme avec les "données à 7bits" les octets CR et LF ne se produisent qu'au titre de séquences CRLF de séparation de lignes et aucun NUL n'est admis.

2.9 Données binaires

"Données binaires" se réfère aux données où toute séquence d'octets quels qu'ils soient est admise.

2.10 Lignes

Les "lignes" sont définies comme des séquences d'octets séparées par une séquence de CRLF. Ceci est cohérent aussi bien avec la RFC 821 que la RFC 822. "Lignes" se réfère seulement à une unité de données dans un message, qui peut correspondre ou non à quelque chose qui est en fait affiché par un agent d'utilsateur.

3. Champs d'en-tête MIME

MIME définit un certain nombre de nouveaux champs d'en-tête de la RFC 822 qui sont utilisés pour décrire le contenu d'une entité MIME. Ces champs d'en-tête se produisent au moins dans deux contextes :
(1) Au titre d'un en-tête régulier de message de la RFC 822.
(2) Dans un en-tête de partie de corps MIME au sein d'une construction multi parties.

La définition formelle de ces champs d'en-tête est la suivante :

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

MIME-message-headers := version de champs entity-headers CRLF
                  ; L'ordre des champs d'en-tête impliqué par cette définition de BNF devrait être ignoré.

MIME-part-headers := entity-headers [ champs ]
                  ; Tout champ qui ne commence pas par "content-" peut n'avoir pas de signification définie et peut être ignoré.
                  ; L'ordre des champs d'en-tête impliqué par cette définition de BNF devrait être ignoré.
```

La syntaxe des divers champs d'en-tête spécifiques de MIME va être décrite dans les sections suivantes.

4. Champ d'en-tête MIME-Version

Depuis que la RFC 822 a été publiée en 1982, il y a réellement eu un seul format standard pour les messages Internet, et il n'y a eu que peu de besoins perçus de déclarer le format standard utilisé. Le présent document est une spécification indépendante qui complète la RFC 822. Bien que les extensions de ce document aient été définies d'une façon telle qu'elle soit compatible avec la RFC 822, il y a des circonstances dans lesquelles il peut être souhaitable qu'un agent de traitement de messagerie sache si un message a été composé en tenant compte du nouveau standard.

Donc, ce document définit un nouveau champ d'en-tête, "MIME-Version", qui est à utiliser pour déclarer la version de la norme de format de corps de message Internet utilisée.

Les messages composés conformément au présent document DOIVENT inclure un tel champ d'en-tête, avec le texte suivant mot à mot : MIME-Version: 1.0

La présence de ce champ d'en-tête est une assertion que le message a été composé en conformité au présent document.

Comme il est possible qu'un futur document étende à nouveau le format de message standard, un BNF formel est donné pour le contenu du champ MIME-Version :

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

Donc, les futures spécifications de format, qui pourraient remplacer ou étendre "1.0", sont obligées d'avoir deux champs d'entiers, séparés par un point. Si un message est reçu avec une valeur de MIME-version autre que "1.0", elle ne peut pas être supposée se conformer au présent document.

Noter que le champ d'en-tête MIME-Version est exigé au niveau supérieur d'un message. Il n'est pas exigé pour chaque partie de corps d'une entité multi parties. Il est exigé pour les en-têtes incorporés d'un corps de type "message/rfc822" ou "message/partial" si et seulement si le message incorporé prétend lui-même être conforme à MIME.

Il n'est pas possible de spécifier complètement comment un lecteur de message peut se conformer à MIME tel que défini dans le présent document pour traiter un message qui pourrait arriver à l'avenir avec une valeur de MIME-Version autre que "1.0".

Il vaut aussi de noter que le contrôle de version pour un type spécifique de support n'est pas réalisé en utilisant le mécanisme MIME-Version. En particulier, certains formats (comme application/postscript) ont des conventions de numérotation de version qui sont internes au format de support. Lorsque il existe de telles conventions, MIME ne fait rien pour les outrepasser. Si il n'existe pas une telle convention, un type de support MIME utilise un paramètre "version" dans le champ "content-type" si nécessaire.

Note de mise en œuvre : Pour vérifier les valeurs de MIME-Version toutes les chaînes de commentaires de la RFC 822 qui sont présentes doivent être ignorées. En particulier, les quatre champs MIME-Version suivants sont équivalents :

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

En l'absence d'un champ MIME-Version, un agent d'utilisateur de messagerie receveur (qu'il soit ou non conforme aux exigences de MIME) peut facultativement choisir d'interpréter le corps du message selon des conventions locales. Beaucoup de ces conventions sont actuellement utilisées et on devrait noter qu'en pratique des messages non MIME peuvent contenir tout et n'importe quoi.

Il est impossible d'être certain qu'un message non MIME est en fait du pur texte dans le jeu de caractères US-ASCII car il peut bien être un message qui, en utilisant un ensemble de conventions locales non standard qui imitent MIME, inclut du texte dans un autre jeu de caractères ou des données non textuelles présentées d'une manière qui ne peut pas être automatiquement reconnue (par exemple, un fichier tar UNIX uuencodé compressé).

5. Champ d'en-tête Content-Type

L'objet du champ Content-Type est de décrire les données contenues dans le corps assez complètement pour que l'agent d'utilisateur receveur puisse prendre un agent ou mécanisme approprié pour présenter les données à l'utilisateur, ou autrement traiter les données de façon appropriée. La valeur dans ce champ est appelée un type de support

Note historique : Le champ d'en-tête Content-Type a été d'abord défini dans la RFC1049. La RFC1049 utilisait une syntaxe plus simple et moins puissante, mais qui est largement compatible avec le mécanisme présenté ici.

Le champ d'en-tête Content-Type spécifie la nature des données dans le corps d'une entité en donnant des identifiants de type et sous type de supports, et en fournissant des informations auxiliaires qui peuvent être exigées pour certains types de supports. Après les noms de type et sous type de supports, le reste du champ d'en-tête est simplement un ensemble de paramètres, spécifiés dans une notation attribut=valeur. L'ordre des paramètres n'est pas significatif.

En général, le type de support de niveau supérieur est utilisé pour déclarer le type général de données, tandis que le sous type spécifie un format spécifique pour ce type de données. Donc, un type de support de "image/xyz" est suffisant pour dire à un agent d'utilisateur que les données sont une image, même si l'agent d'utilisateur n'a pas connaissance du format d'image "xyz" spécifique. De telles informations peuvent être utilisées, par exemple, pour décider de montrer ou non à un utilisateur les données brutes d'un sous type non reconnu -- une telle action peut être raisonnable pour des sous types de texte non reconnus, mais pas pour des sous types non reconnus d'image ou d'audio. Pour cette raison, les sous types enregistrés de texte, image, audio, et vidéo ne devraient pas contenir d'informations incorporées qui sont en fait d'un type différent. De tels formats composés devraient être représenté en utilisant les types "multipart" ou "application".

Les paramètres sont des modificateurs du sous type de support, et à ce titre n'affectent pas fondamentalement la nature du contenu. L'ensemble de paramètres significatifs dépend du type et sous type de support. La plupart des paramètres sont associés à un seul sous type spécifique. Cependant, un certain type de support de niveau supérieur peut définir des paramètres qui sont applicables à tout sous type de ce type. Des paramètres peuvent être exigés par le type ou sous type de contenu qui les définit ou ils peuvent être facultatifs. Les mises en œuvre de MIME doivent ignorer tout paramètre dont elles ne reconnaissent pas le nom.

Par exemple, le paramètre "charset" *(jeu de caractères)* est applicable à tout sous type de "text", tandis que le paramètre "boundary" *(limite)* est exigé pour tout sous type du type de support "multipart" .

Il n'y a pas de paramètre de signification globale qui s'applique à tous les types de supports. Des mécanismes vraiment globaux sont traités, dans le modèle MIME, par la définition de champs d'en-tête Content-* supplémentaires.

Un ensemble initial de sept types de support de niveau supérieur est défini dans la RFC 2046. Cinq d'entre eux sont des types discrets dont le contenu est essentiellement opaque pour ce qui concerne le traitement par MIME. Les deux autres sont des types composites dont le contenu exige un traitement supplémentaire de la part des processeurs MIME.

Cet ensemble de types de supports de niveau supérieur est destiné à être complet dans sa substance. Il est envisagé que des ajouts au plus grand ensemble des types pris en charge puissent être accomplis généralement par la création de nouveaux sous types de ces types initiaux. À l'avenir, plus de types de niveau supérieur pourraient être définis par une extension sur la voie de la normalisation à la présente norme. Si un autre type de niveau supérieur doit être utilisé pour une raison quelconque, il doit recevoir un nom commençant par "X-" pour indiquer son statut non normalisé et pour éviter un potentiel conflit avec un futur nom officiel.

5.1 Syntaxe du champ d'en-tête Content-Type

Dans la notation de BNF augmenté de la RFC 822, une valeur de champ d'en-tête Content-Type est définie comme suit :

content := "Content-Type" ":" type "/" sous type *(";" paramètre)
; la confrontation des type et sous type de supports est TOUJOURS insensible à la casse.

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 := <jeton d'extension défini par une RFC sur la voie de la normalisation et enregistrée par l'IANA.>

x-token := <les deux caractères "X-" ou "x-" suivis, sans espace intercalée, par un jeton>

subtype := extension-token / iana-token

iana-token := <jeton d'extension dont la définition est publiée. Les jetons de cette forme doivent être enregistrés par l'IANA comme spécifié dans la RFC 2048.>

parameter := attribut "=" valeur

attribut := jeton ; la confrontation des attributs est TOUJOURS insensible à la casse.

valeur := jeton / quoted-string

jeton := 1*<tout caractère (US-ASCII) sauf ESPACE, CTL, ou tspecials>

tspecials := "(" / ")" / "<" / ">" / "@" / "," / ";" / ":" / "\" / <"> / "/" / "[" / "]" / "?" / "="
; Doit être dans une chaîne entre guillemets *(quoted-string)* dans les valeurs de paramètres.

Noter que la définition de "tspecials" est la même que la définition de la RFC 822 de "specials" avec l'ajout de trois caractères "/", "?", et "=", et la suppression de ".".

Noter aussi qu'une spécification de sous type est OBLIGATOIRE -- elle ne doit pas être omise d'un champ d'en-tête Content-Type. À ce titre, il n'y a pas de sous types par défaut.

Les noms de type, sous type, et paramètre ne sont pas sensibles à la casse. Par exemple, TEXT, Text, et TeXt sont tous des types de supports de niveau supérieur équivalents. Les valeurs de paramètres sont normalement sensibles à la casse, mais parfois elles sont interprétées de façon insensible à la casse, selon l'utilisation prévue. (Par exemple, les limites de multi parties sont sensibles à la casse, mais le paramètre "access-type" pour message/External-body n'est pas sensible à la casse.)

Noter que la valeur d'un paramètre en chaîne entre guillemets n'inclut pas les guillemets. C'est -à-dire que les guillemets

dans une chaîne entre guillemets ne font pas partie de la valeur du paramètre, mais sont simplement utilisés pour délimiter cette valeur de paramètre. De plus, des commentaires sont permis en accord avec les règles de la RFC 822 pour les champs d'en-tête structurés. Donc les deux formes suivantes

Content-type: text/plain; charset=us-ascii (texte simple)
Content-type: text/plain; charset="us-ascii"

sont complètement équivalentes.

Au delà de cette syntaxe, la seule contrainte syntaxique sur la définition des noms de sous types est le désir d'éviter les conflits d'utilisation. C'est-à-dire, qu'il ne serait pas souhaitable d'avoir deux communautés différentes qui utilisent "Content-Type: application/foobar" pour signifier deux choses différentes. Le processus de définition de nouveaux sous types de supports n'est pas destiné à être un mécanisme pour imposer des restrictions, mais simplement un mécanisme pour rendre publics leur définition et usage. Il y a donc deux mécanismes acceptables pour définir les nouveaux sous types de supports :

1 Les valeurs privées (commençant par "X-") peuvent être définies de façon bilatérale entre deux agents coopérants sans enregistrement ou normalisation extérieure. De telles valeurs ne peuvent pas être enregistrées ou normalisées.
2 De nouvelles valeurs normalisées devraient être enregistrées par l'IANA comme décrit dans la RFC 2048.

Le second document de cet ensemble, la RFC 2046, définit l'ensemble initial des types de supports pour MIME.

5.2 Content-Type par défaut

Par défaut les messages de la RFC 822 sans un en-tête MIME Content-Type sont pris par ce protocole comme étant du texte pur dans le jeu de caractères US-ASCII, qui peut être spécifié explicitement comme :

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

Cette valeur par défaut est supposée si aucun Content-Type n'est spécifié. Il est aussi recommandé que cette valeur par défaut soit supposée quand un champ d'en-tête Content-Type syntaxiquement invalide est rencontré. En présence d'un champ d'en-tête MIME-Version et en l'absence de tout champ d'en-tête Content-Type, un agent d'utilisateur receveur peut aussi supposer que l'intention de l'envoyeur était du texte pur US-ASCII. Le texte pur US-ASCII peut encore être supposé en l'absence d'un champ d'en-tête MIME-Version ou en présence d'un champ d'en-tête Content-Type syntaxiquement invalide, mais l'intention de l'envoyeur pourrait avoir été différente.

6. Champ d'en-tête Content-Transfer-Encoding

De nombreux types de supports qui pourraient être utilement transportés via la messagerie électronique sont représentés, dans leur format "naturel", comme des données en caractères à 8 bits ou en données binaires. De telles données ne peuvent être transmises sur certains protocoles de transfert. Par exemple, la RFC 821 (SMTP) restreint les messages électroniques à être en données US-ASCII à 7 bits avec des lignes de pas plus de 1000 caractères incluant tout séparateur de ligne CRLF en queue.

Il est donc nécessaire de définir un mécanisme standard pour coder de telles données en un format de ligne courte à 7 bits. L'étiquetage approprié du matériel non codé dans des formats moins restrictifs pour une utilisation directe sur des transports moins restrictifs est aussi désirable. Le présent document spécifie que de tels codages seront indiqués par un nouveau champ d'en-tête "Content-Transfer-Encoding" *(codage de transfert de contenu)*. Ce champ n'a été défini par aucune norme antérieure.

6.1 Syntaxe de Content-Transfer-Encoding

La valeur du champ Content-Transfer-Encoding est un seul jeton qui spécifie le type de codage, comme précisé ci-dessous. Formellement :

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

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

Ces valeurs ne sont pas sensibles à la casse -- Base64 et BASE64 et bAsE64 sont tous équivalents. Un type de codage de 7BIT exige que le corps soit déjà dans une représentation de 7bit prête pour la messagerie. C'est la valeur par défaut -- c'est-à-dire, "Content-Transfer-Encoding: 7BIT" est supposé si le champ d'en-tête Content-Transfer-Encoding n'est pas présent.

6.2 Sémantique de Content-Transfer-Encoding

Ce seul jeton Content-Transfer-Encoding fournit en fait deux éléments d'informations. Il spécifie à quelle sorte de transformation de codage le corps a été soumis et donc quelle opération de décodage doit être utilisée pour le restaurer à sa forme d'origine, et il spécifie quel est le domaine du résultat.

La partie transformation de tout Content-Transfer-Encoding spécifie, explicitement ou implicitement, un seul algorithme de décodage bien défini, qui pour toute séquence d'octets codée la transforme en la séquence d'octets d'origine qui a été codée, ou montre qu'elle est illégale comme séquence codée. Les transformations Content-Transfer-Encoding ne dépendent jamais d'informations de profil externe supplémentaires pour un fonctionnement approprié. Noter qu'alors que les décodeurs doivent produire un seul résultat bien défini pour un codage valide, aucune restriction de cette sorte n'existe pour les codeurs : coder une certaine séquence d'octets en séquences codées équivalentes différentes est parfaitement légal.

Trois transformations sont actuellement définies : l'identique, le codage "quoted-printable", et le codage "base64". Les domaines sont "binary", "8bit" et "7bit".

Les valeurs de Content-Transfer-Encoding "7bit", "8bit", et "binary" signifient toutes que la transformation de codage identique (c'est-à-dire NO) a été effectuée. À ce titre, elles servent simplement d'indicateurs du domaine des données de corps, et fournissent des informations utiles sur la sorte de codage qui pourrait être nécessaire pour une transmission dans un certain système de transport. Les termes "données 7bit", "données 8bit", et "données binaires" sont tous définis à la Section 2.

Les codages quoted-printable et base64 transforment leur entrée provenant d'un domaine arbitraire en du matériel dans la gamme "7bit", le rendant donc sûr pour le transport sur des transports qui connaissent des restrictions. La définition spécifique des transformations est donnée ci après.

L'étiquette appropriée de Content-Transfer-Encoding doit toujours être utilisée. L'étiquetage de données non codées contenant des caractères en 8bit comme du "7bit" n'est pas permis, ni ne l'est l'étiquetage de données non codées non calées sur la ligne comme autre chose que "binary".

À la différence des sous types de supports, une prolifération de valeurs de Content-Transfer-Encoding est à la fois indésirable et inutile. Cependant, établir seulement une transformation dans le domaine "7bit" ne semble pas possible. Il y a un compromis à faire entre le désir d'un codage compact et efficace de données largement binaires et le désir d'un codage assez lisible des données qui est principalement, mais pas entièrement, 7bit. Pour cette raison, au moins deux mécanismes de codage sont nécessaires : un codage plus ou moins lisible (quoted-printable) et un codage "dense" ou "uniforme" (base64).

Le transport de messagerie pour des données non codées en 8bit est défini dans la RFC 1652. Au moment de la publication initiale de ce document, il n'y avait pas de transport normalisé de messagerie Internet pour lequel il serait légitime d'inclure des données binaires non codées dans les corps de messages. Donc il n'y a pas de circonstances dans lesquelles le Content-Transfer-Encoding "binary" est en fait valide dans la messagerie Internet. Cependant, pour le cas où le transport de messagerie binaire deviendrait une réalité dans la messagerie Internet, ou quand MIME est utilisé en conjonction avec un autre mécanisme de transport de messagerie à capacité binaire, les corps binaires doivent être étiquetés comme tels en utilisant ce mécanisme.

Note : les cinq valeurs définies pour le champ Content-Transfer-Encoding n'ont pas d'implication sur le type de support autre que l'algorithme par lequel il a été codé ou les exigences du système de transport si il n'est pas codé.

6.3 Nouveaux Content-Transfer-Encoding

Les mises en œuvre peuvent, si nécessaire, définir des valeurs privées de Content-Transfer-Encoding, mais doivent utiliser un jeton x-, qui est un nom précédé de "X-", pour indiquer son statut non standard, par exemple, "Content-Transfer-Encoding: x-mon-nouveau-codage". Les valeurs normalisées de Content-Transfer-Encoding supplémentaires doivent être spécifiées par une RFC sur la voie de la normalisation. Les exigences auxquelles de telles spécifications doivent satisfaire sont données dans la RFC 2048. À ce titre tout l'espace de noms de content-transfer-encoding sauf ceux commençant par "X-" est explicitement réservé à l'IETF pour de futures utilisations.

À la différence des types et sous types de supports, la création de nouvelles valeurs de Content-Transfer-Encoding est FORTEMENT déconseillée, car cela semble entraver l'interopérabilité pour un faible bénéfice potentiel.

6.4 Interprétation et utilisation

Si un champ d'en-tête Content-Transfer-Encoding apparaît au titre d'un en-tête de message, il s'applique à tout le corps de ce message. Si un champ d'en-tête Content-Transfer-Encoding apparaît au titre des en-têtes d'une entité, il ne s'applique qu'au corps de cette entité. Si une entité est du type "multipart" il n'est pas permis au Content-Transfer-Encoding d'avoir une valeur autre que "7bit", "8bit" ou "binary". Des restrictions encore plus sévères s'appliquent à certains sous types du type "message".

Il devrait être noté que la plupart des types de supports sont définis en termes d'octets plutôt que de bits, de sorte que les mécanismes décrits ici sont des mécanismes pour coder des flux d'octets arbitraires, et non des flux de bits. Si un flux de bits est à coder via un de ces mécanismes, il doit d'abord être converti en un flux d'octets de 8bit en utilisant l'ordre normalisé des bits du réseau ("gros boutien") dans lequel les premiers bits d'un flux deviennent les bits de poids fort d'un octet en 8bit. Un flux de bits qui ne se termine pas sur une limite de 8bit doit être bourré avec des zéros. La RFC 2046 donne un mécanisme pour noter l'ajout d'un tel bourrage dans le cas du type de support application/octet-stream, qui a un paramètre "padding" *(bourrage)*.

Les mécanismes de codage définis ici codent explicitement toutes les données en US-ASCII. Donc, par exemple, supposons qu'une entité ait des champs d'en-tête tels que :

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

Ceci doit être interprété comme signifiant que le corps est un codage base64 US-ASCII des données qui étaient à l'origine en ISO-8859-1, et vont être à nouveau dans ce jeu de caractères après le décodage.

Certaines valeurs de Content-Transfer-Encoding ne peuvent être utilisées que sur certains types de supports. En particulier, il est EXPRESSEMENT INTERDIT d'utiliser des codages autres que "7bit", "8bit", ou "binary" avec tout type de support composite, c'est-à-dire un qui inclut de façon récurrente d'autres champs Content-Type. Actuellement, les seuls types de support composites sont "multipart" et "message". Tous les codages qui sont désirés pour des corps de type multipart ou message doivent être faits au niveau le plus interne, en codant le corps réel qui a besoin d'être codé.

On devrait aussi noter que, par définition, si une entité composite a une valeur de codage de transfert de "7bit", mais qu'une des entités encloses a une valeur moins restrictive comme "8bit", soit l'étiquetage externe "7bit" est une erreur, parce que des données 8bit sont incluses, soit l'étiquetage interne "8bit" a placé une contrainte inutilement élevée sur le système de transport parce que les données réellement incluses sont en fait sûres avec 7bit.

Note sur les restrictions de codage : bien que l'interdiction de l'utilisation des content-transfer-encoding sur des données de corps composite puisse sembler trop restrictive, elle est nécessaire pour empêcher des codages incorporés, dans lesquels des données sont passées plusieurs fois à travers un algorithme de codage, et doivent être décodées plusieurs fois afin d'être vues correctement. Les codages incorporés ajoutent une complexité considérable aux agents d'utilisateur : en plus des problèmes évidents d'efficacité de tels codages multiples, ils peuvent obscurcir la structure de base d'un message. En particulier, ils peuvent impliquer que plusieurs opérations de décodage sont nécessaires simplement pour découvrir quels types de corps contient un message. Interdire les codages incorporés peut compliquer la tâche de certaines passerelles de messagerie, mais cela semble moins problématique que l'effet des codages incorporés sur les agents d'utilisateur.

Toute entité avec un Content-Transfer-Encoding non reconnu doit être traitée comme si elle avait un Content-Type de "application/octet-stream", sans considération de ce que dit en fait le champ d'en-tête Content-Type.

Note sur les relations entre content-type et content-transfer-encoding : il peut sembler que le Content-Transfer-Encoding pourrait être déduit des caractéristiques du support à coder, ou, tout au moins, que certains Content-Transfer-Encoding pourraient être rendus obligatoires avec des types de supports spécifiques. Il y a plusieurs raisons pour que ce ne soit pas le cas. D'abord, étant donnés les divers types de transports utilisés pour la messagerie, certains codages peuvent être appropriés pour certaines combinaisons de types de supports et de transports mais pas pour d'autres. (Par exemple, dans un transport 8bit, aucun codage ne serait exigé pour le texte dans certains jeux de caractères, tandis que de tels codages sont clairement exigés pour le SMTP 7bit.) Ensuite, certains types de supports peuvent exiger différents types de codage de transfert dans des circonstances différentes. Par exemple, de nombreux corps PostScript peuvent consister entièrement en courtes lignes de données 7bit et donc n'exiger aucun codage. D'autres corps PostScript (en particulier ceux qui utilisent le mécanisme de codage binaire de PostScript niveau 2) peuvent n'être raisonnablement représentés qu'en utilisant un codage de transport binaire. Finalement, comme le

champ Content-Type est destiné à être un mécanisme de spécification à terminaison ouverte, une spécification stricte d'une association entre types de supports et codages couple effectivement la spécification d'un protocole d'application avec un transport spécifique de niveau inférieur. Ceci n'est pas souhaitable car les développeurs d'un type de support ne devraient pas être au courant de tous les transports en usage et de quelles sont leurs limitations.

6.5 Traduction de codages

Les codages quoted-printable et base64 sont conçus de telle sorte que la conversion entre eux soit possible. La seule question qui se pose dans une telle conversion est le traitement des coupures de ligne dans le résultat du codage quoted-printable. Lors de la conversion de quoted-printable en base64, une coupure de ligne dans la forme quoted-printable représente une séquence CRLF en forme canonique des données. Elle doit donc être convertie en un CRLF codé de façon correspondante dans la forme base64 des données. De façon similaire, une séquence CRLF dans la forme canonique des données obtenue après le décodage base64 doit être convertie en une coupure de ligne en quoted-printable, mais SEULEMENT lors de la conversion de données de texte.

6.6 Modèle de codage canonique

Il y a eu une certaine confusion, dans les précédentes versions de cette RFC, concernant le modèle où les données de messagerie électronique devaient être converties en forme canonique et codées, et en particulier comment ce processus allait affecter le traitement des CRLF, étant donné que la représentation des nouvelles lignes varie largement d'un système à l'autre, et des relations entre les content-transfer-encoding et les jeux de caractères. Un modèle canonique pour le codage est présenté dans la RFC 2049 pour cette raison.

6.7 Content-Transfer-Encoding en Quoted-Printable

Le codage Quoted-Printable est destiné à représenter des données qui consistent principalement en octets qui correspondent aux caractères imprimables dans le jeu de caractères US-ASCII. Il code les données d'une façon telle que les octets résultants ne seront probablement pas modifiés par le transport de messagerie. Si les données codées sont pour la plupart du texte US-ASCII, la forme codée des données reste largement reconnaissable par les humains. Un corps qui est entièrement en US-ASCII peut aussi être codé en Quoted-Printable pour assurer l'intégrité des données si le message passe à travers une passerelle qui traduit les caractères, et/ou va à la ligne.

Dans ce codage, les octets sont à représenter comme déterminé par les règles suivantes :

(1) Représentation générale en 8bit : tout octet, sauf un CR ou LF qui fait partie d'un CRLF de coupure de ligne de la forme canonique (standard) des données à coder, peut être représenté par un "=" suivi par une représentation hexadécimale de deux chiffres de la valeur de l'octet. Les chiffres de l'alphabet hexadécimal sont à cette fin "0123456789ABCDEF". Les lettres majuscules doivent être utilisées ; les lettres minuscules ne sont pas permises. Donc, par exemple, la valeur décimale 12 (espace US-ASCII) peut être représentée par "=0C", et la valeur décimale 61 (signe égal US-ASCII) peut être représentée par "=3D". Cette règle doit être suivie sauf quand la règle suivante permet un autre codage.

(2) Représentation littérale : les octets avec des valeurs décimales de 33 à 60 inclus, et de 62 à 126, inclus, PEUVENT être représentés comme les caractères US-ASCII qui correspondent à ces octets (respectivement de "point d'exclamation" à "moins que", et de "supérieur à" jusqu'à "tilde").

(3) Espaces : les octets des valeurs 9 et 32 PEUVENT être représentés par les caractères US-ASCII TAB (HT) et SPACE, respectivement, mais NE DOIVENT PAS être représentés ainsi à la fin d'une ligne codée. Tout caractère TAB (HT) ou SPACE sur une ligne codée DOIT donc être suivi sur cette ligne par un caractère imprimable. En particulier, un "=" à la fin d'une ligne codée, indiquant une coupure de ligne douce (voir la règle n° 5) peut suivre un ou plusieurs caractères TAB (HT) ou SPACE. Il s'ensuit qu'un octet de valeur décimale 9 ou 32 qui apparait à la fin d'une ligne codée doit être représenté conformément à la règle n° 1. Cette règle est nécessaire parce que certains programmes d'agent de transport de messagerie (MTA, *Message Transport Agent*) qui transportent les messages d'un utilisateur à un autre, ou effectuent une portion de ces transferts) sont connus pour bourrer les lignes de texte avec des SPACE, et d'autres sont connus pour supprimer les caractères "espace" de la fin d'une ligne. Donc, lors du décodage d'un corps Quoted-Printable, toutes les espaces blanches en fin de ligne doivent être supprimées, car elles auront nécessairement été ajoutées par des MTA intermédiaires.

(4) Coupures de ligne dures : une coupure de ligne dans un corps de texte, représentée par une séquence CRLF dans la forme canonique du texte, doit être représentée par une coupure de ligne (RFC 822) qui est aussi une séquence CRLF, dans le codage Quoted-Printable. Comme la représentation canonique des types de supports autres que "text" n'inclut

généralement pas la représentation des coupures de lignes comme des séquences CRLF, aucune coupure de ligne dure (c'est-à-dire une coupure de ligne destinée à être significative et être affichée à l'utilisateur) ne peut se produire dans le codage quoted-printable de ces types. Des séquences comme "=0D", "=0A", "=0A=0D" et "=0D=0A" vont bien sûr couramment apparaître dans des données non textuelles représentées en quoted-printable.

Noter que de nombreuses mises en œuvre peuvent choisir de coder la représentation locale de divers types de contenu directement plutôt que de convertir d'abord en forme canonique, de coder, et ensuite de reconvertir en représentation locale. En particulier, cela peut s'appliquer au matériel de texte pur sur des systèmes qui utilisent des conventions de nouvelle ligne autres que une séquence terminale de CRLF. Une telle optimisation de mise en œuvre est permise, mais seulement quand l'étape de canonisation-codage combinée est équivalente à effectuer les trois étapes séparément.

(5) Coupures de ligne douces : le codage Quoted-Printable EXIGE que les lignes codées ne soit pas longues de plus de 76 caractères. Si des lignes plus longues sont à coder avec le codage Quoted-Printable, des coupures de ligne "douces" doivent être utilisées. Un signe égal comme dernier caractère sur une ligne codée indique une telle coupure de ligne non significative ("douce") dans le texte codé. Donc, si la forme "brute" de la ligne est une seule ligne non codée qui dit : "C'est maintenant le moment où tous doivent venir à l'aide de leur pays". Ceci peut être représenté, en codage Quoted-Printable par :

```
C'est maintenant le moment =
où tous doivent venir =
à l'aide de leur pays.
```

Cela donne un mécanisme par lequel les lignes longues sont codées de façon à être restaurées par l'agent d'utilisateur. La limite de 76 caractères ne compte pas le CRLF de queue, mais compte tous les autres caractères, tous signes "égal" inclus.

Comme le caractère tiret ("-") peut être représenté comme lui-même dans le codage Quoted-Printable, il faut faire attention, quand on encapsule un corps codé en quoted-printable à l'intérieur d'une ou plusieurs entités multi parties, de s'assurer que le délimiteur de frontière n'apparaît nulle part dans le corps codé. (Une bonne stratégie est de choisir une frontière qui comporte une séquence de caractères comme "=_" qui ne peut jamais apparaître dans un corps quoted-printable. Voir la définition des messages multi parties dans la RFC 2046.)

Note : le codage quoted-printable représente une sorte de compromis entre la lisibilité et la fiabilité du transport. Les corps codés avec le codage quoted-printable vont fonctionner de façon fiable sur la plupart des passrelles de messagerie, mais peuvent ne pas fonctionner parfaitement sur quelques passerelles, notamment celles qui impliquent une traduction en EBCDIC. Un niveau de confiance supérieur est offert par le codage de transfert de contenu en base64. Une façon d'obtenir un transport raisonnablement fiable à travers des passerelles EBCDIC est de mettre aussi entre guillemets les caractères US-ASCII !"#$@[\]^` {|}~ conformément à la règle n° 1.

Comme les données quoted-printable sont généralement supposées être en mode ligne, on, peut s'attendre à ce que la représentation des coupures entre les lignes de données quoted-printable puissent être altérée dans le transport, de la même manière que les messages de texte pur ont toujours été altérés dans la messagerie électronique Internet quand elle passe entre des systèmes qui ont des conventions de nouvelle ligne différentes. Si de telles altérations vont probablement constituer une corruption des données, il est probablement plus raisonnable d'utiliser le codage base64 plutôt que le quoted-printable.

Note : plusieurs sortes de sous chaînes ne peuvent pas être générées selon les règles de codage pour le codage de transfert de contenu en quoted-printable, et sont donc formellement illégales si elles apparaissent dans le résultat d'un codeur quoted-printable. Cette note énumère ces cas et suggère des façons de traiter de telles sous chaînes illégales si il s'en rencontre dans des données quoted-printable qui sont à décoder.

(1) Un "=" suivi par deux chiffres hexadécimaux, dont un ou les deux sont des lettres en minuscules dans "abcdef", est formellement illégal. Une mise en œuvre robuste peut choisir de les reconnaître comme les lettres majuscules correspondantes.

(2) Un "=" suivi par un caractère qui n'est ni un chiffre hexadécimal (incluant "abcdef") ni le caractère CR d'une paire CRLF est illégal. Ce cas peut être le résultat d'un texte US-ASCII qui a été inclus dans une partie quoted-printable d'un message sans qu'il ait été lui-même soumis au codage quoted-printable. Une approche raisonnable par une mise en œuvre robuste peut être d'inclure le caractère "=" et le caractère suivant dans les données décodées sans aucune transformation et, si possible, en indiquant à l'utilisateur que le décodage approprié n'a pas été possible à ce moment dans les données.

(3) Un "=" ne peut pas être le dernier ou avant dernier caractère dans un objet codé. Ceci pourrait être traité comme dans le cas (2) ci-dessus.

(4) Les caractères de contrôle autres que TAB, ou CR et LF au titre d'une paire CRLF, ne doivent pas apparaître. Il en est de même pour les octets avec des valeurs décimales supérieures à 126. Si un décodeur en trouve dans des données quoted-printable entrantes, une mise en œuvre robuste pourrait les exclure des données décodées et avertir l'usager que des caractères illégaux ont été décuoverts.

(5) Les lignes codées ne doivent pas être plus longues que 76 caractères, sans compter le CRLF de fin. Si des lignes plus longues sont trouvées dans des données codées entrantes, une mise en œuvre robuste peut néanmoins décoder les lignes, et pourrait rapporter le codage erroné à l'usager.

Avertissement pour les mises en œuvre : si des données binaires sont codées en quoted-printable, il faut faire attention à coder les caractères CR et LF comme "=0D" et "=0A", respectivement. En particulier, une séquence CRLF en données binaires devrait être codée comme "=0D=0A". Autrement, si le CRLF était représenté comme une coupure de ligne dure, il pourrait être décodé incorrectement sur une plateforme qui a des conventions de coupure de ligne différentes.

Pour les formalistes, la syntaxe des données quoted-printable est décrite par la grammaire suivante :

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

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

qp-part := qp-section ; Longueur maximum de 76 caractères

qp-segment := qp-section *(SPACE / TAB) "=" ; Longueur maximum de 76 caractères

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

ptext := hex-octet / safe-char

safe-char := <tout octet d'une valeur décimale de 33 à 60 inclus, et de 62 à 126>
; Les caractères non listés comme "mail-safe" dans la RFC 2049 sont aussi non recommandés.

hex-octet := "=" 2(DIGIT / "A" / "B" / "C" / "D" / "E" / "F")
; Octet doit être utilisé pour les caractères > 127, =, SPACE ou TAB à la fin des lignes, et est recommandé pour tout caractère non listé dans la RFC 2049 comme "mail-safe".

transport-padding := *LWSP-char
; les composeurs NE DOIVENT PAS générer de bourrage de transport de longueur non zéro, mais les receveurs DOIVENT être capables de traiter le bourrage ajouté par les transports de message.

IMPORTANT : L'ajout de LWSP entre les éléments montrés dans ce BNF N'EST PAS permis car le BNF ne spécifie pas un champ d'en-tête structuré.

6.8 Content-Transfer-Encoding en base64

Le Content-Transfer-Encoding en base64 est destiné à représenter des séquences arbitraires d'octets sous une forme qui n'a pas besoin d'être lisible par l'homme. Les algorithmes de codage et décodage sont simples, mais les données codées ne sont que d'environ 33 pour cent plus grandes que les données non codées. Ce codage est virtuellement identique à celui utilisé dans les applications de messagerie à confidentialité améliorée (PEM, *Privacy Enhanced Mail*) comme défini dans la RFC1421.

Un sous ensemble de 65 caractères de l'US-ASCII est utilisé, permettant que 6 bits soient représentés par caractère imprimable. (Le 65ème caractère, "=", est utilisé pour signifier une fonction de traitement spécial.)

Note : ce sous ensemble a l'importante propriété d'être représenté de façon identique dans toutes les versions de ISO 646, incluant US-ASCII, et tous les caractères du sous ensemble sont aussi représentés à l'identique dans toutes les versions de EBCDIC. D'autres codages populaires, comme le codage utilisé par l'utilitaire "uuencode" de Macintosh binhex 4.0 [RFC1741], et le codage base85 spécifié au titre de PostScript niveau 2, ne partagent pas ces propriétés, et ne satisfont donc pas aux exigences de portabilité qu'un codage de transport binaire de messagerie doit satisfaire.

Le processus de codage représente des groupes de 24 bits de bits d'entrée comme des chaînes de résultat de quatre caractères codés. En procédant de gauche à droite, un groupe d'entrée de 24 bits est formé en enchaînant trois groupes d'entrée de 8 bits. Ces 24 bits sont alors traités comme quatre groupes de 6 bits enchaînés, dont chacun est traduit en un seul chiffre dans l'alphabet base64. Lors du codage d'un flux de bits via le codage base64, le flux de bits doit être présumé ordonné avec le bit de poids fort en premier. C'est-à-dire que le premier bit du flux va être le bit de poids fort du premier octet 8bit, et le huitième bit va être le bit de moindre poids du premier octet 8bit, et ainsi de suite.

Chaque groupe de 6 bits est utilisé comme indice dans une matrice de 64 caractères imprimables. Le caractère référencé par l'indice est placé dans la chaîne de sortie. Ces caractères, identifiés dans le Tableau 1, ci-dessous, sont choisis de façon à être universellement représentables, et l'ensemble exclut les caractères avec une signification particulière dans SMTP (par exemple, ".", CR, LF) et pour les délimiteurs de frontière multi parties définis dans la RFC 2046 (par exemple, "-").

Tableau 1 : Alphabet base64

Codage de valeur	Codage de valeur	Codage de valeur	Codage de valeur
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	

Le flux de résultat codé doit être représenté en lignes de pas plus de 76 caractères. Toutes les coupures de ligne ou autres caractères qui ne se trouvent pas dans le Tableau 1 doivent être ignorés par le logiciel de décodage. En données en base64, les caractères autres que ceux du Tableau 1, les coupures de ligne, et autres espaces indiquent probablement une erreur de transmission, au sujet de laquelle un message d'avertissement ou même un rejet de message pourrait être approprié dans certaines circonstances.

Un traitement particulier est effectué si moins de 24 bits sont disponibles à la fin des données à coder. Une pleine quantité de codage est toujours complétée à la fin d'un corps. Quand moins de 24 bits d'entrée sont disponibles dans un groupe d'entrée, des bits à zéro sont ajoutés (sur la droite) pour former un nombre entier de groupes de 6 bits. Le bourrage à la fin des données est effectué en utilisant le caractère "=". Comme toute entrée de base64 est un nombre entier d'octets, seuls les cas suivants peuvent se produire : (1) la quantité finale d'entrée de codage est un multiple entier de 24 bits ; ici, l'unité finale du résultat codé va être un multiple entier de 4 caractères sans bourrage de "=", (2) la quantité finale d'entrée de codage est exactement 8 bits ; ici, l'unité finale du résultat codé va être deux caractères suivis par deux caractères "=" de bourrage, ou (3) la quantité finale d'entrée de codage est exactement 16 bits ; ici, l'unité finale de résultat codé va être trois caractères suivis par un caractère "=" de bourrage.

Parce qu'il est seulement utilisé pour le bourrage à la fin des données, l'occurrence de tout caractère "=" peut être prise comme preuve que la fin des données a été atteinte (sans troncature dans le transit). Aucune assurance comparable n'est cependant possible quand le nombre d'octets transmis était un multiple de trois et qu'aucun caractère "=" n'est présent.

Tous les caractères qui ne sont pas dans l'alphabet base64 sont à ignorer dans les données codées en base64.

Il faut veiller à utiliser les octets appropriés pour les coupures de ligne si le codage base64 est appliqué directement au matériel de texte qui n'a pas été converti en forme canonique. En particulier, les coupures de ligne de texte doivent être converties en séquences CRLF avant le codage en base64. La chose importante à noter est que cela peut être fait directement par le codeur plutôt que dans une étape antérieure de canonisation dans certaines mises en œuvre.

Note : il n'y a pas à se soucier de mettre entre guillemets les potentiels délimiteurs de frontières dans les corps codés en base64 au sein des entités multi parties parce que aucun caractère "tiret" n'est utilisé dans le codage base64.

7. Champ d'en-tête Content-ID

En construisant un agent d'utilisateur de niveau élevé, il peut être désirable de permettre à un corps de faire référence à un autre. Par conséquent, les corps peuvent être étiquetés en utilisant le champ d'en-tête "Content-ID", qui est syntaxiquement identique au champ d'en-tête "Message-ID" :

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

Comme les valeurs de Message-ID, les valeurs de Content-ID doit être générées comme étant uniques au monde.

La valeur de Content-ID peut être utilisée pour identifier de façon univoque les entités MIME dans plusieurs contextes, en particulier pour mettre en antémémoire les données référencées par le mécanisme message/external-body. Bien que l'en-tête Content-ID soit généralement facultatif, son utilisation est EXIGÉE dans les mises en œuvre qui génèrent des données du type de support MIME facultatif "message/external-body". C'est-à-dire que chaque entité message/external-body doit avoir un champ Content-ID pour permettre de mettre de telles données en antémémoire.

Il est aussi à noter que la valeur de Content-ID a une sémantique particulière dans le cas du type de support multipart/alternative. Ceci est expliqué dans la section de la RFC 2046 qui traite de multipart/alternative.

8. Champ d'en-tête Content-Description

La capacité d'associer des informations descriptives à un certain corps est souvent désirable. Par exemple, il peut être utile de marquer un corps "image" comme une "représentation du vaisseau spatial Endeavour". Un tel texte peut être placé dans le champ d'en-tête Content-Description. Ce champ d'en-tête est toujours facultatif.

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

La description est présumée être donnée dans le jeu de caractères US-ASCII, bien que le mécanisme spécifié dans la RFC2047 puisse être utilisé pour des valeurs de Content-Description non US-ASCII.

9. Champs d'en-tête MIME supplémentaires

De futurs documents pourront choisir de définir des champs d'en-tête MIME supplémentaires pour divers objets. Tout nouveau champ d'en-tête qui décrit le contenu d'un message devrait commencer par la chaîne "Content-" pour permettre que de tels champs qui apparaissent dans un en-tête de message soient distingués des champs d'en-tête ordinaires de message RFC0822.

MIME-extension-field := <Tout champ d'en-tête RFC 822 qui commence par la chaîne "Content-">

10. Résumé

En utilisant les champs d'en-tête MIME-Version, Content-Type, et Content-Transfer-Encoding, il est possible d'inclure, d'une façon normalisée, des types arbitraires de données dans des messages conformes à la messagerie de la RFC 822. Aucune restriction imposée par la RFC 821 ou la RFC 822 n'est violée, et il faut veiller à éviter les problèmes causés par des restrictions supplémentaires imposées par les caractéristiques de certains mécanismes de transport de messagerie Internet (voir la RFC 2049).

Le prochain document de cet ensemble, la RFC 2046, spécifie l'ensemble initial de types de supports qui peut être étiqueté et transporté en utilisant ces en-têtes.

11. Considérations sur la sécurité

Les questions de sécurité sont discutées dans le second document de cet ensemble, la RFC 2046.

12. Adresse des auteurs

Pour plus d'informations, il vaut mieux contacter les auteurs de ce document par messagerie Internet :

Ned Freed
Innosoft International, Inc.
1050 East Garvey Avenue South
West Covina, CA 91790
USA
téléphone : +1 818 919 3600
fax : +1 818 919 3614
mél : ned@innosoft.com

Nathaniel S. Borenstein
First Virtual Holdings
25 Washington Avenue
Morristown, NJ 07960
USA
téléphone : +1 201 540 8967
fax : +1 201 993 3032
mél : nsb@nsb.fv.com

MIME est le résultat du travail du groupe de travail sur les extensions à la RFC0822 de l'équipe d'ingénierie de l'Internet. On peut contacter le président de ce groupe, Greg Vaudreuil, à :

Gregory M. Vaudreuil
Octel Network Services
17080 Dallas Parkway
Dallas, TX 75248-1905
USA
mél : Greg.Vaudreuil@Octel.Com

Appendice A -- Récapitulation de la grammaire

Cet appendice contient la grammaire BNF complète pour toute la syntaxe spécifiée dans ce document.

Par elle-même, cependant, cette grammaire est incomplète. Elle se réfère par nom à plusieurs règles de syntaxe qui sont définies par la RFC 822. Plutôt que de reproduire ces définitions ici, et risquer des différences involontaires entre les deux, ce document renvoie simplement le lecteur à la RFC 822 pour les définitions restantes. Chaque fois qu'un terme n'est pas défini, il se réfère à la définition de la RFC 822.

```
attribute := token
             ; La confrontation des attributs est TOUJOURS insensible à la casse.

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

content := "Content-Type" ":" type "/" subtype *(";" parameter)
           ; La confrontation des types et sous types de supports est TOUJOURS insensible à la casse.

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")
; Un octet doit être utilisé pour les caractères > 127, =, les SPACE ou TAB à la fin des lignes, et est recommandé pour tout caractère non listé dans la RFC 2049 comme "mail-safe".

iana-token := <Jeton d'extension défini publiquement. Les jetons de cette forme doivent être enregistrés par l'IANA comme spécifié dans la RFC 2048.>

ietf-token := <Jeton d'extension défini par une RFC sur la voie de la normalisation et enregistré par l'IANA.>

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

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

MIME-extension-field := <Tout champ d'en-tête de la RFC 822 qui commence par la chaîne "Content-">

MIME-message-headers := version de champs entity-headers CRLF
; L'ordre des champs d'en-tête impliqué par cette définition de BNF devrait être ignoré.

MIME-part-headers := entity-headers [champs]
; Tout champ qui ne commence pas par "content-" peut n'avoir pas de signification définie et peut être ignoré.
; L'ordre des champs d'en-tête impliqué par cette définition de BNF devrait être ignoré.

parameter := attribute "=" value

ptext := hex-octet / safe-char

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

qp-part := qp-section
; Longueur maximum de 76 caractères.

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

qp-segment := qp-section *(SPACE / TAB) "="
; Longueur maximum de 76 caractères.

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

safe-char := <tout octet de valeur décimale de 33 à 60 inclus, et de 62 à 126>
; Les caractères non listés comme "mail-safe" dans la RFC 2049 sont aussi non recommandés.

subtype := extension-token / iana-token

token := 1*<tout caractère (US-ASCII) sauf SPACE, CTL, ou tspecials>

transport-padding := *LWSP-char
; Les composeurs NE DOIVENT PAS générer de bourrage de transport de longueur non zéro, mais les receveurs DOIVENT être capables de traiter le bourrage ajouté par les transports de mesages.

tspecials := "(" / ")" / "<" / ">" / "@" / "," / ";" / ":" / "\" / <"> "/" / "[" / "]" / "?" / "="
; Doit être dans une quoted-string, à utiliser dans les valeurs de paramètres.

type := discrete-type / composite-type

value := token / quoted-string

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

x-token := <Les deux caractères "X-" ou "x-" suivis, sans espace intercalée, par un jeton>