

RFC 2045 : Multipurpose Internet Mail Extensions (MIME)

Part One: Format of Internet Message Bodies

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 4 Avril 2008

Date de publication du RFC : Novembre 1996

<http://www.bortzmeyer.org/2045.html>

Autrefois, du temps des dinosaures, les internautes vivaient dans des cavernes et gravaient au silex des courriers en texte seul, sur des écorces d'arbre. En 1992, MIME a été inventé et, depuis, les humains sont civilisés et envoient du spam en HTML, des vidéos débiles et des photos ratées par courrier électronique. Notre RFC 2045¹ est l'actuelle norme MIME.

À l'origine du courrier électronique, la norme (aujourd'hui le RFC 5322) décrivait surtout les métadonnées, les **en-têtes** du message, comme `From:` (l'expéditeur) ou `Date:` (la date d'envoi). Le **corps** du message, lui, était laissé au seul texte brut, dans le seul jeu de caractères US-ASCII.

Pour transmettre du non-texte, par exemple des images de Carla Bruni, on se servait d'encodages non-standard, qui transformaient le fichier binaire en texte. C'était l'époque de uuencode (ou de Binhex pour les macounistes). Même pour le texte seul, dès qu'on dépassait les limites du jeu de caractères ASCII, donc, dès qu'on écrivait autre chose que l'anglais, on ne pouvait plus rester strictement dans les limites du RFC 822 (la norme du courrier de l'époque). Ces limites et les motivations pour MIME sont décrites dans la section 1 du RFC.

MIME a changé cela. Normalisé à l'origine dans le RFC 1341, puis dans le RFC 1521 (et suivants) et finalement dans notre RFC 2045 et ses compagnons suivants, MIME repose sur quatre piliers :

- Un format pour le corps du message, normalisé dans le RFC 2045, format récursif, permettant d'inclure diverses parties dans un message,
- Des encodages pour les parties binaires,

¹Pour voir le RFC de numéro NNN, <http://www.ietf.org/rfc/rfcNNN.txt>, par exemple <http://www.ietf.org/rfc/rfc2045.txt>

- Une notion de **type**, normalisée dans le RFC 2046, permettant d'étiqueter sans ambiguïté les parties du message, notion qui a eu un grand succès et est utilisée par bien d'autres protocoles,
- Un truc permettant de mettre des caractères non-ASCII dans les en-têtes (RFC 2047).

Pour le texte, MIME utilise la notion de "*character set*" (section 2.2). Comme l'explique une note de cette section, le terme est erroné, puisqu'il s'agit en fait, dans MIME, d'un encodage, pas simplement d'un jeu de caractères.

Avec la section 3, commence la description du format MIME lui-même. MIME ajoute quelques nouveaux en-têtes à commencer par `MIME-Version` (section 4) qui indique qu'il s'agit d'un message MIME. La section 5 décrit l'en-tête `Content-type` qui spécifie le type MIME du message ou de la partie du message. Il existe un zoo entier de ces types, conservé dans un registre (<http://www.iana.org/assignments/media-types/index.html>) à l'IANA. Le principe est que le type est composé de deux parties, le **type** proprement dit, qui identifie le genre de données (texte, image, son, ...) et le **sous-type** qui identifie le format. Ainsi, `image/png` est une image au format PNG, `text/troff` est du texte au format troff (RFC 4263). Plus complexes, les types `message` ou `application`. Le premier identifie un message MIME inclus dans un autre message MIME (par exemple lors de l'envoi d'un avis de non-réception (RFC 3461). Le second identifie des données binaires qui ne peuvent être comprises que par une application spécifique, indiquée par le sous-type. La classification MIME n'est pas toujours aussi évidente et je n'ai personnellement jamais compris pourquoi XML est en `application/xml` et pas `text/xml`. Enfin, des paramètres peuvent se trouver dans cet en-tête, par exemple pour indiquer l'encodage d'un texte (`Content-Type:text/plain;charset=utf-8`) ou bien pour indiquer le site où récupérer un document pointé (comme dans le programme d'exemple ci-dessous).

La section 6 introduit un autre en-tête, `Content-Transfer-Encoding` qui indique comment le contenu a été emballé pour passer à travers des serveurs (qui peuvent n'accepter que l'ASCII). Un `Content-Transfer-Encoding:8bit` spécifie que le contenu n'a pas été modifié du tout, il faudra alors que tous les serveurs laissent passer le 8bits. À une époque très lointaine, où les mammoths étaient encore fréquents en France, certains serveurs mettaient le huitième bit à zéro ou autres horreurs ; une longue polémique, au début des années 1990, avait opposé ceux qui voulaient transmettre les textes en français en 8bits (et tant pis pour les quelques serveurs archaïques) et ceux qui préféraient l'encoder dans des systèmes comme `Quoted-Printable` (`Content-Transfer-Encoding:quoted-printable`, décrit en détail dans la section 6.7), qui étaient plutôt pires pour les MUA de l'époque, qui affichaient `caf=E9` au lieu de `café`...

À titre d'exemple, voici un programme Python qui analyse un message MIME. Le programme n'est pas récursif, bien que cela soit souvent utile pour les messages MIME, la définition même du format étant récursive (un message peut être composé de parties qui sont elles-mêmes composées de parties, etc, cf. la note de la section 2.6). Ces messages contiennent une partie de type `message/external-body` qui donne des instructions d'accès à un fichier par FTP :

```
import email
from email.MIMEText import MIMEText
...
# We receive the message on standard input (for instance through
# procmail and its pipe action)
message = email.message_from_file(sys.stdin)
subject = message.get("subject")

# A MIME message can be single-part but we are interested only in multi-parts
if not message.is_multipart():
    raise NotForMe("Not multipart")
...
# The body of the message
parts = message.get_payload()
# Iterate over the parts of the message
for part in parts:
```

```
if part.get_content_type() == "message/external-body":
    params = dict(part.get_params())
    if params["access-type"] == "anon-ftp":
        url = "ftp://%s/%s/%s" % (params["site"],
                                params["directory"],
                                params["name"])
...

```

À l'époque du RFC 822, écrire un programme qui analysait les messages électroniques était trivial, on pouvait travailler directement avec le texte, même un simple `grep` suffisait. Avec MIME, les parties multiples (section 5.1 du RFC 2046), et les encodages variés, l'utilisation de bibliothèques comme `email` en Python (<http://docs.python.org/lib/module-email.html>) devient indispensable. Par exemple, les spammeurs encodent souvent leurs messages en Base64 (section 6.8, désormais dans le RFC 4648), afin de tromper les analyseurs de mots-clés naïfs.

On note que ce RFC, ce qui est relativement rare, contient également d'intéressantes notes (introduites par "NOTE") sur les choix effectués par les concepteurs du format MIME. Leur lecture permet de mieux comprendre ces choix.