RFC 2046 : Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 1 mars 2009

Date de publication du RFC : Novembre 1996

https://www.bortzmeyer.org/2046.html

La norme MIME permettant de distribuer du contenu multimédia par courrier électronique est composée de plusieurs RFC dont le plus connu est le RFC 2045 [1] qui décrit la syntaxe des messages MIME. Mais notre RFC 2046 est tout aussi important car il traite d'un concept central de MIME, le **type** des données. En effet, MIME attribue un type à chaque groupe de données qu'il gère. Cela permet d'identifier du premier coup telle suite de bits comme étant une image PNG ou une chanson codée en Vorbis. Ces types ont eu un tel succès qu'ils ont été repris par bien d'autres systèmes que MIME par exemple dans HTTP pour indiquer le type de la ressource envoyée au navigateur Web (avec Firefox, on peut l'afficher via le menu "*Tools*", option "*Page info*"), ou même par certains systèmes d'exploitation comme BeOS qui, dans les métadonnées du système de fichiers, stocke le type de chaque fichier sous forme d'un type de média. Logiquement, le nom officiel a donc changé et on ne devrait plus dire type MIME mais « type du média ».

Ces types, conçus à l'origine pour être codés dans le message et jamais vus par les utilisateurs ont finalement été diffusés largement et presque tout auteur de contenu sur l'Internet a déjà vu des codes comme `text/html` (une page HTML), `image/jpg` (une image JPEG) ou bien `application/octet-stream` (un fichier binaire au contenu inconnu). Si on veut voir la liste complète, ils sont enregistrés dans un registre à l'IANA <https://www.iana.org/assignments/media-types/index.html>, géré selon les règles du RFC 6838.

Quelle est la structure des types de média ? Chacun est composé de deux parties (section 1 du RFC), le **type de premier niveau** comme `text` ou `audio` qui identifie le genre de fichiers auquel on a affaire (section 2) et le **sous-type** qui identifie le format utilisé. Un type est donc composé d'un

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

type de premier niveau et d'un sous-type, séparés par une barre oblique. Ainsi, `text/plain` désigne des données au format « texte seul » (type `text` et format `plain`, c'est-à-dire sans aucun balisage). `application/msword` est un fichier lisible uniquement par un certain programme ou catégorie de programmes (d'où le type de premier niveau `application`) et au format `msword`, c'est-à-dire le traitement de textes de Microsoft. Notez aussi le type `multipart`, utile lorsqu'on transporte en même temps des données de types différents (chaque partie sera alors marquée avec son propre type de média).

Les types de média sont ensuite indiqués selon une méthode qui dépend du protocole. Dans le courrier électronique, cela se fait avec l'en-tête `Content-Type` par exemple pour un message en texte seul :

```
Content-Type: text/plain;
            charset="utf-8"
```

(Notez un paramètre supplémentaire, la mention de l'encodage, appelé par MIME "*charset*", car un texte brut peut être en ASCII, en Latin-1, en UTF-32, etc. Voir section 4.1.2.)

Dans le monde du Web, voici un exemple affiché par le client HTTP wget :

```
% wget http://upload.wikimedia.org/wikipedia/commons/e/ec/En-us-type.ogg
...
Length: 12635 (12K) [application/ogg]
...
```

Ici, un fichier Ogg (la prononciation de « type » en anglais <`http://upload.wikimedia.org/wikipedia/commons/e/ec/En-us-type.ogg`>) a été chargé et le serveur HTTP avait annoncé le type MIME `application/ogg` (depuis le RFC 5334, cela devrait être `audio/ogg` mais le changement est lent sur l'Internet).

La définition complète des types de premier niveau figure dans la section 2. pour chaque type, on trouve sa description, les paramètres qui ont du sens pour lui (comme `charset` dans l'exemple plus haut), la manière de gérer les sous-types inconnus, etc.

La section 3 décrit les sept types de premier niveau de l'époque (ils sont dix, début 2017). Il y a deux types composites et cinq qui ne le sont pas, détaillés en section 4.

Le plus simple, `text` est... du texte et qu'on puisse visualiser sans logiciel particulier. L'idée est également qu'on peut toujours montrer de telles données à un utilisateur humain, même si le sous-type est inconnu (règle simple mais pas toujours d'application simple, comme l'ont montré les polémiques sur `text/xml`). Cela fait donc de `text` un excellent format stable et lisible par tous (et qui est donc utilisé pour les RFC eux-même). Les formats illisibles sans logiciel particulier ne sont donc pas du `text` au sens MIME. Des exemples de formats `text` ? Évidement `plain` cité plus haut, le texte brut mais aussi :

— `troff` (RFC 4263),
— `enriched` alias `richtext`, qui avait été créé spécialement pour MIME (RFC 1896) mais qui n'a jamais eu beaucoup de succès.
— `csv` (RFC 4180) pour des données au populaire format CSV.

En théorie, les formats de balisage légers modernes comme reST ou comme ceux des Wiki par exemple le langage de MediaWiki pourraient rentrer ici mais, en pratique, aucun sous-type n'a encore été enregistré.

La section sur les encodages, 4.1.2, recommande que le paramètre `charset` soit le PGCD des encodages possibles. Par exemple, si le texte ne comprend que des carractères ASCII, `charset="UTF-8"` est correct (puisque UTF-8 est un sur-ensemble d'ASCII) mais déconseillé. mutt met en œuvre automatiquement cette règle. Si on écrit dans sa configuration, `set send_charset=us-ascii:iso-8859-15:utf-8`, mutt marquera le message comme ASCII s'il ne comprend que des caractères ASCII, sinon ISO-8859-15 s'il ne comprend que du Latin-9 et UTF-8 autrement.

`image` (section 4.2), `audio` (section 4.3) et `video` (section 4.4) sont décrits ensuite.

`application` (section 4.5) est plus difficile à saisir. Ce type de premier niveau est conçu pour le cas où les données sont définies par une application (ou un groupe d'applications puisque, par exemple, de nombreuses applications peuvent lire le format PDF, que le RFC 3778 a enregistré sous le nom de `application/pdf`). Le RFC détaille l'exemple de `application/postscript` pour le langage Postscript (section 4.5.2). Ce type `application` est souvent associé à des problèmes de sécurité car charger un fichier sans précaution peut être dangereux avec certains types de fichier ! Le RFC cite des exemples comme l'opérateur `deletefile` de Postscript mais beaucoup d'autres formats ont des contenus potentiellement « actifs ».

Il y a un certain recouvrement entre les domaines de `text` et de `application`, parfaitement illustré par le vieux débat sur le statut de HTML : est-ce du texte ou pas <https://www.bortzmeyer.org/xhtml-content-type.html> ?

La section 3 couvre aussi le cas des types de premier niveau **composés** comme `multipart` et `message`, détaillés en section 5. Le premier (section 5.1) permet de traiter des messages comportant plusieurs parties, chaque partie ayant son propre type. Par exemple, un message comportant du texte brut et une musique en Vorbis sera de type `multipart/mixed` et comportera deux parties de type `text/plain` et `audio/ogg` (ou, plus précisément, `audio/ogg; codecs="vorbis"`). On peut aussi avoir plusieurs parties équivalentes (par exemple le même texte en texte brut et en HTML) et utiliser alors le type `multipart/alternative`. (Mais attention à bien synchroniser les deux parties <https://www.bortzmeyer.org/mime-newsletter.html>.)

Le second type composé, `message` (section 5.2) sert à encapsuler des messages électroniques dans d'autres messages, par exemple pour les faire suivre ou bien pour un avis de non-remise. Un message au format du RFC 5322 sera ainsi marqué comme `message/rfc822` (le RFC 822 étant l'ancienne version de la norme).

On peut aussi avoir des types de premier niveau et des sous-types **privés** (section 6), attribués par simple décision locale. Pour éviter toute confusion avec les identificateurs enregistrés, ces identificateurs privés doivent commencer par `x-` comme par exemple `application/x-monformat`.

À noter qu'il n'existe pas de types MIME pour les langages de programmation. Quasiment tous sont du texte brut et devraient être servis en `text/plain`. Leur donner un type n'aurait d'intérêt que si on voulait déclencher automatiquement une action sur ces textes... ce qui n'est en général pas une bonne idée, pour des raisons de sécurité. C'est ainsi que seuls les langages "*sandboxables*" comme Javascript (RFC 4329) ont leur type MIME. Résultat, on voit souvent les types privés utilisés, par exemple le fichier `/etc/mime.types` qui, sur beaucoup d'Unix, contient la correspondance entre une extension et un type MIME, inclus, sur une Debian, `text/x-python` pour les fichiers `.py` (code Python). Par contre, pour C, le type est `text/x-csrc` et pas simplement `text/x-c`, sans doute pour être plus explicite. Autre curiosité, je n'y trouve pas de type MIME pour les sources en Lisp...

Ce RFC a remplacé l'ancien RFC 1522, les changements, importants sur la forme mais peu sur le fond, étant décrits dans l'annexe B du RFC 2049.