RFC 7303 : XML Media Types

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 11 juillet 2014

Date de publication du RFC : Juillet 2014

https://www.bortzmeyer.org/7303.html

Voici la nouvelle norme décrivant les types MIME XML, comme `application/xml`. Elle remplace le RFC 3023 [1]. Cette norme décrit aussi l'utilisation de `+xml` comme suffixe pour des formats de données fondés sur XML comme TEI, avec son `application/tei+xml`.

XML est une norme du W3C et n'est donc pas dans un RFC mais dans un document du W3C <http://www.w3.org/XML/>. Lorsque des documents sont envoyés sur l'Internet, ils sont souvent étiquetés avec un type de média, dit aussi, pour des raisons historiques, type MIME. Pour XML, il existe cinq types, `application/xml`, `text/xml`, `application/xml-external-parsed-entity`, `text/xml-external-parsed-entity` et `application/xml-dtd`. Il existe aussi une convention, formalisée dans le RFC 6838, pour les formats bâtis au-dessus de XML, comme Atom (RFC 4287). On écrit les types de média pour ces formats avec un `+xml` à la fin, indiquant ainsi qu'un processeur XML généraliste pourra toujours en faire quelque chose, même s'il ne connait pas ce format spécifique. Ainsi, Atom est `application/atom+xml`. (Ces suffixes sont désormais banals mais, à l'époque du RFC 3023, ils étaient contestés et l'annexe A du RFC 3023 contient une justification détaillée, qui n'a pas été reprise dans ce nouveau RFC 7303.)

Pour normaliser quelque chose à propos de XML, il faut d'abord se pencher sur les problèmes d'encodage de caractères. Le modèle de caractères de XML est forcément Unicode.Les encodages de documents peuvent être très divers, mais la norme Unicode n'en définit que trois, UTF-8, UTF-16 et UTF-32. UTF-8 (RFC 3629) représente les caractères par une suite d'octets de longueur variable. Il n'a qu'une seule sérialisation en octets possible. UTF-16 en a deux, car il représente les caractères (enfin, ceux du PMB) par des seizets. Selon qu'on mette l'octet de poids fort du seizet en premier ou en dernier, on parle d'UTF-16 gros-boutien (`utf-16be` dans une déclaration XML) ou d'UTF-16 petit-boutien (`utf-16le`). Quant à UTF-32, qui est le plus simple des encodages, car le plus uniforme (tout caractère Unicode est

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc3023.txt`

représenté par quatre octets, codant son point de code), il a quatre sérialisations possibles en théorie mais deux seulement sont définies, `UTF-32BE`, gros-boutien, et `UTF32-LE`, petit-boutien. UTF-32 est malheureusement peu déployé (qu'on ne me dise pas que c'est parce qu'il prend plus de place : quatre octets par caractère, à l'époque où on s'échange des vidéos HD, ce n'est rien). Ce RFC déconseille désormais son usage. (Une bonne discussion des problèmes avec UTF-32 est dans ce rapport de bogue chez Mozilla `<https://bugzilla.mozilla.org/show_bug.cgi?id=604317>`.)

La section 3 du RFC décrit toutes les recommandations actuelles pour l'encodage des documents XML envoyés sur l'Internet. UTF-8 est recommandé (et sans BOM : il faut le retirer si, par exemple, on convertit de l'UTF-16 en UTF-8). Les producteurs/envoyeurs de XML devraient mettre un paramètre `charset` (pour les protocoles qui ont ce paramètre) et utiliser la déclaration XML (le truc qui commence par `<?xml version="1.0" encoding=...`) lorsqu'ils utilisent autre chose qu'UTF-8.

Cela suppose qu'ils sachent exactement quel est l'encodage. Si on envoie de l'XML et qu'on ne peut pas déterminer son encodage, il vaut mieux ne rien dire que de raconter n'importe quoi. Le RFC note que c'est particulièrement important pour les serveurs Web : avoir un paramètre global pour tout le site qui indique un encodage est dangereux, à moins qu'on puisse être absolument sûr que tous les documents du site auront cet encodage. Le RFC recommande que cet étiquetage soit configurable par l'utilisateur, pour qu'il puisse l'adapter à ses fichiers (`AddDefaultCharset` et `AddCharset` dans Apache.)

Quant aux consommateurs de documents XML, ils doivent considérer le BOM comme faisant autorité, et utiliser le paramètre `charset` si le BOM est absent. Les consommateurs qui comprennent le XML peuvent, s'il n'y a ni BOM, ni `charset`, utiliser les techniques purement XML, exposées dans la norme de ce format (en gros, utiliser la déclaration par exemple `<?xml version="1.0" encoding="ISO 8859-1"?>`. Il y a plein d'exemples en section 8 du RFC. Par exemple, ce cas-ci :

```
Content-Type: application/xml; charset=utf-8
...
<?xml version="1.0" encoding="utf-8"?>
```

est trivial : tout le monde est d'accord, le document est en UTF-8 et c'est explicite. En revanche, ce cas :

```
Content-Type: application/xml; charset=iso-8859-1
...
<?xml version="1.0" encoding="utf-8"?>
```

est pathologique : le type MIME dans l'en-tête HTTP et la déclaration XML se contredisent. Ici, un logiciel respectueux des normes doit traiter le document comme de l'ISO 8859-1, le type MIME ayant priorité (il n'y a pas de BOM, qui aurait une priorité supérieure). Mais, évidemment, l'envoyeur n'aurait pas dû générer ces informations incohérentes. Il vaut mieux ne pas envoyer de `charset` que d'envoyer un incorrect (voir aussi annexe C.2). Ainsi, dans ce troisième exemple :

```
Content-Type: application/xml
...
<?xml version="1.0"?>
```

il n'y a rien d'explicite : pas de BOM, pas de `charset` dans le type MIME et pas d'encodage indiqué dans la déclaration XML. Dans ce cas, puisque MIME ne dit rien, on applique la règle XML : l'encodage par défaut en XML est UTF-8.

À noter que le transport le plus utilisé pour XML, HTTP, avait un encodage de caractères par défaut, ISO 8859-1 mais ce n'est plus le cas depuis le RFC 7230. Voir aussi le RFC 6657 pour l'encodage par défaut des types `text/*`.

Assez fait d'encodage, place aux types MIME, sujet principal de ce RFC. La section 4 donne leur liste :

— Les documents XML eux-mêmes auront le type `application/xml` ou `text/xml`. Le RFC ne tranche pas entre les deux possibilités, alors que ce choix a fait se déplacer beaucoup d'électrons, lors des innombrables discussions sur le « meilleur » type. (Le prédécesseur, le RFC 3023 recommandait `text/xml`.)
— Les DTD auront le type `application/xml-dtd`.
— Les « entités externes analysées » ("*external parsed entities*"), sont de type `application/xml-external-parsed-` ou `text/xml-external-parsed-entity`. Si elles sont par ailleurs des documents XML bien formés (ce qui n'est pas toujours le cas), elles peuvent avoir les types `application/xml` ou `text/xml`.
— Les paramètres externes ("*external parameter entities*") ont le même type que les DTD, `application/xml-dtd`.

Ils sont tous mis dans le registre IANA <`https://www.iana.org/assignments/media-types/media-types.xhtml`> (cf. section 9 du RFC).

Il y a aussi le suffixe `+xml` (section 4.2), utilisant la notion de suffixe du RFC 6839. Il a vocation à être utilisé pour tous les formats XML. Si les concepteurs de ce format ne veulent pas d'un traitement générique XML (un cas probablement rare), ils doivent choisir un type MIME sans ce suffixe.

À noter que, si on envoie du XML en HTTP, le système de négociation de contenu de HTTP ne prévoit pas de mécanisme pour dire « j'accepte le XML, quel que soit le format ». Pas question de dire `Accept: text/*+xml`, donc.

La section 5 décrit ensuite le cas des identificateurs de fragment (le texte après le # dans un URI, pour désigner une partie d'un document). Ils sont décrits dans la section 3.5 du RFC 3986. Pour le cas particulier de XML, on se sert de la syntaxe XPointer pour écrire ces identificateurs (pas exemple, `http://www.example.org/data.html#xpointer(/foo/bar)`).

Le document XML envoyé peut utiliser l'attribut `xml:base` (section 6) pour spécifier un URI de référence, utilisé pour construire des URI absolus à partir des relatifs qu'on peut trouver dans le document XML (par exemple, lors de l'inclusion d'un autre document XML).

Et les versions de XML (il existe actuellement deux versions normalisées, 1.0 et 1.1) ? Voyez la section 7, qui précise que les types MIME sont exactement les mêmes pour toutes les versions : un analyseur XML doit utiliser les techniques XML (le champ `version` dans la déclaration) pour identifier les versions, pas les types MIME.

Enfin, si vous vous intéressez à la sécurité, voyez la section 10 qui résume rapidement les grandes questions de sécurité de XML. D'abord, les documents XML étant souvent modulaires (un document déclenche le chargement de paramètres ou d'entités XML externes, ou encore de feuilles de style), la sécurité doit prendre en compte **tous** les documents extérieurs et leur technique de chargement. Si un document XML est chargé en HTTPS mais qu'il contient une référence HTTP à une feuille de style CSS, la sécurité de HTTPS ne protégera pas cette dernière, alors qu'elle pourrait, par exemple, faire disparaître

certains des éléments XML (`display: none;`...) Ainsi, beaucoup de documents XML se réfèrent à des entités XML qui sont sur les serveurs du W3C, sans se préoccuper d'analyser la sécurité desdits serveurs. Si la définition de l'entité `mdash` est remplacée par le texte de Winnie l'ourson, de drôles de résultats peuvent survenir.

Il y a aussi des attaques par déni de service : un document XML qui contient une référence à une entité qui contient à son tour de nombreuses autres références, qui contiennent chacune de nombreuses autres références, et ainsi de suite, jusqu'au dépassement de pile. (La forme la plus triviale de l'attaque, où l'entité se référence elle-même, est normalement empêchée par la prohibition de ces auto-références.)

Également d'un grand intérêt pratique, l'annexe C, sur les questions opérationnelles. La situation actuelle, notamment pour trouver l'encodage des caractères d'un document, n'est pas idéale. La nouvelle règle sur les BOM (priorité sur le paramètre `charset`) casse, en théorie, la compatibilité mais, en pratique, n'aggrave pas la situation et devrait l'améliorer à terme. Si on veut se simplifier la vie lors de l'envoi de XML, on le met en UTF-8 : c'est l'encodage recommandé par défaut, et il se passe de BOM. Si on n'utilise pas UTF-8, alors, il faut mettre un BOM.

Et le consommateur de XML ? Il lui suffit de regarder le BOM sinon le `charset` sinon la déclaration XML. En suivant ces règles simples, tout le monde devrait être heureux.

Depuis le RFC 3023, la précédente norme, que s'est-il passé ? L'annexe D résume les nombreux et sérieux changements. Les sous-types de `text` comme `text/xml` ont vu leur définition alignée avec celle du sous-type d'`application`. `text/xml` est donc désormais un simple synonyme de `application/xml`, ce qui reflète la réalité des mises en œuvre logicielles de XML. Le BOM est désormais prioritaire sur le `charset`. D'autre part, XPointer est désormais intégré. Et UTF-32 est maintenant officiellement déconseillé, au nom de l'interopérabilité.