Groupe de travail Réseau
Request for Comments : 2781
Catégorie : Information
Traduction Claude Brière de L'Isle

P. Hoffman, Internet Mail Consortium
F. Yergeau, Alis Technologies

février 2000

UTF-16, format de codage de la norme ISO 10646

1. Introduction

Le présent document décrit le codage UTF-16 de la norme Unicode/ISO-10646, traite les problèmes de mise en série de l'UTF-16 comme flux d'octets pour transmission sur l'Internet, discute de la désignation des jeux de caractères MIME comme décrits dans la [RFC2278], et contient l'enregistrement de trois valeurs de paramètre de jeu de caractères MIME : UTF-16BE (gros boutien), UTF-16LE (petit boutien), et UTF-16.

1.1 Fondements et motivation

La norme Unicode [UNICODE] et la norme ISO/CEI 10646 [ISO-10646] définissent conjointement un jeu de caractères codés (CCS, *coded character set*), qu'on appellera ici Unicode, qui englobe la plupart des systèmes d'écriture du monde [RFC2130]. UTF-16, objet de la présente spécification, est une des façons standard de coder les données de caractères Unicode ; il a pour caractéristique de coder tous les caractères actuellement définis (dans le tableau multilingue de base 0, ou BMP) dans exactement deux octets et d'être capable de coder tous les autres caractères qui auraient une probabilité d'être définis (les 16 plans suivants) sur exactement quatre octets.

La norme Unicode définit de plus des propriétés de caractères supplémentaires et d'autres détails d'application de grand intérêt pour les mises en œuvre. Jusqu'à présent, les changements d'Unicode et les amendements à la norme ISO/CEI 10646 se sont suivis l'un l'autre, afin que les répertoires de caractères et les allocations de codets restent synchronisés. Les comités de normalisation pertinents se sont engagés à conserver ce très utile synchronisme, ainsi qu'à ne pas allouer de caractères en dehors des 17 plans accessibles à l'UTF-16.

La politique de l'IETF sur les jeux de caractères et les langages [RFC2277] dit que les protocoles de l'IETF DOIVENT être capables d'utiliser le schéma de codage de caractères UTF-8 [RFC2279]. Certains produits et réseaux standard spécifient déjà UTF-16, en faisant un codage important pour l'Internet. Le présent document n'est pas une mise à jour de la [RFC2277], mais seulement une description du codage UTF-16.

1.2 Terminologie

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" en majuscules dans ce document sont à interpréter comme décrit dans le BCP 14, [RFC2119].

Dans le présent document, les valeurs de caractères sont données en notation hexadécimale. Par exemple, "0x013C" est le caractère dont la valeur est le caractère auquel est allouée la valeur d'entier 316 (en décimal) dans le CCS.

2. Définition de UTF-16

UTF-16 est décrit dans la norme Unicode, version 3.0 [UNICODE]. La référence définitive est l'Annexe Q de la norme ISO/CEI 10646-1 [ISO-10646]. Le reste de cette section reprend la définition de ses propres termes.

Dans la norme ISO 10646, un numéro est alloué à chaque caractère, ce que Unicode appelle la valeur scalaire Unicode. Ce numéro est le même que la valeur UCS-4 du caractère, et le présent document va s'y référer pour faire court comme à la "valeur du caractère". Dans le codage UTF-16, les caractères sont représenté en utilisant un ou deux entiers non signés de 16 bits, selon la valeur du caractère. La mise en série de ces entiers pour transmission comme flux d'octets est exposée à la Section 3.

Les règles de codage des caractères en UTF-16 sont :

- Les caractères qui ont des valeurs inférieures à 0x10000 sont représentées sur un seul entier de 16 bits avec une valeur égale à celle du numéro du caractère.

- Les caractères qui ont des valeurs entre 0x10000 et 0x10FFFF ont représentées par un entier de 16 bits avec une valeur entre 0xD800 et 0xDBFF (dans ce qu'on appelle la demi zone haute ou la zone de remplacement haute) suivie par un entier de 16 bits d'une valeur entre 0xDC00 et 0xDFFF (dans ce qu'on appelle la demi zone basse ou zone de remplacement basse).

- Les caractères qui ont des valeurs supérieures à 0x10FFFF ne peuvent pas être codés en UTF-16.

Note : Les valeurs entre 0xD800 et 0xDFFF sont spécifiquement réservée à l'usage de UTF-16, et aucun caractère ne leur est alloué.

2.1 Codage UTF-16

Le codage d'un seul caractère d'une valeur de caractère de l'ISO 10646 en UTF-16 se fait comme suit. Soit U le numéro de caractère, non supérieur à 0x10FFFF.

1) Si U < 0x10000, coder U comme un entier de 16 bits non signé et terminer.

2) Soit U' = U - 0x10000. Parce que U est inférieur ou égal à 0x10FFFF, U' doit être inférieur ou égal à 0xFFFFF. C'est à dire que U' peut être représenté sur 20 bits.

3) Initialiser deux entiers de 16 bits non signés, W1 et W2, respectivement à 0xD800 et 0xDC00. Ces entiers ont chacun 10 bits libres pour coder la valeur du caractère, pour un total de 20 bits.

4) Allouer les 10 bits de poids fort des 20 bits de U' aux 10 bits de moindre poids de W1 et les 10 bits de moindre poids de U' aux 10 bits de moindre poids de W2. Terminer.

Graphiquement, les étapes 2 à 4 sont :
U' = yyyyyyyyyyxxxxxxxxxx
W1 = 110110yyyyyyyyyy
W2 = 110111xxxxxxxxxx

2.2 Décodage UTF-16

Décoder un seul caractère de UTF-16 en une valeur de caractère ISO 10646 se fait comme suit. Soit W1 le prochain entier de 16 bits dans la séquence des entiers représentant le texte. Soit W2 le prochain (éventuel) entier W1 suivant.

1) Si W1 < 0xD800 ou si W1 > 0xDFFF, la valeur de caractère U est la valeur de W1. Terminer.

2) Déterminer si W1 est entre 0xD800 et 0xDBFF. Sinon, la séquence est erronée et aucun caractère valide ne peut être obtenu en utilisant W1. Terminer.

3) Si il n'y a pas de W2 (c'est-à-dire, si la séquence se termine par W1) ou si W2 n'est pas entre 0xDC00 et 0xDFFF, la séquence est erronée. Terminer.

4) Construire un entier non signé de 20 bits U', en prenant les 10 bits de moindre poids de W1 comme les 10 bits de poids fort et les 10 bits de moindre poids de W2 comme les 10 bits de moindre poids.

5) Ajouter 0x10000 à U' pour obtenir la valeur du caractère U. Terminer.

Noter que les étapes 2 et 3 indiquent des erreurs. La récupération d'erreur n'est pas spécifiée dans ce document. Lorsque on

termine sur une erreur dans les étapes 2 et 3, il peut être sage de régler U à la valeur de W1 pour aider l'appelant à diagnostiquer l'erreur et ne pas perdre les informations. Noter aussi qu'un algorithme de décodage de chaîne, par opposition au décodage d'un seul caractère décrit ci-dessus, n'a pas besoin de se terminer sur la détection d'une erreur, si un rapport d'erreur approprié et une récupération sont fournis.

3. Étiquetage de texte UTF-16

L'Appendice A de la présente spécification contient l'enregistrement de trois jeux de caractères MIME : "UTF-16BE", "UTF-16LE", et "UTF-16". Les jeux de caractères MIME représentent la combinaison d'un CCS (jeu de caractères codé) et d'un schéma de codage de caractères (CES, *character encoding scheme*). Ici, le CCS est Unicode/ISO 10646 et le CES est le même dans les trois cas, sauf l'ordre de mise en série des octets dans chaque caractère, et la détermination externe de quelle mise en série est utilisée.

Cette section décrit laquelle des trois étiquettes appliquer à un flux de texte. La Section 4 décrit comment interpréter les étiquettes sur un flux de texte.

3.1 Définition de gros boutien et petit boutien

Historiquement, les ordinateurs ont traité des entités de deux octets comme des entiers de 16 bits d'une des deux façons suivantes : les matériels dits "gros boutiens" traitent les entités de deux octets avec l'octet de poids fort en premier, c'est-à-dire à la plus faible adresse dans la mémoire ; lorsque il est écrit sur le disque ou dans une interface réseau (mise en série), l'octet de poids fort apparaît donc en premier dans le flux de données. D'un autre côté, les matériels "petits boutiens" traitent les entités de deux octets avec l'octet de moindre poids en premier. Les matériels des deux sortes sont courants aujourd'hui.

Par exemple, l'entier non signé de 16 bits qui représente le nombre décimal 258 est 0x0102. La mise en série grosse boutiennne de ce nombre est l'octet 0x01 suivi par l'octet 0x02. La mise en série petite boutienne de ce nombre est l'octet 0x02 suivi par l'octet 0x01. Le fragment de code C suivant montre une façon d'écrire les quantités de 16 bits dans un fichier gros boutien, sans considération de l'ordre natif des octets du matériel.

```
void write_be(unsigned short u, FILE f)          /* on suppose que short est 16 bits */
{
  putc(u >> 8,   f);                             /* octet de poids fort en sortie */
  putc(u & 0xFF, f);                             /* puis de moindre poids */
}
```

Les termes "ordre des octets du réseau" a été utilisé dans de nombreuses RFC pour indique la mise en série grosse boutienne, bien que ces termes n'aient pas encore été formellement définis dans un document sur la voie de la normalisation. Bien que la norme ISO 10646 préfère la mise en série grosse boutienne (paragraphe 6.3 de [ISO-10646]), l'ordre petit boutien est parfois utilisé dans l'Internet.

3.2 Marque d'ordre des octets

La norme Unicode et ISO 10646 définissent le caractère "ESPACE INSÉCABLE DE LARGEUR ZÉRO" *(ZERO WIDTH NON-BREAKING SPACE)* (0xFEFF), qui est aussi appelé de façon informelle la "MARQUE D'ORDRE DES OCTETS" (BOM, *BYTE ORDER MARK*). Ce dernier nom donne une indication d'une seconde utilisation possible du caractère, en plus de son utilisation normale comme une authentique "ESPACE INSÉCABLE DE LARGEUR ZÉRO" au sein d'un texte. Cet usage, suggéré par le paragraphe 2.4 de Unicode et l'Annexe F de ISO 10646 (pour information) est d'ajouter un caractère 0xFEFF à un flux de caractères Unicode comme une "signature" ; le receveur d'un tel flux mis en série peut alors utiliser le caractère initial à la fois comme l'indication que le flux consiste en caractères Unicode et comme moyen de reconnaître l'ordre de mise en série. Dans UTF-16 mis en série précédé d'une telle signature, l'ordre est gros boutien si les deux premiers octets sont 0xFE suivis par 0xFF ; si c'est 0xFF suivi par 0xFE, l'ordre est petit boutien. Noter que 0xFFFE n'est pas un caractère Unicode, précisément pour préserver l'utilité de 0xFEFF comme marque de l'ordre des octets.

Il est important de comprendre que le caractère 0xFEFF apparaissant à toute position autre que le début d'un flux DOIT être interprété avec la sémantique d'espace insécable de largeur zéro, et NE DOIT PAS être interprété comme marque d'ordre des octets. L'inverse de cette assertion n'est pas toujours vrai : le caractère 0xFEFF en première position d'un flux PEUT être interprété comme une espace insécable de largeur zéro, et n'est pas toujours une marque d'ordre des octets. Par exemple, si un processus partage une chaîne UTF-16 en plusieurs parties, une d'elles peut commencer par 0xFEFF parce qu'il y avait une espace insécable de largeur zéro au début de cette sous chaîne.

La norme Unicode suggère de plus qu'un caractère initial 0xFEFF peut être supprimé avant de traiter le texte, la raison en étant qu'un tel caractère en position initiale pourrait être un artifice de codage (une signature de codage) et non une authentique "ESPACE INSÉCABLE DE LARGEUR ZÉRO" voulue. Noter qu'une telle suppression peut affecter un processus externe à une couche différente (comme une signature numérique ou un compte des caractères) qui s'appuie sur la présence de tous les caractères du flux.

En particulier, dans un texte source UTF-16, il est probable, mais pas certain, qu'un 0xFEFF initial soit une signature. Lorsque on enchaîne deux chaînes, il est important de supprimer ces signatures, parce que autrement, la chaîne résultante peut contenir une "ESPACE INSÉCABLE DE LARGEUR ZÉRO" inattendue au point de connexion. Aussi, certaines spécifications rendent obligatoire un caractère initial 0xFEFF dans les objets marqués comme UTF-16 et spécifient que cette signature ne fait pas partie de l'objet.

3.3 Choisir une étiquette pour les texte UTF-16

Toute application d'étiquetage qui utilise le codage de caractères UTF-16 et étiquette le texte de façon explicite, et connaît l'ordre de mise en série des caractères dans le texte, DEVRAIT étiqueter le texte soit comme "UTF-16BE" soit comme "UTF-16LE", selon ce qui est approprié sur la base de la caractéristique de bout du texte. Cela permet aux applications qui traitent le texte, mais ne peuvent pas regarder à l'intérieur du texte, de connaître avec certitude la mise en série.

Le texte dans le jeu de caractères "UTF-16BE" DOIT être mis en série avec les octets qui constituent une seule valeur UTF-16 de 16 bits en ordre gros boutien. Les systèmes qui étiquettent le texte UTF-16BE NE DOIVENT PAS ajouter un BOM au texte.

Le texte dans le jeu de caractères "UTF-16LE" DOIT être mis en série avec les octets qui constituent une seule valeur UTF-16 de 16 octets dans l'ordre petit boutien. Les systèmes qui étiquettent le texte UTF-16LE NE DOIVENT PAS ajouter de BOM au texte.

Toute application d'étiquetage qui utilise le codage de caractères UTF-16 et met une étiquette explicite de jeu de caractères sur le texte, et ne sait pas l'ordre de mise en série des caractères du texte, DOIT étiqueter le texte comme "UTF-16", et DEVRAIT s'assurer que le texte commence par 0xFEFF.

Une exception à la règle "DEVRAIT" d'utiliser "UTF-16BE" ou "UTF-16LE" va se produire avec les formats de document qui rendent obligatoire un BOM dans du texte UTF-16, exigeant par là l'utilisation de la seule étiquette "UTF-16".

4. Interprétation des étiquettes de texte

Lorsque un programme voit un texte étiqueté comme "UTF-16BE", "UTF-16LE", ou "UTF-16", il peut faire des hypothèses, sur la base des règles d'étiquetage données dans la section précédente. Ces hypothèses permettent au programme de traiter alors le texte.

4.1 Interprétation de texte étiqueté UTF-16BE

Le texte étiqueté "UTF-16BE" peut toujours être interprété comme étant gros boutien. La détection d'un BOM initial n'affecte pas la dé sérialisation d'un texte étiqueté UTF-16BE. Trouver un 0xFF suivi de 0xFE est une erreur car il n'y a pas de caractère Unicode 0xFFFE.

4.2 Interprétation de texte étiqueté UTF-16LE

Le texte étiqueté "UTF-16LE" peut toujours être interprété comme étant petit boutien. La détection d'un BOM initial n'affecte pas la dé sérialisation d'un texte étiqueté UTF-16LE. Trouver un 0xFE suivi de 0xFF est une erreur car il n'y a pas de caractère Unicode 0xFFFE, ce qui serait l'interprétation de ces octets dans l'ordre petit boutien.

4.3 Interprétation de texte étiqueté UTF-16

Le texte étiqueté avec le jeu de caractères "UTF-16" pourrait être mis en série soit en ordre gros boutien, soit en ordre petit boutien. Si les deux premiers octets du texte sont 0xFE suivi de 0xFF, le texte peut alors être interprété comme étant gros boutien. Si les deux premiers octets du texte sont 0xFF suivi de 0xFE, le texte peut alors être interprété comme étant petit boutien. Si les deux premiers octets du texte ne sont pas 0xFE suivi de 0xFF, et ne sont pas 0xFF suivi de 0xFE, le texte

DEVRAIT alors être interprété comme étant gros boutien.

Toutes les applications qui traitent du texte qui a l'étiquette de jeu de caractère "UTF-16" DOIVENT être capables de lire au moins les deux premiers octets du texte et être capables de traiter ces octets afin de déterminer l'ordre de mise en série du texte. Les applications qui traitent du texte qui a l'étiquette "UTF-16" NE DOIVENT PAS supposer la mise en série avant d'avoir d'abord vérifié les deux premiers octets pour voir si c'est un BOM gros boutien, un BOM petit boutien, ou pas un BOM. Toutes les applications qui traitent du texte qui a l'étiquette de jeu de caractères "UTF-16" DOIVENT être capables d'interpréter le texte gros boutien aussi bien que petit boutien.

5. Exemples

À titre d'exemple, supposons qu'il y a un caractère hiéroglyphique représentant le dieu égyptien Râ avec une valeur de caractère de 0x12345 (ce caractère n'existe pas présentement dans Unicode).

Les exemples donnés ici évaluent tous la phrase : *=Ra

où le "*" représente le hiéroglyphe Râ (0x12345).

Texte étiqueté UTF-16BE, sans BOM : D8 08 DF 45 00 3D 00 52 00 61

Texte étiqueté UTF-16LE, sans BOM : 08 D8 45 DF 3D 00 52 00 61 00

Texte gros boutien étiqueté UTF-16, avec un BOM : FE FF D8 08 DF 45 00 3D 00 52 00 61

Texte petit boutien étiqueté UTF-16, avec un BOM : FF FE 08 D8 45 DF 3D 00 52 00 61 00

6. Versions des normes

La norme ISO/CEI 10646 est mise à jour de temps en temps par la publication d'amendements ; de même, différentes versions de la norme Unicode existent : 1.0, 1.1, 2.0, 2.1, et 3.0 au moment de cette rédaction. Chaque nouvelle version remplace la précédente, mais les mises en œuvre, et plus significatif, les données, ne sont pas mises à jour instantanément.

En général, les changements reviennent à ajouter de nouveaux caractères, ce qui ne pose pas de problèmes particuliers pour les vieilles données. L'amendement 5 à la norme ISO/CEI 10646 a cependant déplacé et étendu le bloc coréen Hangul, rendant ainsi invalides toutes les données antérieures contenant des caractères Hangul dans la nouvelle version. Unicode 2.0 a la même différence avec Unicode 1.1. La justification officielle pour permettre un tel changement incompatible était qu'il n'existait aucune mise en œuvre significative de données contenant du Hangul, déclaration qui est probablement vrai mais reste impossible à prouver. L'incident a été brocardé sous le nom de "bazar coréen", et les comités compétents ont promis de ne jamais, au grand jamais, faire un tel changement incompatible.

Les nouvelles versions, et en particulier tout changement incompatible, a des conséquences sur les étiquettes de codage de caractères MIME, dont on discute dans l'Appendice A.

7. Considérations relatives à l'IANA

IANA va enregistrer les jeux de caractères qu'on trouvera dans les Appendices A.1, A.2, et A.3 conformément à la RFC2278, en utilisant les gabarits d'enregistrement de ces appendices.

8. Considérations sur la sécurité

UTF-16 se fonde sur le jeu de caractères de la norme ISO 10646, à laquelle sont faites de fréquentes additions, comme décrit à la Section 6 et l'Appendice A du présent document. Les processeurs doivent être capables de traiter les caractères qui ne sont pas définis au moment de la création du processeur d'une façon telle qu'elle ne permette pas à un attaquant de faire des dommages à un receveur en incluant des caractères inconnus.

Les processeurs qui traitent tout type de texte, y compris du texte codé en UTF-16, doivent être vigilants lors de la

vérification des caractères de contrôle qui peuvent reprogrammer un terminal d'affichage ou un clavier. De même, les processeurs qui interprètent des entités de texte (comme à la recherche de code de programmation incorporé) doivent être attentif s à ne pas exécuter le code sans avoir d'abord averti le receveur.

Le texte en UTF-16 peut contenir des caractères spéciaux, comme le CARACTÈRE DE REMPLACEMENT D'OBJET *(OBJECT REPLACEMENT CHARACTER)* (0xFFFC), qui peut causer un traitement externe, selon l'interprétation du programme de traitement et la disponibilité d'un flux de données externes qui serait exécuté. Ce traitement externe peut avoir des effets collatéraux qui permettent à l'envoyeur d'un message d'attaquer le système receveur.

Les mises en œuvre de UTF-16 doivent considérer les aspects de sécurité qui découlent du traitement de séquences UTF-16 illégales (c'est-à-dire, des séquences impliquant des paires de substituts qui ont des valeurs illégales ou des substituts non appariés). Il est concevable que dans certaines circonstances un attaquant sera capable d'exploiter un analyseur UTF-16 imprudent en lui envoyant une séquence d'octets qui n'est pas permise par la syntaxe UTF-16, lui causant un comportement anormal.

9. Références

[ISO-10646] Norme internationale ISO/CEI 10646-1:1993. "Technologies de l'information – Jeu de caractère universel codé sur plusieurs octets (UCS) -- Partie 1 : Architecture et plan multilingue de base". 22 amendements et deux corrigendums techniques ont été publiés jusqu'à présent. UTF-16 est décrit dans l'Annexe Q, publiée comme Amendement 1. De nombreux autres amendements sont actuellement à divers stades de normalisation. Une seconde édition est en préparation, à publier probablement en 2000 ; dans cette nouvelle édition, UTF-16 sera probablement décrit dans l'Annexe C.

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997.

[RFC2130] C. Weider, C. Preston, K. Simonsen, H. Alvestrand, R. Atkinson, M. Crispin, P. Svanberg, "Rapport de l'atelier Jeux de caractères de l'IAB tenu du 29 février au 1er mars 1996", avril 1997. *(Information)*

[RFC2277] H. Alvestrand, "Politique de l'IETF en matière de jeux de caractères et de langages", BCP 18, janvier 1998.

[RFC2278] N. Freed, J. Postel, "Procédures d'enregistrement des jeux de caractères par l'IANA", janvier 1998. (*Obsolète, voir* RFC2978) *(Information)*

[RFC2279] F. Yergeau, "UTF-8, un format de transformation de la norme ISO 10646", janvier 1998. (*Obsolète, voir* RFC3629) *(D.S.)*

[RFC2616] R. Fielding et autres, "Protocole de transfert hypertexte -- HTTP/1.1", juin 1999. *(D.S., MàJ par* 2817, 6585)

[UNICODE] The Unicode Consortium, "The Unicode Standard -- Version 3.0", ISBN 0-201-61633-5. Décrit à <http://www.unicode.org/unicode/standard/versions/Unicode3.0.html>.

10. Remerciements

Deborah Goldsmith a écrit une grande partie de la formulation initiale de la présente spécification. Martin Duerst a proposé de nombreux changements significatifs. D'autres contributeurs significatifs sont Mati Allouche, Walt Daniels, Mark Davis, Ned Freed, Asmus Freytag, Lloyd Honomichl, Dan Kegel, Murata Makoto, Larry Masinter, Markus Scherer, Keld Simonsen, et Ken Whistler.

Une partie du texte de cette spécification a été copié de la [RFC2279], et ce document a été travaillé par de nombreuses personnes. Prière de se reporter à la Section "Remerciements" de ce document pour voir plus de contributeurs indirects au présent document.

A. Enregistrements de jeu de caractères

Le présent mémoire est destiné à servir de base pour l'enregistrement de trois jeux de caractères MIME [RFC2278]. Les jeux de caractères proposés sont "UTF-16BE", "UTF-16LE", et "UTF-16". Ces objets d'étiquette de chaînes contiennent du

texte consistant en caractères tirés du répertoire de la norme ISO/CEI 10646 incluant tous les amendements au moins jusqu'à l'amendement 5 (bloc coréen) codés en une séquence d'octets utilisant le codage et les schémas de mise en série mentionnés plus haut.

Noter que "UTF-16BE", "UTF-16LE", et "UTF-16" NE CONVIENNENT PAS pour une utilisation dans les types de supports sous le type de niveau supérieur "text", parce qu'ils ne codent pas les terminaisons de ligne de la façon requise pour les types de support MIME "text". Une exception à ceci est HTTP, qui utilise un mécanisme semblable à MIME, mais est exempt des restrictions sur le type texte de niveau supérieur (voir le paragraphe 19.4.2 de HTTP 1.1 [RFC2616]).

On notera que les étiquettes décrites ici ne contiennent pas d'identification de version, se référant de façon générique à la norme ISO/CEI 10646. Ceci est intentionnel, en voici la raison :

Un jeu de caractères MIME est conçu pour donner juste les informations nécessaires pour interpréter une séquence d'octets reçus sur le réseau en une séquence de caractères, et rien de plus (voir MIME, au paragraphe 2.2 de la [RFC2045]). Tant qu'une norme de jeu de caractères ne change pas de façon incompatible, les numéros de version ne servent à rien, parce que personne n'a rien à apprendre de l'étiquette parce que les nouveaux caractères alloués peuvent être reçus sans même qu'on s'en aperçoive. L'étiquette elle-même n'enseigne rien sur les nouveaux caractères, qui seront reçus de toutes façons.

Donc, tant que la norme évolue de façon compatible, l'avantage apparent d'avoir des étiquettes qui identifient les versions n'est que cela : apparent. Mais il y a des inconvénients à de telles étiquettes dépendantes de la version : lorsque qu'une application plus ancienne reçoit des données accompagnées d'une étiquette plus récente, inconnue, elle peut échouer à reconnaître l'étiquette et être complètement incapable de traiter les données, tandis qu'une étiquette générique, connue, aurait déclanché un traitement très correct des données, qui pourraient bien ne contenir aucun des nouveaux caractères.

Le "bazar coréen" (ISO/CEI 10646 amendement 5) est un changement incompatible, contredisant en principe le caractère approprié du jeu de caractère MIME indépendant de la version comme décrit ci-dessus. Mais le problème de la compatibilité ne peut apparaître que lorsque des données contenant des caractères coréens Hangul codés selon Unicode 1.1 (ou son équivalent ISO/CEI 10646 avant l'amendement 5) et il n'y a probablement à s'inquiéter d'aucune de ces données, ce qui est la vraie raison pour laquelle le changement incompatible a été réputé acceptable.

En pratique donc, une étiquette indépendante de la version est garantie, pourvu que l'étiquette soit comprise comme se référant à toutes les versions après l'Amendement 5, et pourvu qu'aucun changement incompatible ne se produise réellement. Si des changements incompatibles se produisent dans une version ultérieure de ISO/CEI 10646, les jeux de caractères MIME définis ici resteront alignés sur la version précédente jusqu'à ce que l'IETF en décide spécifiquement autrement.

A.1 Enregistrement de UTF-16BE

À : ietf-charsets@iana.org
Objet : Enregistrement d'un nouveau jeu de caractères
Nom du jeu de caractères : UTF-16BE
Spécification publiée : cette spécification
Convient pour l'utilisation dans les types de contenu MIME sous le type de niveau supérieur "text" : Non
Adresse personnelle & de messagerie à contacter pour plus d'informations : Paul Hoffman < phoffman@imc.org >
Francois Yergeau < fyergeau@alis.com >

A.2 Enregistrement de UTF-16LE

À : ietf-charsets@iana.org
Objet : Enregistrement d'un nouveau jeu de caractères
Nom du jeu de caractères : UTF-16LE
Spécification publiée : cette spécification
Convient pour l'utilisation dans les types de contenu MIME sous le type de niveau supérieur "text" : Non
Adresse personnelle & de messagerie à contacter pour plus d'informations : Paul Hoffman < phoffman@imc.org >
Francois Yergeau < fyergeau@alis.com >

A.3 Enregistrement de UTF-16

À : ietf-charsets@iana.org
Objet : Enregistrement d'un nouveau jeu de caractères
Nom du jeu de caractères : UTF-16

Spécification publiée : cette spécification
Convient pour l'utilisation dans les types de contenu MIME sous le type de niveau supérieur "text" : Non
Adresse personnelle & de messagerie à contacter pour plus d'informations : Paul Hoffman < phoffman@imc.org >
Francois Yergeau < fyergeau@alis.com >

Adresse des auteurs

Paul Hoffman
Internet Mail Consortium
127 Segre Place
Santa Cruz, CA 95060 USA
mél : phoffman@imc.org

Francois Yergeau
Alis Technologies
100, boul. Alexis-Nihon, Suite 600
Montreal QC H4M 2P2 Canada
mél : fyergeau@alis.com

Déclaration complète de droits de reproduction

Remerciement
Le financement de la fonction d'éditeur des RFC est actuellement fourni par la Internet Society.