Groupe de travail Réseau
Request for Comments: 3629
STD: 63
RFC rendue obsolète : 2279
Catégorie : En cours de normalisation

F. Yergeau, Alis Technologies

novembre 2003

Traduction Claude Brière de L'Isle

UTF-8, un format de transformation de ISO 10646

Statut de ce mémoire

Le présent document spécifie un protocole Internet en cours de normalisation pour la communauté de l'Internet, et appelle à des discussion et des suggestions pour son amélioration. Prière de se reporter à l'édition actuelle du STD 1 "Normes des protocoles officiels de l'Internet" pour connaître l'état de normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé

La norme ISO/IEC 10646-1 définit un grand ensemble de caractères appelé ensemble de caractères universel (UCS, *Universal Character Set*) qui englobe la plupart des systèmes d'écriture du monde. Les codages proposés à l'origine par l'UCS, n'étaient cependant pas compatibles avec de nombreux protocoles et applications courants, et cela a conduit au développement de l'UTF-8, objet du présent mémoire. UTF-8 a pour caractéristique de préserver la gamme complète de l'US-ASCII, assurant la compatibilité avec les systèmes de fichiers, analyseurs et autres logiciels qui s'appuient sur les valeurs de l'US-ASCII mais sont transparents aux autre valeurs. Le présent mémoire rend obsolète la RFC 2279 et la remplace.

Table des matières

1. Introduction

La norme ISO/CEI 10646 [ISO.10646] définit un grand jeu de caractères appelé Jeu de caractères universel (UCS, *Universal Character Set*), qui renferme la plupart des systèmes d'écriture du monde. Le même jeu de caractères est défini par la norme Unicode [UNICODE], qui définit en plus des propriétés de caractère supplémentaires et autres détails d'application de grand intérêt pour les développeurs. Jusqu'à présent, changements dans Unicode et amendements et ajouts à ISO/CEI 10646 se sont répondus les uns les autres, de sorte que les répertoires de caractères et les allocations de codets

sont restés synchrones. Les comités de normalisation pertinents se sont engagés à maintenir ce très utile synchronisme.

La norme ISO/CEI 10646 et Unicode définissent plusieurs formes de codage de leur répertoire commun : UTF-8, UCS-2, UTF-16, UCS-4 et UTF-32. Dans une forme de codage, chaque caractère est représenté par une ou plusieurs unités de codage. Toutes les formes de codage standard de l'UCS excepté UTF-8 ont une unité de codage supérieure à un octet, ce qui rend leur utilisation difficile dans de nombreux protocoles et applications courantes qui supposent des caractères de 8 ou même 7 bits.
UTF-8, l'objet du présent mémoire, a une unité de codage de un octet. Il utilise tous les bits d'un octet, mais a l'avantage de préserver la gamme complète de l'[US-ASCII] : les caractères US-ASCII sont codés sur un octet avec la valeur US-ASCII normale, et tout octet avec une telle valeur ne peut correspondre qu'à un caractère US-ASCII, et à rien d'autre.

UTF-8 code les caractères UCS selon un nombre variable d'octets, où le nombre des octets, et la valeur de chacun, dépend de la valeur entière allouée au caractère dans la norme ISO/CEI 10646 (le numéro de caractère, aussi dit position de code, point de code ou valeur scalaire Unicode). Cette forme de codage a les caractéristiques suivantes (toutes les valeurs sont en hexadécimal) :

- Les numéros de caractère de U+0000 à U+007F (répertoire US-ASCII) correspondent aux octets 00 à 7F (valeurs US-ASCII à 7 bits). Une conséquence directe en est qu'une chaîne ASCII intégrale est aussi une chaîne UTF-8 valide.

- Les valeurs d'octet US-ASCII n'apparaissent pas autrement dans un flux de caractères codés en UTF-8. Cela assure la compatibilité avec les systèmes de fichiers ou autres logiciels (par exemple, la fonction printf() dans les bibliothèques C) qui analysent sur la base des valeurs US-ASCII mais sont transparents aux autres valeurs.

- Les allers-retours de conversion sont aisés entre UTF-8 et les autres formes de codage.

- Le premier octet d'une séquence multi-octets indique le nombre des octets de la séquence.

- Les valeurs d'octet C0, C1, F5 à FF n'apparaissent jamais.

- Les limites des caractères sont trouvées facilement à partir de n'importe quel point d'un flux d'octets.

- L'ordre de tri lexicographique par valeur d'octet des chaînes UTF-8 est le même que s'il était ordonné par numéro de caractère. Bien sûr, ceci est d'un intérêt limité dans la mesure où un ordre de tri fondé sur des numéros de caractère n'est presque jamais culturellement valide.

- L'algorithme de recherche rapide Boyer-Moore peut être utilisé avec les données UTF-8.

- Les chaînes UTF-8 peuvent très fiablement être reconnues comme telles par un algorithme simple, c'est-à-dire que la probabilité qu'une chaîne de caractères dans tout autre codage apparaisse comme de l'UTF-8 valide est faible, et diminue avec la longueur de la chaîne.

UTF-8 a été conçu en septembre 1992 par Ken Thompson, guidé par des critères de conception spécifiés par Rob Pike, avec l'objectif de concevoir un format de transformation de UCS utilisable de façon non perturbatrice dans le système d'exploitation Plan9. Le concept de Thompson a été porté dans les organes de normalisation par le groupe d'internationalisation conjoint X/Open XOJIG (voir [FSS_UTF]), sous les noms de FSS-UTF (variante FSS/UTF), UTF-2 et finalement UTF-8.

2. Conventions rédactionnelles

Dans le présent document, les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" sont à interpréter comme décrit dans la RFC 2119

Les caractères UCS sont désignés par la notation U+HHHH, où HHHH est une chaîne de 4 à 6 chiffres hexadécimaux représentant le numéro de caractère de la norme ISO/CEI 10646.

3. Définition de l'UTF-8

UTF-8 est défini par la norme Unicode [UNICODE]. Les descriptions et les formules se trouvent aussi à l'annexe D de la norme ISO/CEI 10646-1 [ISO.10646]

En UTF-8, les caractères de la gamme U+0000..U+10FFFF (la gamme UTF-16 accessible) sont codés en utilisant des séquences de 1 à 4 octets. Le seul octet d'une "séquence" de un octet a le bit de plus fort poids mis à 0, les 7 bits restants étant utilisés pour coder le numéro de caractère. Dans une séquence de n octets, n>1, l'octet initial a les n bits de plus fort poids mis à 1, suivis par un bit mis à 0. Le ou les bits restants de cet octet contiennent les bits du numéro de caractère à coder. Le ou les octets suivants ont tous leur bit de plus fort poids mis à 1 et le bit suivant mis à 0, laissant 6 bits dans chacun pour contenir les bits du caractère à coder.

Le tableau ci-dessous résume le format de ces différents types d'octet. La lettre x indique les bits disponibles pour coder les bits du numéro de caractère.

Gamme de numéro de caractère (hexadécimal	**Séquence d'octet UTF-8 (binaire)**
0000 0000-0000 007F	0xxxxxxx
0000 0080-0000 07FF	110xxxxx 10xxxxxx
0000 0800-0000 FFFF	1110xxxx 10xxxxxx 10xxxxxx
0001 0000-0010 FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

Le codage d'un caractère en UTF-8 se passe de la façon suivante :

1. Déterminer le nombre d'octets requis d'après le numéro de caractère et la première colonne du tableau ci-dessus. Il est important de noter que les lignes du tableau s'excluent mutuellement, c'est-à-dire qu'il n'y a qu'une seule façon valide de coder un caractère donné.

2. Préparer les bits de plus fort poids de l'octets selon la seconde colonne du tableau.

3. Remplir les bits marqués x d'après les bits du numéro de caractère, exprimé en binaire. On commence par mettre le bit de moindre poids du numéro de caractère dans la position de moindre poids du dernier octet de la séquence, puis on met le prochain bit de poids supérieur du numéro de caractère dans la position supérieure de l'octet, etc. Lorsque les x bits du dernier octet sont remplis, passer à l'avant dernier octet, puis au précédent, etc. jusqu'à ce que tous les x bits soient remplis.

La définition de UTF-8 interdit de coder les numéros de caractères entre U+D800 et U+DFFF, qui sont réservés pour être utilisés avec la forme de codage UTF-16 (comme paires de substitution) et ne représentent pas directement des caractères. Lors d'un codage en UTF-8 à partir de données en UTF-16, il est nécessaire de décoder d'abord les données UTF-16 pour obtenir les numéros de caractère, qui sont alors codés en UTF-8 comme décrit ci-dessus. Cela fait un contraste avec CESU-8 [CESU-8], qui est un codage de style UTF-8 qui n'est pas destiné à être utilisé sur l'Internet. CESU-8 fonctionne d'une façon similaire à celle d'UTF-8 mais code les valeurs de code UTF-16 (quantités de 16 bits) au lieu du numéro de caractère (codet). Cela conduit à des résultats différents pour les numéros de caractère au-dessus de 0xFFFF ; le codage CESU-8 de ces caractères N'EST PAS de l'UTF-8 valide.

Le décodage d'un caractère UTF-8 se passe comme suit :

1. Initialiser un numéro binaire avec tous les bits mis à 0. Jusqu'à 21 bits peuvent être nécessaires.
2. Déterminer quels bits codent le numéro de caractère à partir du nombre d'octets dans la séquence et la seconde colonne du tableau ci-dessus (les bits marqués x).
3. Distribuer les bits à partir de la séquence du numéro binaire, d'abord les bits de moindre poids à partir du dernier octet de la séquence et en continuant vers la gauche jusqu'à ce qu'aucun bits x ne reste. Le numéro binaire est maintenant égal au numéro de caractère.

Les mises en œuvre de l'algorithme de décodage ci-dessus DOIVENT se protéger contre les séquences de décodage invalides. Par exemple, une mise en œuvre peu élaborée pourrait décoder la très longue séquence UTF-8 C0 80 en caractère U+0000, ou la paire de substitution ED A1 8C ED BE B4 en U+233B4. Le décodage de séquences invalides peut avoir des conséquences pour la sécurité ou causer d'autres problèmes. Voir les Considérations pour la sécurité (Section 10) ci-dessous.

4. Syntaxe des séquences d'octet UTF-8

Pour l'agrément des développeurs qui utilisent l'ABNF, une définition de l'UTF-8 en syntaxe ABNF est données ci-dessous.

Une chaîne UTF-8 est une séquence d'octets représentant une séquence de caractères UCS. Une séquence d'octets n'est de l'UTF-8 valide que si elle satisfait à la syntaxe suivante, qui est dérivée des règles de codage de l'UTF-8 et est exprimée dans l'ABNF de la [RFC2234].

```
UTF8-octets  = *( UTF8-char )
UTF8-char    = UTF8-1 / UTF8-2 / UTF8-3 / UTF8-4
UTF8-1       = %x00-7F
UTF8-2       = %xC2-DF UTF8-tail
UTF8-3       = %xE0 %xA0-BF UTF8-tail / %xE1-EC 2( UTF8-tail ) / %xED %x80-9F UTF8-tail / %xEE-EF 2( UTF8-
tail )
UTF8-4       = %xF0 %x90-BF 2( UTF8-tail ) / %xF1-F3 3( UTF8-tail ) / %xF4 %x80-8F 2( UTF8-tail )
UTF8-tail    = %x80-BF
```

NOTE – La définition d'UTF-8 qui fait autorité est dans [UNICODE]. Cette grammaire est censée décrire la même chose qu'Unicode, mais ne prétend pas faire autorité. Les développeurs sont invités à s'appuyer sur la source qui fait autorité, plutôt que sur cet ABNF.

5. Versions des normes

La norme ISO/CEI 10646 est mise à jour de temps en temps par la publication d'amendements et parties additionnelles ; de même, de nouvelles versions de la norme Unicode sont publiées parfois. Chaque nouvelle version rend obsolète et remplace la précédente, mais les mises en œuvre, et plus significativement, 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 avec les données anciennes. En 1996, l'amendement 5 à l'édition 1993 de la norme ISO/CEI 10646 et Unicode 2.0 ont déplace et étendu le bloc du Coréen Hangul, rendant par là toutes les précédentes données qui contenaient des caractères Hangul invalides sous la nouvelle version. Unicode 2.0 a la même différence par rapport à Unicode 1.1. La justification de l'admission d'un tel changement incompatible était qu'il n'y avait pas de mises en œuvre majeures et pas de quantités significatives de données contenant du Hangul. L'incident a fait l(objet de plaisanteries sur le "désordre Coréen", et le comités concernés ont promis de ne jamais recommencer à faire de tels changements incompatibles (voir Unicode Consortium Policies [1]).

Les nouvelles versions, et en particulier tout changement incompatible, a des conséquences sur les étiquettes d'ensemble de caractères MIME, à discuter dans l'enregistrement MIME (Section 8).

6. Marque d'ordre d'octet (BOM, *Byte order mark*)

Le caractère UCS U+FEFF "ZERO WIDTH NO-BREAK SPACE" *(espace sans coupure de largeur zéro)* est aussi connu de façon informelle sous le nom de "BYTE ORDER MARK" (abrégé en "BOM", *marque d'ordre d'octet*). Ce caractère peut être utilisé comme une authentique "ESPACE SANS COUPURE DE LARGEUR ZÉRO" au sein d'un texte, mais le nom de BOM indique un second usage possible du caractère, qui est d'ajouter un caractère U+FEFF en tête d'un flux de caractères UCS comme "signature". Le receveur d'un tel flux sérié peut alors utiliser le caractère initial comme indication que le flux consiste en caractères UCS et aussi reconnaître quel codage UCS est impliqué et, avec des codages qui ont une unité de codage multi-octet, il a le moyen de reconnaître l'ordre de série des octets. UTF-8 ayant une unité de codage d'un seul octet, cette dernière fonction est inutile et le BOM apparaîtra toujours comme la séquence d'octets EF BB BF.

Il est important de comprendre que le caractère U+FEFF apparaissant dans toute position autre que le début d'un flux DOIT être interprété selon la sémantique de l'espace sans coupure de largeur zéro, et NE DOIT PAS être interprété comme une signature. Lorsque il est interprété comme une signature, la norme Unicode suggère qu'un caractère U+FEFF initial peut être enlevé avant de traiter le texte. Une telle suppression est nécessaire dans certains cas (par exemple, lors de l'enchaînement de deux chaînes, parce qu'autrement la chaîne résultante pourrait contenir une "ESPACE SANS COUPURE DE LARGEUR ZÉRO" non intentionnelle au point de connexion), mais peut affecter un processus externe à une couche différente (comme une signature numérique ou un compte de caractères) qui s'appuie sur la présence de tous les caractères du flux. Il est donc RECOMMANDÉ d'éviter d'enlever un U+FEFF initial interprété comme signature sans une bonne raison, et de l'ignorer au lieu de l'enlever quand c'est approprié (comme pour un affichage) et de ne l'enlever que lorsque c'est vraiment nécessaire.

U+FEFF en première position d'un flux PEUT être interprété comme une espace sans coupure de largeur zéro, et n'est pas

toujours une signature. Pour essayer de diminuer cette incertitude, Unicode 3.2 a ajouté un nouveau caractère, U+2060 "WORD JOINER" *(jonction de mot)*, avec exactement la même sémantique et utilisation que U+FEFF excepté la fonction de signature, et recommande vivement sont usage exclusif pour exprimer la sémantique de jonction de mot. En fin de compte, suivre cette recommandation rendra tout sauf certain qu'un U+FEFF est une signature, et non une "ESPACE SANS COUPURE DE LARGEUR ZÉRO" volontaire.

En attendant, l'incertitude subsiste malheureusement et peut affect les protocoles de l'Internet. Les spécifications de protocole PEUVEN restreindre l'usage de U+FEFF comme signature afin de réduire ou éliminer les mauvais effets potentiels de cette incertitude. Pour maintenir l'équilibre entre avantages (la réduction d'incertitude) et inconvénients (perte de la fonction de signature) de telles restrictions, il est utile de distinguer quelques cas :

- Un protocole DEVRAIT interdire l'utilisation de U+FEFF comme signature des éléments de protocole textuels qui le protocole oblige à être toujours en UTF-8, la fonction signature étant totalement inutile dans ces cas.

- Un protocole DEVRAIT aussi interdire l'utilisation de U+FEFF comme signature des éléments de protocole textuels pour lesquels le protocole fournit des mécanismes d'identification de codage de caractère, lorsqu'il est prévu que les mises en œuvre du protocole seront toujours en mesure d'utiliser correctement le mécanisme. Cela sera le cas lorsque les éléments de protocole sont maintenus étroitement sous le contrôle de la mise en œuvre depuis le moment de leur création jusqu'à celui de leur transmission (étiquetée de façon appropriée).

- Un protocole NE DEVRAIT PAS interdire l'usage de U+FEFF comme signature pour les éléments de protocole textuels pour lesquels le protocole ne fournit pas de mécanisme d'identification de codage de caractère, lorsqu'une interdiction serait inapplicable, ou quand on pense que les mises en œuvre du protocole ne seront pas en mesure d'utiliser toujours correctement le mécanisme. Ces deux derniers cas surviendront plus probablement avec de plus grands éléments de protocole comme les entités MIME, tout particulièrement lorsque les mises en œuvre du protocole obtiendront de telles entités de systèmes de fichiers, de protocoles qui n'ont pas de mécanisme d'identification de codage de charge utile (comme FTP) ou d'autres protocoles qui ne garantissent pas une identification correcte du codage de caractère (comme HTTP).

Lorsque un protocole interdit l'utilisation de U+FEFF comme signature pour un certain élément de protocole, tout U+FEFF initial dans cet élément de protocole DOIT alors être interprété comme "ESPACE SANS COUPURE DE LARGEUR ZÉRO". Lorsque un protocole N'INTERDIT PAS l'utilisation de U+FEFF comme signature pour un certain élément de protocole, les mises en œuvre DEVRAIENT alors être prêtes à traiter une signature dans cet élément et à réagir de façon appropriée : utiliser la signature pour identifier le codage de caractère si nécessaire et enlever or ignorer la signature en tant que de besoin.

7. Exemples

La séquence de caractères U+0041 U+2262 U+0391 U+002E "A<NOT IDENTICAL TO><ALPHA>." est codée en UTF-8 comme suit :

41	E2 89 A2	CE 91	2E

La séquence de caractères U+D55C U+AD6D U+C5B4 (du coréen "hangugeo", signifiant "langage coréen") est codé en UTF-8 comme suit :

ED 95 9C	EA B5 AD	EC 96 B4

La séquence de caractères U+65E5 U+672C U+8A9E (du japonais "nihongo", signifiant "langage japonais") est codé en UTF-8 comme suit :

E6 97 A5	E6 9C AC	E8 AA 9E

Le caractère U+233B4 (caractère chinois signifiant 'souche d'arbre') précédé d'un BOM UTF-8, est codé en UTF-8 comme suit :

EF BB BF	FO A3 8E B4

8. Enregistrement MIME

Le présent mémoire sert de fondement à l'enregistrement du paramètre de jeu de caractère MIME pour UTF-8, conformément à la [RFC2978]. La valeur du paramètre jeu de caractère *(charset)* est "UTF-8". Cette chaîne étiquette les types de support contenant du texte qui consiste en caractères provenant du répertoire de la norme ISO/IEC 10646 incluant tous les amendements au moins jusqu'à l'amendement 5 de l'édition de 1993 (bloc coréen), codé en une séquence d'octets utilisant le schéma de codage mentionné ci-dessus. UTF-8 convient pour une utilisation dans les types de contenu MIME du type de niveau supérieur "text".

Il vaut de noter que l'étiquette "UTF-8" ne contient pas d'identification de version, se référant de façon générique à la norme ISO/IEC 10646. Ceci est intentionnel, la raison en étant que :

Une étiquette de jeu de caractère MIME est conçue pour donner juste les informations nécessaires pour interpréter une séquence d'octets reçus sur le réseau comme séquence de caractères, et rien de plus (voir le paragraphe 2.2 de la [RFC2045]). Tant qu'une norme de jeu de caractère ne change pas de façon incompatible, les numéros de version ne servent à rien, parce qu'on ne gagne rien à apprendre de l'étiquette que de nouvelles allocations de caractères dont on ne sait rien peuvent être reçues. L'étiquette elle-même n'enseigne rien sur les nouveaux caractères, qui seront reçus de toutes façons.

Et donc, tant que les normes évoluent de façon compatible, l'avantage apparent d'avoir des étiquettes qui identifient les versions n'est qu'apparent. Mais il y a un inconvénient à de telles étiquettes variant selon la version : lorsqu'une application plus ancienne reçoit des données accompagnées d'un étiquette plus récente, inconnue, elle peut échouer à reconnaître l'étiquette et être complètement incapable de traiter les données, alors qu'une étiquette générique, connue va déclancher dans la plupart des cas un traitement correct des données, qui peuvent fort bien ne contenir aucun nouveau caractère.

Le "désordre coréen" (amendement 5 à la norme ISO/CEI 10646) est maintenant un changement incompatible, contredisant en principe l'adéquation d'une version indépendamment de l'étiquette de jeu de caractère MIME comme décrit ci-dessus. Mais le problème de compatibilité ne peut apparaître qu'avec des données contenant des caractères coréens Hangul codés selon Unicode 1.1 (ou son équivalent ISO/CEI 10646 avant l'amendement 5), et il est indiscutable qu'il n'y a pas à s'inquiéter pour de telles données, ce qui est la raison même qui a fait juger acceptable ce changement incompatible.

En pratique, il est garanti, sous réserve que l'étiquette soit comprise, qu'une étiquette indépendante de la version se réfère à toutes les versions postérieures à l'amendement 5, et pourvu qu'aucun changement incompatible ne survienne réellement. Si des changements incompatibles devaient survenir dans une version ultérieure de la norme ISO/CEI 10646, l'étiquette de jeu de caractère MIME définie ici resterait alignée avec la version précédente jusqu'à ce que l'IETF en décide spécifiquement autrement.

9. Considérations relatives à l'IANA

L'entrée pour UTF-8 dans le registre des jeu de caractères de l'IANA a été mis à jour pour pointer sur le présent mémoire.

10. Considérations pour la sécurité

Les développeurs de UTF-8 ont besoin de considérer les aspects pour la sécurité du traitement de séquences UTF-8 illégales. On peut concevoir des circonstances où un agresseur serait capable d'exploiter un analyseur UTF-8 imprudent en lui envoyant une séquence d'octet non permise par la syntaxe UTF-8.

Une forme particulièrement subtile de cette attaque peut être entreprise contre un analyseur qui effectue des vérifications de validité critiques pour la sécurité sur des formes codées en UTF-8 des ses entrées, mais interprète certaines séquences illégales d'octet comme des caractères. Par exemple, un analyseur pourrait interdire le caractère NUL lorsqu'il est codé comme la séquence d'un seul octet 00, mais permettre à tort la séquence illégale de deux octets C0 80 et l'interpréter comme un caractère NUL. Un autre exemple pourrait être celui d'un analyseur qui interdit la séquence d'octet 2F 2E 2E 2F ("/../"), tout en permettant la séquence d'octets illégale 2F C0 AE 2E 2F. Ce dernier exploit a réellement été utilisé dans un virus largement répandu qui attaquait les serveurs de la Toile en 2001 ; et donc, la menace contre la sécurité est très réelle.

Un autre problème de sécurité survient lors du codage en UTF-8 : la description de la norme ISO/CEI 10646 de l'UTF-8 permet le codage des numéros de caractères jusqu'à U+7FFFFFFF, donnant des séquences allant jusqu'à 6 octets. Il y a donc un risque de débordement de mémoire tampon si la gamme des numéros de caractères n'est pas explicitement limitée à U+10FFFF ou si le dimensionnement de mémoire tampon ne tient pas compte de la possibilité de séquences de 5 ou 6 octets.

La sécurité peut aussi être impactée par une caractéristique de plusieurs codages de caractères, incluant UTF-8 : la "même chose" (pour autant qu'un usager puisse le dire) peut être représentée par plusieurs séquences de caractères distinctes. Par exemple, un e avec un accent aigu peut être représenté par le caractère pré composé U+00E9 (E ACUTE) ou par la séquence canoniquement équivalente U+0065 U+0301 (E + COMBINING ACUTE). Bien que UTF-8 fournisse une seule séquence d'octets pour chaque séquence de caractères, l'existence de plusieurs séquences de caractères pour "la même chose" peut avoir des conséquences pour la sécurité chaque fois que la correspondance de chaîne, l'indexation, la recherche, le tri, la mise en correspondance d'expression régulière et la sélection sont impliquées. Un exemple serait la correspondance de chaîne d'un identifiant apparaissant dans un accréditif et dans des entrées de liste de contrôle d'accès. La question est soumise à des solutions fondes sur les formes de normalisation d'Unicode, voir [UAX15].

11. Remerciements

Les personnes suivantes ont participé à la rédaction du projet et à la discussion du présent mémoire : James E. Agenbroad, Harald Alvestrand, Andries Brouwer, Mark Davis, Martin J. Duerst, Patrick Faltstrom, Ned Freed, David Goldsmith, Tony Hansen, Edwin F. Hart, Paul Hoffman, David Hopwood, Simon Josefsson, Kent Karlsson, Dan Kohn, Markus Kuhn, Michael Kung, Alain LaBonte, Ira McDonald, Alexey Melnikov, MURATA Makoto, John Gardiner Myers, Chris Newman, Dan Oscarsson, Roozbeh Pournader, Murray Sargent, Markus Scherer, Keld Simonsen, Arnold Winkler, Kenneth Whistler et Misha Wolf.

12. Changements par rapport à la RFC 2279

- Restriction de la gamme de caractères à 0000-10FFFF (la gamme UTF-16 accessible).
- Faire d'Unicode la source de la définition normative de UTF-8, en gardant ISO/IEC 10646 comme référence pour les caractères.
- Renforcement de la terminologie. UTF-8 est maintenant décrit en termes de forme de codage du numéro de caractère. UCS-2 et UCS-4 ont presque disparu.
- Transformation de la note d'avertissement contre le décodage en séquences invalides en un NE DOIT PAS normatif.
- Ajout d'une nouvelle section sur le BOM UTF-8, avec un avis pour les protocoles.
- Retrait de l'enregistrement du jeu de caractères MIME UTF-8 UNICODE-1-1-UTF-8 suggéré.
- Ajout d'une syntaxe ABNF pour les séquences d'octets UTF-8 valides
- Développement de la section Considérations pour la sécurité, en particulier l'impact de la normalisation Unicode.

13. Références normatives

[RFC2119] Bradner, S., "Mots clés à utiliser dans les RFC pour indiquer les exigences de niveau", BCP 14, RFC 2119, mars 1997.

[ISO.10646] Organisation International de normalisation, "Technologies de l'information – Jeu de caractères universel codé sur plusieurs octets (UCS)", norme ISO/CEI 10646, composée de ISO/CEI 10646-1:2000, "Technologies de l'information -- Jeu de caractères universel codé sur plusieurs octets (UCS) – Partie 1 : Architecture et plan multilingue de base", ISO/CEI 10646-2:2001, "Technologies de l'information – Jeu de caractères universel codé sur plusieurs octets (UCS) -- Partie 2 : Plans supplémentaires" et ISO/CEI 10646-1:2000/Amendement 1:2002, "Symboles mathématiques et autres caractères".

[UNICODE] The Unicode Consortium, "The Unicode Standard -- Version 4.0", défini par The Unicode Standard, Version 4.0 (Boston, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1), avril 2003, <http://www.unicode.org/unicode/standard/ versions/enumeratedversions.html#Unicode_4_0_0>.

14. Références informatives

[CESU-8] Phipps, T., "Unicode Technical Report #26: Compatibility Encoding Scheme for UTF-16: 8-Bit (CESU-8)", UTR 26, avril 2002, <http://www.unicode.org/unicode/reports/tr26/>.

[FSS_UTF] X/Open Company Ltd., "X/Open Preliminary Specification -- File System Safe UCS Transformation Format (FSS-UTF)", mai 1993, <http://wwwold.dkuug.dk/jtc1/sc22/wg20/docs/ N193-FSS-UTF.pdf>.

[RFC2045] N. Freed et N. Borenstein, "Extensions de messagerie Internet multi objets (MIME) Partie 1 : Format des corps de messages Internet", RFC 2045, novembre 1996.

[RFC2234] D. Crocker et P. Overell, "BNF augmenté pour les spécifications de syntaxe : ABNF", RFC 2234, novembre 1997.

[RFC2978] N. Freed et J. Postel, "Procédures de l'IANA pour l'enregistrement des jeux de caractères", BCP 19, RFC 2978, octobre 2000.

[UAX15] Davis, M. et M. Duerst, "Unicode Standard Annex #15: Unicode Normalization Forms", An integral part of The Unicode Standard, Version 4.0.0, avril 2003, <http://www.unicode.org/unicode/reports/tr15>.

[US-ASCII] American National Standards Institute, "Coded Character Set - 7-bit American Standard Code for Information Interchange", ANSI X3.4, 1986.

15. URI

[1] <http://www.unicode.org/unicode/standard/policies.html>

16. Déclaration de droits de propriété intellectuelle

17. Adresse de l'auteur

Francois Yergeau
Alis Technologies
100, boul. Alexis-Nihon, bureau 600
Montreal, QC H4M 2P2
Canada
téléphone : +1 514 747 2547
Fax : +1 514 747 2561
mèl : fyergeau@alis.com

18. Déclaration complète de droits de reproduction

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