Groupe de travail Réseau J. Klensin,
Request for Comments : 5137 février 2008
BCP 137
Catégorie : Bonnes pratiques actuelles Traduction Claude Brière de L'Isle

Échappement ASCII des caractères Unicode

Résumé
Dans un certain nombre de circonstances un mécanisme d'échappement est nécessaire en conjonction avec un protocole pour coder des caractères qui ne peuvent pas être représentés ou transmis directement. Avec le codage ASCII, l'échappement traditionnel a été la valeur numérique décimal ou hexadécimale du caractère, écrite de diverses façons. Le passage à Unicode, où les caractères occupent deux octets ou plus et peuvent être codés de plusieurs formes différentes, a encore compliqué la question des échappements. Le présent document discute certaines options utilisées actuellement et discute des considérations relatives au choix de celles à utiliser dans les nouvaux protocoles de l'IETF, et des protocoles qui sont maintenant en cours d'internationalisation.

Table des Matières

1. Introduction

1.1 Contexte et fondements

Dans un certain nombre de circonstances un mécanisme d'échappement est nécessaire en conjonction avec un protocole pour coder des caractères qui ne peuvent pas être représentés ou transmis directement. Avec le codage ASCII [ASCII], l'échappement traditionnel était la valeur numérique décimale ou hexadécimale du caractère, écrit de diverses façons. Par exemple, dans différents contextes, on a vu %dNN ou %NN pour la forme décimale, %NN, %xNN, X'nn', et %X'NN' pour la forme hexadécimale. "%NN" est devenu populaire ces dernières années pour représenter une valeur hexadécimale sans autre qualification, peut-être par suite de son utilisation dans les URL et leur prévalence. Il y a même certaines applications

dans lesquelles des formes octales sont utilisées et, bien qu'elles ne soient pas généralisées, les formes MIME Quoted-Printable et Encoded-word peuvent être vues comme encore un autre ensemble d'échappements. Donc, même pour des cas très simples de constructions ASCII et d'extensions standard d'ASCII, comme celles de la famille ISO 8859, on a vécu avec plusieurs formes différentes d'échappement, chacune résultant de sa propre histoire.

Quand on passe à Unicode [Unicode], [ISO10646], où les caractères occupent deux octets ou plus et peuvent être codés dans plusieurs formes différentes, la question des échappements devient encore plus compliquée. Unicode représente les caractères comme des codets : les valeurs numériques de 0 à hex 10FFFF. Quand on se réfère à des codets dans le texte, ils sont représentés en utilisant la notation dite "U+", comme des valeurs de U+0000 à U+10FFFF. Quand ils sont mis dans des octets, ces codets peuvent être représentés sous des formes différentes :

- o en UTF-8 avec un à quatre octets [RFC3629]
- o en UTF-16 avec deux ou quatre octets (ou une ou deux seizets -- unités de16 bits)
- o en UTF-32 avec exactement quatre octets (ou une unité de 32 bits)

Quand on échappe des caractères, on a vu une très large utilisation de représentations hexadécimales des formes sérialisées et de variations sur la notation U+, appelées des échappements de codets.

En accord avec les recommandations de bonnes pratiques existantes [RFC2277], de nouveaux protocoles qui sont exigés pour porter du contenu textuel pour l'usage humain DEVRAIENT être conçus de telle façon que le répertoire complet de caractères Unicode puisse être représenté dans ce texte.

Le présent document propose que les protocoles existants soient internationalisés, et que ceux qui ont besoin d'un mécanisme d'échappement, DEVRAIENT utiliser une variante contextuellement appropriée sur les références aux codets comme décrit à la Section 2 sauf si d'autres considérations contre-balancent celles qu'on décrit ici.

Cette recommandation n'est pas applicable aux protocoles qui acceptent déjà l'UTF-8 natif ou autre codage de Unicode. En général, quand des protocoles sont internationalisés, il est préférable d'accepter ces formes plutôt que d'utiliser des échappements. Cette recommandation s'applique aux cas, incluant les arrangements de transition, dans lesquels ce n'est pas pratique.

En plus des contextes de protocoles traités par la présente spécification, les échappements pour représenter les caractères Unicode apparaissent aussi dans les présentations aux utilisateurs, c'est-à-dire, dans les interfaces d'utilisateur (UI, *User Interface*). Les formats spécifiés dans le présent document, et le raisonnement, peuvent être aussi applicables dans les contextes d'UI, mais ceci n'est pas une proposition de normaliser les UI ou les formes de présentation.

Le présent document ne fait pas de recommandations générales pour le traitement des chaînes Unicode ou de leur contenu. Il suppose que les chaînes qu'on peut vouloir échapper sont valides et raisonnables et que la définition de "valide et raisonnable" relève d'autres documents. Les recommandations sur le traitement général des chaînes Unicode peuvent être trouvées à de nombreux endroits, incluant la norme Unicode elle-même, le modèle de caractères du W3C [CharMod], ainsi que des règles spécifiques dans des protocoles individuels.

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].

La terminologie supplémentaire spécifique de Unicode apparaît dans [UnicodeGlos], mais n'est pas nécessaire pour comprendre la présente spécification.

1.3 Liste de discussion

La discussion de ce document devrait être adressée à la liste de diffusion discuss@apps.ietf.org.

2. Codages représentant les codets Unicode : position de code contre octets UTF-8 ou UTF-16

Il y a deux familles majeures de façons d'échapper les caractères Unicode. L'une utilise le codet dans certaines représentations (voir le paragraphe suivant) l'autre code les octets du codage UTF-8 ou autre codage dans certaines représentations. D'autres options sont possibles, mais elles sont rares en pratique. La présente spécification recommande

que, en l'absence de raisons impérieuses de faire autrement, les codets Unicode DEVRAIENT être utilisés plutôt qu'une représentation d'octets UTF-8 (ou UTF-16). Il y a plusieurs raisons à cela, parmi lesquelles :

- Une raison du succès de nombreux protocoles de l'IETF est qu'ils utilisent des formes de texte interprétables par l'homme pour communiquer, plutôt que des codages qui exigent généralement des programmes informatiques (ou la simulation manuelles d'algorithmes) pour les décoder. Cela suggère que la forme de présentation devrait faire référence aux tableaux de caractères Unicode et de façon aussi simple que possible.

- À cause de la nature de UTF-8, pour qu'un humain interprète une représentation numérique décimale ou hexadécimale d'octets UTF-8, il faut une ou plusieurs étapes de décodage pour déterminer un codet Unicode qui peut être utilisé pour chercher le caractère dans un tableau. Cela peut être approprié dans certains cas où le but est réellement de représenter la forme UTF-8 mais, en général, cela obscurcit juste les informations désirées et rend les erreurs plus probables et le débogage plus difficile.

- Sauf pour les caractères du sous ensemble ASCII d'Unicode (U+0000 à U+007F) la forme codet est généralement plus compacte que les formes fondées sur le codage des octets UTF-8, parfois beaucoup plus compacte.

Les mêmes considérations qui s'appliquent à la représentation des octets du codage UTF-8 s'appliquent aussi à des codages ACE plus compacts comme le codage "bootstring" [RFC3492] avec ou sans son profil "Punycode".

Des considérations similaires s'appliquent au codage UTF-16, comme la forme \uNNNN utilisée dans Java (voir le paragraphe 6.3). Bien que ces formes soient équivalentes aux références de codets pour le plan de base multi langues (BMP, *Basic Multilingual Plane*) (plan 0) un processus de décodage en deux étapes est nécessaire pour traiter les substituts pour accéder aux plans supérieurs.

3. Référence aux caractères Unicode

Sans considération des décisions prises sur les échappements pour les caractères Unicode dans le protocole ou les contextes similaires, un texte qui se réfère à un codet Unicode DEVRAIT utiliser la syntaxe U+NNNN[N[N]], comme spécifiée dans la norme Unicode, où la chaîne NNNN... consiste en nombres hexadécimaux. Le texte qui contient réellement un caractère Unicode DEVRAIT utiliser une syntaxe plus convenable au traitement automatique.

4. Syntaxe des échappements de codets

Il y a de nombreuses options pour les échappements de codets, dont certaines sont récapitulées ci-dessous. Toutes sont équivalentes en contenu et sémantique -- les différences résident dans la syntaxe. Le meilleur choix de syntaxe pour un protocole particulier ou autre application dépend de cette application : une forme peut simplement "aller" mieux dans un certain contexte que dans d'autres. Il est cependant clair que les valeurs hexadécimales sont préférables aux autres solutions : les systèmes fondés sur des décalages décimaux ou octaux NE DEVRAIENT PAS être utilisés.

Comme la présente spécification ne recommande pas une syntaxe spécifique, les spécifications de protocole qui utilisent des échappements DOIVENT définir la syntaxe qu'elles utilisent, incluant tous les échappements nécessaires pour permettre que la séquence d'échappement soit utilisée littéralement.

Le concepteur d'application devrait au moins considérer les facteurs suivants pour choisir un format :

- Si des protocoles similaires ou en relation utilisent déjà une forme, il peut être meilleur de choisir cette forme pour la cohérence et la prévisibilité.

- Un codet Unicode peut tomber dans la gamme de U+0000 à U+10FFFF. Différents systèmes d'échappement peuvent utiliser quatre, cinq, six, ou huit chiffres hexadécimaux. Pour éviter des manœuvres habiles de syntaxe et le risque conséquent de confusion et d'erreurs, les formes qui utilisent des délimiteurs explicites de chaîne sont généralement préférés aux autres solutions. Dans de nombreux contextes, des délimiteurs en paire symétrique sont plus faciles à reconnaître et comprendre que ceux qui sont sans relation visuelle.

- Les formes syntaxiques qui commencent par "\u", sans délimiteur explicite, ont été utilisées dans plusieurs systèmes d'échappement différents, incluant la syntaxe à quatre ou huit chiffres de C [ISO-C] (voir le paragraphe 6.1) le codage UTF-16 de Java [Java] (voir le paragraphe 6.3) et certains arrangements qui peuvent suivre le "\u" avec quatre, cinq, ou

six chiffres. La confusion possible sur quelle option est réellement utilisée peut plaider contre l'utilisation de toutes ces formes.

- Les formes qui exigent de décoder des paires de substitution partagent la plupart des problèmes qui apparaissent avec le codage des octets UTF-8. Les protocoles Internet NE DEVRAIENT PAS utiliser de paires de substitution.

5. Variantes de présentation recommandées pour l'échappement de codet Unicode

Il y a un certain nombre de façons différentes de représenter la position d'un codet Unicode. Aucune d'elles ne paraît être "la meilleure" dans tous les contextes. De plus, quand un échappement est nécessaire pour le mécanisme d'échappement lui-même, l'optimal parmi eux peut différer de celui d'un autre contexte.

Certaines formes qui sont d'utilisation populaire et qui pourraient raisonnablement être envisagées dans un certain protocole sont décrites ci-dessous et identifiées avec un contexte d'utilisation courante quand c'est faisable. Les deux de cette section sont d'utilisation recommandée dans les protocoles de l'Internet. D'autres aussi populaires apparaissent à la Section 6 avec une discussion de leurs inconvénients.

5.1 Barre oblique inverse U avec délimiteurs

Une des formes recommandées est une variante des nombreuses formes qui commencent par "\u" (voir, par exemple, le paragraphe 6.1) mais utilise des délimiteurs explicites pour les raisons discutées ailleurs.

Spécifiquement, en ABNF [RFC5234],

EmbeddedUnicodeChar = %x5C.75.27 4*6HEXDIG %x27
; commençant par la minuscule "\u" et "" et se terminant par "". Noter que les codages sont considérés être des abstractions pour les caractères pertinents, non les désignations d'octets spécifiques.

HEXDIG = "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9" / "A" / "B" / "C" / "D" / "E" / "F"
; effectivement identique à la définition de la RFC 5234.

Les concepteurs de protocoles d'applications qui utilisent cette forme devraient spécifier un moyen d'échapper la barre oblique inverse introductive ("\") si nécessaire. "\\" est une possibilité évidente, mais pas la seule.

5.2 XML et HTML

L'autre forme recommandée est celle utilisée dans XML. Elle utilise la forme "&#xNNNN;". Comme la forme Perl (paragraphe 6.2) cette forme a un clair délimiteur de fin, réduisant l'ambiguïté. HTML utilise une forme similaire, mais le point-virgule peut être omis dans certains cas. Si c'est fait, les avantages du délimiteur disparaissent de sorte que la forme HTML sans le point-virgule NE DEVRAIT PAS être utilisée. Cependant, ce format est souvent considéré comme pas beau et bizarre en dehors de son contexte HTML, XML, et similaires, natif.

En ABNF :

EmbeddedUnicodeChar = %x26.23.78 2*6HEXDIG %x3B
; commence par "&#x" et se termine par ";"

Noter qu'un "&" littéral peut être exprimé par "&" quand on utilise ce style.

6. Formes normalement non recommandées

6.1 Langage de programmation C : barre oblique inverse -U

Les formes \UNNNNNNNN (pour tout caractère Unicode) et \uNNNN (pour les caractères Unicode dans le plan 0) sont utilisées dans le langage de programmation C [ISO-C] quand un échappement ASCII pour des caractères Unicode incorporés est nécessaire.

Les inconvénients de cette forme peuvent être significatifs. D'abord, l'utilisation d'une variation de casse (entre le "u" pour la forme à quatre chiffres et "U" pour la forme à huit chiffres) peut ne pas sembler naturelle dans des environnements où la

majuscule et la minuscule sont généralement considérées comme équivalentes et pourraient être confondues par les gens qui ne sont pas très familiarisés avec les alphabets fondés sur les caractères latins (bien que ces gens pourraient être encore plus troublés quand ils lisent le texte et les explications pertinentes en anglais). Ensuite, comme discuté à la Section 4, le fait même qu'il y ait plusieurs conventions différentes qui commencent par \u ou \U peut devenir une source de confusion lorsque les gens font des hypothèses incorrectes sur ce qu'ils sont en train de regarder.

6.2 Perl : chaîne hexadécimale

Perl utilise la forme \x{NNNN...}. L'avantage de cette forme est qu'il y a des délimiteurs explicites, résolvant la question des chaînes de longueur variable ou d'utiliser le mécanisme de changement de casse de la forme proposée pour distinguer entre le plan 0 et les formes plus générales. Certains autres langages de programmation inclineraient en faveur des formes X'NNNN...' pour les chaînes hexadécimales et peut-être U'NNNN...' pour les chaînes spécifiques de Unicode, mais ces formes ne semblent pas être utilisées autour de l'IETF.

Noter qu'il y a une possible ambiguïté dans la façon dont des séquences de deux caractères ou de faible numéro de cette notation sont comprises, c'est-à-dire, que les octets dans la gamme \x(00) à \x(FF) peuvent être construits comme étant dans le jeu local de caractères, non comme des codets Unicode. À cause de cette ambiguïté apparente, et parce que les documents de l'IETF ne contiennent pas de dispositions pour les "pragmas" (voir [PERL] pour des informations sur le "codage" de "pragma" dans Perl et autres détails) la forme Perl devrait être utilisée avec une extrême prudence, si elle doit l'être.

6.3 Java : UTF-16 avec échappement

Java [Java] utilise la forme \uNNNN, mais comme référence aux valeurs UTF-16, pas à des codets Unicode. Bien qu'il utilise une syntaxe similaire à celle décrite au paragraphe 6.1, cette relation à l'UTF-16 le rend, à bien des aspects, plus similaire aux codages de UTF-8 discutés ci-dessus qu'à un échappement qui désigne des codets Unicode. Noter que la forme UTF-16, et donc, la notation d'échappement de Java, ne peut représenter des caractères en dehors du plan 0 (c'est-à-dire, au dessus de U+FFFF) que par l'utilisation de paires de substituts, soulevant les mêmes problèmes que l'utilisation des octets UTF-8 discutée précédemment. Pour les caractères dans le plan 0, la forme Java est indistinguable de la forme seulement plan 0 décrite au paragraphe 6.1. Au moins pour cette seule raison, il NE DEVRAIT PAS être utilisé comme un échappement sauf dans les contextes Java dans lesquels il est naturel.

7. Considérations sur la sécurité

Le présent document propose un ensemble de règles pour le codage des caractères Unicode quand d'autres considérations ne s'appliquent pas. Comme tous les codages recommandés sont sans ambiguïté et que des questions de normalisation ne sont pas impliquées, il ne devrait pas introduire de problème de sécurité qui ne serait pas présent par suite du simple usage de caractères non ASCII, quel que soit leur codage. Les mécanismes suggérés devraient légèrement diminuer les risques de confusion des utilisateurs avec des caractères codés en rendant l'identité des caractères utilisés un peu plus évidente que dans certaines des alternatives.

Un mécanisme d'échappement comme celui spécifié dans le présent document peut permettre que les caractères soient représentés de plus d'une façon. Lorsque le logiciel interprète la forme échappée, il y a un risque que des vérifications de sécurité, et toute vérification nécessaire pour, par exemple, des formes minimales ou normalisées, soient faites au mauvais endroit.

8. Remerciements

Le présent document a été produit en réponse à une série de discussions au sein de la zone Applications de l'IETF et au titre d'un travail sur l'internationalisation de la messagerie électronique et de la mise à jour des noms de domaine internationalisés. C'est une synthèse d'un grand nombre de discussions, des commentaires des participants, dont ils sont remerciés avec gratitude. L'aide de Mark Davis pour construire une liste des autres présentations et du choix parmi elles a été particulièrement important.

Tim Bray, Peter Constable, Stephane Bortzmeyer, Chris Newman, Frank Ellermann, Clive D.W. Feather, Philip Guenther, Bjoern Hoehrmann, Simon Josefsson, Bill McQuillan, der Mouse, Phil Pennock, et Julian Reschke ont fourni une relecture attentive, des corrections et suggestions sur les divers projets de travail qui ont précédé ce document. Toutes ensemble, ces

suggestions ont motivé une révision significative de ce document et de ses recommandations entre la version 00 et la version 01 et d'autres améliorations dans les versions suivantes.

9. Références

9.1 Références normatives

[ISO-10646] Norme ISO/CEI 10646, "Technologie de l'information – Jeu de caractères universel codé sur plusieurs octets (UCS) - Partie 1 : Architecture et plan multilingue de base", décembre 2003.

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997. *(MàJ par* RFC8174)

[RFC3629] F. Yergeau, "UTF-8, un format de transformation de la norme ISO 10646", STD 63, novembre 2003.

[RFC5234] D. Crocker, P. Overell, "BNF augmenté pour les spécifications de syntaxe : ABNF", janvier 2008. (STD0068)

[Unicode] The Unicode Consortium, "The Unicode Standard, Version 5.0", 2006. (Addison-Wesley, 2006. ISBN 0-321-48091-0).

9.2 Références pour information

[ASCII] American National Standards Institute, "USA Code for Information Interchange", ANSI X3.4-1968, 1968. ANSI X3.4-1968 a été remplacé par de nouvelles versions avec de petites modifications, mais la version 1968 reste celle de référence pour l'Internet.

[ISO-C] Norme ISO/CEI 9899:1999, "Technologie de l'information -- Langages de programmation -- C", Organisation Internationale de normalisation, 1999.

[Java] Sun Microsystems, Inc., "Java Language Specification, Third Edition", 2005, <http://java.sun.com/docs/books/jls/third_edition/html/lexical.html#95413p>.

[PERL] Hietaniemi, J., "perluniintro", Perl documentation 5.8.8, 2002, <http://perldoc.perl.org/perluniintro.html>.

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

[RFC3492] A. Costello, "Punycode : Codage Bootstring d'Unicode pour les noms de domaine internationalisés dans les applications (IDNA)", mars 2003. *(P.S.)*

[UnicodeGlos] The Unicode Consortium, "Glossary of Unicode Terms", juin 2007, <http://www.unicode.org/glossary>.

[CharMod] Duerst, M., "Character Model for the World Wide Web 1.0", W3C Recommendation, février 2005, <http://www.w3.org/TR/charmod/>.

Appendice A. Syntaxe formelle pour formes non recommandées

Bien que la syntaxe des formes d'échappement qui ne sont pas recommandées ci-dessus (Section 6) ne soit pas donnée soit en ligne avec l'espoir de décourager leur utilisation, elles sont fournies dans cet appendice dans l'espoir que ceux qui choisissent de les utiliser le feront de façon conséquente. Le lecteur est averti que certaines de ces formes ne sont pas définies avec précision dans les spécifications originales et que d'autres ont évolué au fil du temps de façons qui ne sont pas précisément cohérentes. Par conséquent, ces définitions ne sont pas normatives et peuvent ne même pas correspondre précisément à des interprétations raisonnables de leurs sources.

La définition de "HEXDIG" pour les formes qui suivent est donnée au paragraphe 5.1.

A.1 Forme du langage de programmation C

Spécifiquement, en ABNF [RFC5234],

```
EmbeddedUnicodeChar =  BMP-form / Full-form

BMP-form =  %x5C.75 4HEXDIG          ; commençant par la minuscule "\u"
                                     ; les codages sont considérés comme des abstractions pour les
                                     caractères pertinents, pas des désignations d'octets spécifiques.

Full-form =  %x5C.55 8HEXDIG         ; commençant par la majuscule "\U"
```

A.2 Forme Perl

```
EmbeddedUnicodeChar = %x5C.78 "{" 2*6HEXDIG "}"      ; commence par "\x"
```

A.3 Forme Java

```
EmbeddedUnicodeChar = %x5C.7A 4HEXDIG                ; commence par "\u"
```

Adresse de l'auteur

John C Klensin
1770 Massachusetts Ave, #322
Cambridge, MA 02140
USA
téléphone : +1 617 245 1457
mél : john-ietf@jck.com

Déclaration complète de droits de reproduction