Groupe de travail Réseau
Request for Comments : 3641
Catégorie : En cours de normalisation

S. Legg, Adacel Technologies
octobre 2003
Traduction Claude Brière de L'Isle

Règles génériques de codage de chaîne (GSER) pour les types ASN.1

Statut du présent mémoire
Le présent document spécifie un protocole de l'Internet en cours de normalisation pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Protocoles officiels de l'Internet" (STD 1) pour voir l'état de normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé
Le présent document définit un ensemble de règles de codage de notation de syntaxe abstraite numéro un (ASN.1) appelé les règles génériques de codage de chaîne (GSER, *Generic String Encoding Rules*) qui produit un codage de texte lisible par l'homme pour les valeurs de tout type de données en ASN.1.

Table des Matières

1. Introduction

Le présent document définit un ensemble de règles de codage ASN.1 [X.680] appelé règles génériques de codage de chaîne (GSER, *Generic String Encoding Rules*) qui produit un codage de chaîne de caractères UTF-8 lisible par l'homme [RFC2279]

de valeurs ASN.1 de tout type arbitraire ASN.1.

Noter que "valeur ASN.1" ne signifie pas une valeur codée selon les règles de codage de base (BER, *Basic Encoding Rules*) [X.690]. La valeur ASN.1 est un concept abstrait qui est indépendant de tout codage particulier. BER est juste un des codages possibles d'une valeur ASN.1.

GSER se fonde sur une notation de valeur ASN.1 [X.680], avec des changements pour accommoder l'utilisation de la notation comme syntaxe de transfert, et pour prendre en charge des codages ad hoc bien établis pour les types de données de répertoire du protocole léger d'accès à un répertoire (LDAP, *Lightweight Directory Access Protocol*) [RFC3377].

Bien que principalement destinées à définir le codage spécifique de LDAP des nouvelles syntaxes d'attributs LDAP et des syntaxes d'assertion, ces règles de codage peuvent aussi être utilisées dans d'autres domaines où serait utile un rendu lisible par l'homme des valeurs ASN.1.

Référencer GSER est suffisant pour définir un codage de texte lisible par l'homme pour les valeurs d'un type spécifique d'ASN.1, cependant, d'autres spécifications peuvent souhaiter fournit une description personnalisée en format Backus-Naur augmenté (ABNF) [RFC2234], indépendant de l'ASN.1, comme un moyen pratique pour la mise en œuvre (un ABNF équivalent pour le codage GSER aux types ASN.1 couramment utilisés dans les syntaxes LDAP est fourni dans un document séparé [RFC3642]). Une telle spécification DEVRAIT déclarer, si il y a une discordance entre l'ABNF personnalisé et le codage GSER défini dans le présent document, que le codage GSER a la préséance.

2. Conventions

Tout au long du présent document, "type" devra être interprété comme signifiant un type ASN.1, et "valeur" devra être compris comme une valeur ASN.1.

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].

3. Règles génériques de codage de chaîne

Le codage GSER d'une valeur de tout type ASN.1 est décrit par l'ABNF suivant [RFC2234] :

Valeur = BitStringValue / BooleanValue / CharacterStringValue / ChoiceValue / EmbeddedPDVValue / EnumeratedValue / ExternalValue / GeneralizedTimeValue / IntegerValue / InstanceOfValue / NullValue / ObjectDescriptorValue / ObjectIdentifierValue / OctetStringValue / RealValue / RelativeOIDValue / SequenceOfValue / SequenceValue / SetOfValue / SetValue / StringValue / UTCTimeValue / VariantEncoding

L'ABNF pour chacune des règles ci-dessus est donné dans les paragraphes qui suivent.

3.1 Notations de référence de type

La valeur d'un type qui a un nom de type défini est codée conformément à la définition du type sur le côté droit de l'allocation de type pour le nom de type.

La valeur d'un type noté par l'utilisation d'un type paramétré avec des paramètres réels est codé conformément au type paramétré avec les références factices [X.683] substituées aux paramètres réels.

Une valeur d'un type étiqueté ou contraint est codée comme une valeur du type sans respectivement l'étiquette ou la contrainte. Les étiquettes n'apparaissent pas dans les codages de chaîne définis dans le présent document. Voir les détails de la notation des contraintes de l'ASN.1 dans [X.680] et [X.682].

Une valeur d'un type ouvert noté par un ObjectClassFieldType *(type de champ de classe d'objet)* (Clause 14 de [X.681]) est codée conformément au type spécifique de la valeur.

Une valeur d'un type fixé noté par un ObjectClassFieldType est codée conformément à ce type fixé.

Une valeur d'un type selection est codé conformément au type référencé par le type selection.

Une valeur d'un type décrit par la notation TypeFromObject (Clause 15 de [X.681]) est codé conformément au type noté.

Une valeur d'un type décrit par la notation ValueSetFromObjects (Clause 15 de [X.681]) est codé conformément au type gouvernant.

3.2 Types de chaînes à caractères restreints

Le contenu d'une valeur de chaîne est codé comme une chaîne de caractères UTF-8 entre des guillemets, sans considération du type de chaîne ASN.1. Selon le type de chaîne ASN.1 et la représentation interne de ce type de chaîne par une application, une traduction en, ou du, codage de caractères UTF-8, peut être nécessaire. NumericString, PrintableString, IA5String, et VisibleString (ISO646String) sont compatibles avec UTF-8 et n'exigent aucune traduction. BMPString (UCS-2) et UniversalString (UCS-4) ont une transposition directe de et en UTF-8 [RFC2279]. Pour les types de chaîne restants, voir [X.680]. Tous les guillemets incorporés dans la chaîne de caractères UTF-8 résultante sont échappés en répétant les caractères guillemets *(double quote)*.

Une valeur des types NumericString, PrintableString, TeletexString (T61String), VideotexString, IA5String, GraphicString, VisibleString (ISO646String), GeneralString, BMPString, UniversalString ou UTF8String est codée conformément à la règle <StringValue> *(valeur de chaîne)*.

```
StringValue      = dquote *SafeUTF8Character dquote     ("caractère UTF-8 sûr")
dquote           = %x22                                 ; " (guillemets)
SafeUTF8Character = %x00-21 / %x23-7F /                 ; moins ASCII guillemets
                   dquote dquote /                      ; guillemets échappés
                   %xC0-DF %x80-BF /                    ; caractère UTF-8 de deux octets
                   %xE0-EF 2(%x80-BF) /                 ; caractère UTF-8 de trois octets
                   %xF0-F7 3(%x80-BF)                   ; caractère UTF-8 de quatre octets
```

Une valeur du type GeneralizedTime *(temps généralisé)*, UTCTime *(heure universelle coordonnée)* ou ObjectDescriptor *(descripteur d'objet)* est codée comme une valeur de chaîne. GeneralizedTime et UTCTime utilisent le jeu de caractères VisibleString *(chaîne visible)* de sorte que la conversion en UTF-8 est triviale. ObjectDescriptor utilise le type GraphicString *(chaîne graphique)*.

```
GeneralizedTimeValue  = StringValue
UTCTimeValue          = StringValue
ObjectDescriptorValue = StringValue
```

3.3 Types ChoiceOfStrings

Il n'est pas rare que les spécifications d'ASN.1 définissent des types qui offrent un CHOIX entre deux ou plusieurs solutions de remplacement de type de chaîne ASN.1, où la solution particulière choisie ne porte pas de signification sémantique (DirectoryString [X.520] en étant le principal exemple). De tels types sont définis pour éviter d'avoir à utiliser un codage de caractères compliqué pour toutes les valeurs lorsque la plupart d'entre elles pourraient utiliser un type de chaîne plus simple, ou d'avoir à faire avec des exigences évolutives qui impliquent l'utilisation d'un jeu de caractères plus large tout en conservant la rétro compatibilité.

GSER code les valeurs de tous les types de chaîne ASN.1 comme des chaînes de caractères UTF-8 afin que la solution particulière choisie à partir d'un CHOIX purement syntaxique des types de chaînes ne fasse pas de différence matérielle pour le codage final de la valeur de chaîne.

Bien que certaines constructions ASN.1 trahissent la signification sémantique de la solution au sein d'un type de CHOIX , l'absence de ces constructions ne signifie pas nécessairement qu'un type de CHOIX soit purement syntaxique. Donc, il est nécessaire que les spécifications déclarent les types de CHOIX purement syntaxique afin qu'ils soient codés de façon plus compacte (voir au paragraphe 3.12). Ces types de CHOIX déclarés sont appelés des types ChoiceOfStrings *(choix de chaînes)*.

Pour pouvoir être déclaré un type ChoiceOfStrings, un type ASN.1 DOIT satisfaire aux conditions suivantes :

a) Le type est un CHOIX.
b) Le type composant de chaque solution de remplacement est un des types de chaîne ASN.1 restreinte suivants : NumericString, PrintableString, TeletexString (T61String), VideotexString, IA5String, GraphicString, VisibleString

(ISO646String), GeneralString, BMPString, UniversalString ou UTF8String.

c) Toutes les solutions de remplacement sont des types de chaîne restreinte différents, c'est-à-dire, aucune des solutions n'a le même type de chaîne ASN.1 restreinte.
d) Soit aucune des solutions n'a de contrainte, soit toutes les solutions ont exactement la même contrainte.

L'étiquetage de types de solution de remplacement est ignoré.

Définition du type de paramètre ASN.1 de DirectoryString *(chaîne de répertoire)* :

```
DirectoryString { ENTIER : maxSize } ::= CHOIX {
      teletexString     TeletexString (TAILLE (1..maxSize)),
      printableString   PrintableString (TAILLE (1..maxSize)),
      bmpString         BMPString (TAILLE (1..maxSize)),
      universalString   UniversalString (TAILLE (1..maxSize)),
      uTF8String        UTF8String (TAILLE (1..maxSize)) }
```

Toute utilisation du type DirectoryString paramétré avec un paramètre réel définit un type ASN.1 qui satisfait aux conditions ci-dessus. Reconnaissant que la solution de remplacement au sein d'une DirectoryString ne porte aucune signification sémantique, le présent document déclare (toute utilisation de) DirectoryString{} comme étant du type ChoiceOfStrings.

D'autres spécifications PEUVENT déclarer que d'autres types qui satisfont aux conditions ci-dessus sont du type ChoiceOfStrings. La déclaration DEVRAIT être faite à l'endroit où le type ASN.1 est défini ; autrement, elle DEVRAIT être faite à l'endroit où il est introduit comme, ou dans, un attribut LDAP ou une syntaxe d'assertion.

3.4 Identifiants

Un <identifiant> se conforme à la définition d'un identifiant dans la notation ASN.1 (paragraphe 11.3 de [X.680]). Il commence par une lettre minuscule et est suivi par zéro, une ou plusieurs lettres, chiffres, et traits d'union. Un trait d'union n'est pas permis au dernier caractère, et ne doit pas être suivi par un autre trait d'union. La casse des lettres dans un identifiant est toujours significative.

```
identifiant = minuscule *alphanumérique *(trait d'union 1*alphanumérique)
alphanumérique = majuscule / minuscule / chiffre-décimal
majuscule = %x41-5A                    ; "A" à "Z"
minuscule  = %x61-7A                   ; "a" à "z"
chiffre-décimal = %x30-39              ; "0"à "9"
trait d'union = "-"
```

3.5 BIT STRING

Une valeur du type BIT STRING *(CHAINE BINAIRE)* est codée conformément à la règle <BitStringValue>. Si la définition du type BIT STRING comporte une liste de bits nommée, la forme <bit-list> de <BitStringValue> PEUT être utilisée. Si le nombre de bits dans une valeur de BIT STRING est un multiple de quatre, la forme <hstring> de <BitStringValue> PEUT être utilisée. Autrement, la forme <bstring> de <BitStringValue> est utilisée.

```
BitStringValue = bstring / hstring / bit-list
```

La règle <bit-list> code les bits "un" de la valeur de la chaîne binaire comme une liste d'identifiants séparés par des virgules. Chaque <identifiant> DOIT être un des identifiants de la liste de bits nommée, et NE DOIT PAS apparaître plus d'une fois dans la même <bit-list>. La règle <bstring> code chaque bit comme le caractère "0" ou "1" dans l'ordre, du premier au dernier bit. La règle <hstring> code chaque groupe de quatre bits comme nombre hexadécimal où le premier bit est celui de poids fort. Un nombre impair de chiffres hexadécimaux est permis.

```
bit-list = "{" [ sp identifiant *( "," sp identifiant ) ] sp "}"
hstring = squote *chiffre-hexadécimal squote %x48          ; '...'H
chiffre-hexadécimal = %x30-39 /  ; "0" to "9" %x41-46      ; "A" à "F"
bstring = squote *chiffre-binaire squote %x42              ; '...'B
chiffre-binaire = "0" / "1"
sp = *%x20                                                 ; zéro, un ou plusieurs caractères espace
squote =  %x27                                             ; ' (guillemet simple)
```

3.6 BOOLEAN

Une valeur du type BOOLEAN *(BOOLÉEN)* est codée conformément à la règle <BooleanValue>.

```
BooleanValue = %x54.52.55.45 /                  ; "VRAI"
               %x46.41.4C.53.45                 ; "FAUX"
```

3.7 ENUMERATED

Une valeur du type ENUMERATED *(ÉNUMÉRÉ)* est codée conformément à la règle <EnumeratedValue> . Le <identifiant> DOIT être un de ceux de la liste des énumérations dans la définition du type ENUMERATED.

EnumeratedValue = identifiant

3.8 INTEGER

Une valeur du type INTEGER *(ENTIER)* est codée conformément à la règle <IntegerValue>. Si la définition du type INTEGER inclut une liste de nombres désignée, la forme <identifiant> de <IntegerValue> PEUT être utilisée, auquel cas le <identifiant> DOIT être un de ceux de la liste de nombres désignée.

IntegerValue = "0" / nombre-positif / ("-" nombre-positif) / identifiant

nombre-positif = chiffre-non-zero *chiffre-decimal
chiffre-non-zero = %x31-39 ; "1" à "9"

3.9 NULL

Une valeur du type NULL est codée conformément à la règle <NullValue> .

NullValue = %x4E.55.4C.4C ; "NUL"

3.10 OBJECT IDENTIFIER et RELATIVE-OID

Une valeur du type OBJECT IDENTIFIER *(IDENTIFIANT D'OBJET)* est codée conformément à la règle <ObjectIdentifierValue>. La règle <ObjectIdentifierValue> permet soit une représentation en décimal séparé par des points de la valeur de l'IDENTIFIANT D'OBJET, soit un nom de descripteur d'objet, c'est-à-dire, <descr>. La règle <descr> est décrite dans la [RFC2252]. Un nom de descripteur d'objet est potentiellement ambigu et devrait être utilisé avec précaution.

ObjectIdentifierValue = oid-numérique / descr
oid-numérique = composant-d'oid 1*("." composant-d'oid)
composant-d'oid = "0" / nombre-positif

Une valeur de RELATIVE-OID *(OID RELATIF)* est codée conformément à la règle <RelativeOIDValue>.

RelativeOIDValue = composant-d'oid *("." composant-d'oid)

3.11 OCTET STRING

Une valeur du type OCTET STRING est codée conformément à la règle <OctetStringValue>. Les octets sont codés dans l'ordre du premier octet au dernier octet. Chaque octet est codé comme une paire de chiffres hexadécimaux où le premier chiffre correspond aux quatre bits de plus fort poids de l'octet. Si la chaîne hexadécimale n'a pas un nombre pair de chiffres, les quatre bits de moindre poids du dernier octet sont supposés être à zéro.

OctetStringValue = hstring

3.12 CHOICE

Une valeur du type CHOICE *(CHOIX)* est codée conformément à la règle <ChoiceValue>. Le codage <ChoiceOfStringsValue> PEUT être utilisé si le type CHOIX correspondant a été déclaré comme un type ChoiceOfStrings. Le présent document déclare que DirectoryString est un type ChoiceOfStrings (voir au paragraphe 3.3). Autrement, la forme <IdentifiedChoiceValue> de <ChoiceValue> est utilisée.

ChoiceValue = IdentifiedChoiceValue / ChoiceOfStringsValue
IdentifiedChoiceValue = identifiant ":" Valeur
ChoiceOfStringsValue = StringValue

Pour les mises en œuvre qui reconnaissent la structure interne du type de CHOIX DirectoryString (par exemple, les répertoires [X.500]) si la chaîne de caractères entre les guillemets dans une <StringValue> contient seulement des caractères qui sont permis dans une PrintableString, la DirectoryString est supposée utiliser la solution printableString ; autrement, on suppose qu'elle utilise la solution uTF8String. La règle <IdentifiedChoiceValue> PEUT être utilisée pour une valeur de type DirectoryString pour indiquer une solution autre que celle qui aurait été supposée à partir du contenu de la chaîne. Quelle que soit la solution choisie, la <Valeur> va quand même être une chaîne de caractères codée en UTF-8. Cependant, c'est une erreur de syntaxe si les caractères dans la chaîne UTF-8 ne peuvent pas être représentés dans le type de chaîne de la solution choisie.

Les mises en œuvre qui ne se soucient pas de la structure interne d'une valeur de DirectoryString DOIVENT être capables d'analyser la forme <IdentifiedChoiceValue> pour une valeur de DirectoryString, bien que l'identifiant particulier trouvé ne soit d'aucun intérêt.

3.13 SEQUENCE et SET

Une valeur du type SEQUENCE est codée conformément à la règle <SequenceValue>. La règle <ComponentList> code une liste séparée par des virgules des valeurs de composant particulier présentes dans la valeur SEQUENCE, où chaque valeur de composant est précédée par l'identifiant correspondant provenant de la définition du type SEQUENCE. Les composants sont codés dans l'ordre de leur définition dans le type SEQUENCE.

SequenceValue = ComponentList

ComponentList = "{" [sp NamedValue *("," sp NamedValue)] sp "}"
NamedValue = identifiant msp Valeur
msp = 1*%x20 ; un ou plusieurs caractères espace.

Une valeur de SET *(ENSEMBLE)* est codée conformément à la règle <SetValue>. Les composants sont codés dans l'ordre de leur définition dans le type SET (c'est-à-dire, juste comme une valeur de SEQUENCE). Cela se distingue délibérément de la notation de valeur ASN.1 où les composants d'un SET peuvent être écrits dans un ordre quelconque.

SetValue = ComponentList

Les définitions de type de SEQUENCE et de SET sont parfois étendues par l'inclusion de types de composant supplémentaires, et donc une mise en œuvre DEVRAIT être capable de sauter tout codage de <NamedValue> avec un identifiant qui n'est pas reconnu, en supposant que l'envoyeur utilise une définition plus récente du type SEQUENCE ou SET.

3.14 SEQUENCE OF et SET OF

Une valeur du type SEQUENCE OF *(SEQUENCE DE)* est codée conformément à la règle <SequenceOfValue>, comme une liste séparée par des virgules des instances dans la valeur. Chaque instance est codée conformément au type de composant du type SEQUENCE OF.

SequenceOfValue = "{" [sp Valeur *("," sp Valeur)] sp "}"

Une valeur de SET OF *(ENSEMBLE DE)* est codée conformément à la règle <SetOfValue>, comme une liste des instances dans la valeur. Chaque instance est codée conformément au type de composant du type SET OF.

SetOfValue = "{" [sp Valeur *("," sp Valeur)] sp "}"

3.15 CHARACTER STRING

Une valeur du type CHARACTER STRING *(CHAINE DE CARACTERES)* sans restriction est codée conformément à la règle correspondante du type SEQUENCE définie au paragraphe 40.5 de [X.680] (voir la [RFC3642] pour l'ABNF équivalent).

CharacterStringValue = SequenceValue

3.16 EMBEDDED PDV

Une valeur du type EMBEDDED PDV *(valeurs de données de présentation encapsulées)* est codée conformément à la règle correspondante de type SEQUENCE définie au paragraphe 33.5 de [X.680] (voir dans la [RFC3642] l'ABNF équivalent).

EmbeddedPDVValue = SequenceValue

3.17 EXTERNAL

Une valeur du type EXTERNAL *(EXTERNE)* est codée conformément à la règle correspondante de type SEQUENCE définie au paragraphe 8.18.1 de [X.690] (voir dans la [RFC3642] l'ABNF équivalent).

ExternalValue = SequenceValue

3.18. INSTANCE OF

Une valeur du type INSTANCE OF *(INSTANCE DE)* est codée conformément à la règle correspondante de type SEQUENCE définie à l'Annexe C de [X.681].

InstanceOfValue = SequenceValue

3.19 REAL

Une valeur du type REAL DOIT être codée comme "0" si elle est zéro ; autrement, elle est codée comme valeur spéciale <PLUS-INFINITY>, <MINUS-INFINITY>, un <realnumber> *(nombre réel)* facultativement signé, ou une valeur du type SEQUENCE correspondant pour REAL défini au paragraphe 20.5 de [X.680] (voir dans la [RFC3642] l'ABNF équivalent).

```
RealValue  = "0"                        ; valeur zéro RÉELLE
           / PLUS-INFINITY              ; infini positif
           / MINUS-INFINITY             ; infini négatif
           / realnumber                 ; valeur positive RÉELLE en base 10
           / "-" realnumber             ; valeur négative RÉELLE en base 10
           / SequenceValue              ; valeur RÉELLE non zéro, en base 2 ou 10

realnumber = mantisse exposant
mantisse   = (nombre-positif [ "." *chiffre-décimal ]) / ( "0." *("0") nombre-positif )
exposant   = "E" ( "0" / ([ "-" ] nombre-positif))

PLUS-INFINITY  = %x50.4C.55.53.2D.49.4E.46.49.4E.49.54.59          ; "PLUS-INFINI"
MINUS-INFINITY = %x4D.49.4E.55.53.2D.49.4E.46.49.4E.49.54.59       ; "MOINS-INFINI"
```

3.20 Variantes de codage

Les valeurs de certains types complexes ASN.1 nommés ont un codage de chaîne spécial. Ces codages spéciaux sont toujours utilisés à la place du codage qui serait autrement appliqué sur la base de la définition du type ASN.1.

VariantEncoding = RDNSequenceValue / RelativeDistinguishedNameValue / ORAddressValue

Une valeur du type RDNSequence, c'est-à-dire, un nom distinctif, est codé conformément à la règle <RDNSequenceValue>, comme une chaîne de caractères LDAPDN entre guillemets. La chaîne de caractères est d'abord déduite conformément à la règle <distinguishedName> de la Section 3 de la [RFC2253], puis codée comme si c'était une valeur de UTF8String, c'est-à-dire, entre des doubles guillemets avec tous les doubles guillemets incorporés échappés en les répétant.

RDNSequenceValue = StringValue

Une valeur de RelativeDistinguishedName *(nom distinctif relatif)* qui ne fait pas partie d'une valeur de RDNSequence est codée conformément à la règle <RelativeDistinguishedNameValue> comme une chaîne de caractères entre guillemets. La chaîne de caractères est d'abord déduite conformément à la règle <name-component> de la Section 3 de la [RFC2253], puis codée comme si c'était une valeur de UTF8String.

RelativeDistinguishedNameValue = StringValue

Une valeur de ORAddress est codée conformément à la règle <ORAddressValue> comme une chaîne de caractères entre guillemets. La chaîne de caractères est d'abord déduite conformément à la représentation textuelle de MTS.ORAddress d'après la [RFC2156], puis codée comme si elle était une valeur IA5String.

ORAddressValue = StringValue

4. Syntaxe de transfert GSER

L'IDENTIFIANT D'OBJET suivant a été alloué par Adacel Technologies, sous un arc alloué à Adacel par Standards Australia, pour identifier les règles génériques de codage de chaînes :

{ 1 2 36 79672281 0 0 }

Cet IDENTIFIANT D'OBJET sera utilisé, par exemple, pour décrire la syntaxe de transfert pour une valeur de données codées en GSER dans une valeur EMBEDDED PDV.

5. Considérations sur la sécurité

Les règles générique de codage de chaîne ne définissent pas un codage canonique. C'est à dire qu'une transformation d'un codage GSER en un autre codage (par exemple, BER) et le retour en GSER ne va pas nécessairement reproduire le codage des octets GSER d'origine. Donc, GSER NE DOIT PAS être utilisé lorsque un codage canonique est nécessaire.

De plus, GSER ne permet pas nécessairement que le codage exact en octets des valeurs des types TeletexString, VideotexString, GraphicString ou GeneralString types soit reconstruit, de sorte qu'une transformation d'un codage selon les règles de codage distinctif (DER, *Distinguished Encoding Rules*) [X.690] en GSER et retour en DER peut ne pas reproduire le codage DER original. Donc, GSER NE DOIT PAS être utilisé pour recoder, que ce soit pour la mémorisation ou la transmission, des valeurs abstraites ASN.1 dont le codage binaire original doit être récupérable. Une telle récupération est nécessaire pour la vérification des signatures numériques. Dans de tels cas, les protocoles devraient utiliser le DER ou un codage réversible en DER.

Lors de l'interprétation de champs sensibles pour la sécurité, et en particulier les champs utilisés pour accorder ou refuser l'accès, les mises en œuvre DOIVENT s'assurer que toute comparaison est faite sur la valeur abstraite sous-jacente, sans considération du codage particulier utilisé.

6. Références

6.1 Références normatives

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

[RFC2156] S. Kille, "MIXER (Relais amélioré Mime Internet X.400) : transposition entre X.400 et la RFC0822/MIME ", janvier 1998. *(Remplace* RFC0987, RFC1026, RFC1138, RFC1148, RFC1327, RFC1495) *(MàJ* RFC0822) *(P.S.)*

[RFC2234] D. Crocker et P. Overell, "BNF augmenté pour les spécifications de syntaxe : ABNF", novembre 1997. (*Obsolète, voir* RFC5234)

[RFC2252] M. Wahl, A. Coulbeck, T. Howes, S. Kille, "Protocole léger d'accès à un répertoire (v3) : Définitions de syntaxe d'attribut", décembre 1997. (*Obsolète, voir* RFC4510, RFC4517, RFC4523, RFC4512) (*MàJ par* RFC3377) *(P.S.)*

[RFC2253] M. Wahl, S. Kille et T. Howes, "Protocole léger d'accès à un répertoire (LDAPv3) : Représentation de chaîne UTF-8 des noms distinctifs", décembre 1997.

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

[X.520] Recommendation UIT-T X.520 (1993) | ISO/CEI 9594-6:1994, "Technologies de l'information - Interconnexion des systèmes ouverts – L'Annuaire : Types d'attributs choisis".

[X.680] Recommendation UIT-T X.680 (07/02) | ISO/CEI 8824-1:2002 "Technologies de l'information – Notation de syntaxe abstraite numéro un (ASN.1) : Spécification de la notation de base".

[X.681] Recommendation UIT-T X.681 (07/02) | ISO/CEI 8824-2:2002"Technologies de l'information - Notation de syntaxe abstraite numéro un (ASN.1) : Spécification des objets d'information".

[X.682] Recommendation UIT-T X.682 (07/02) | ISO/CEI 8824-3:2002 "Technologies de l'information - Notation de syntaxe abstraite numéro un (ASN.1) : Spécification des contraintes".

[X.683] Recommendation UIT-T X.683 (07/02) | ISO/CEI 8824-4:2002 "Technologies de l'information - Notation de syntaxe abstraite numéro un (ASN.1): Paramétérisation des spécifications ASN.1".

[X.690] Recommendation UIT-T X.690 (07/02) | ISO/CEI 8825-1:2002 "Technologies de l'information – Règles de codage ASN.1 : Spécification des règles de codage de base (BER), des règles de codage canonique (CER) et des règles de codage distinctif (DER)

6.2 Références Informatives

[RFC2028] R. Hovey, S. Bradner, "Les organisations impliquées dans le processus de normalisation de l'IETF", octobre 1996. (*MàJ par* RFC3668, RFC3979) (BCP0011)

[RFC3377] J. Hodges, R. Morgan, "Protocole léger d'accès à un répertoire (v3) : Spécification technique", septembre 2002. *Obsolète, voir* RFC4510) *(P.S.)*

[RFC3642] S. Legg, "Éléments communs des règles génériques de codage de chaînes (GSER)", octobre 2003.

[X.500] Recommendation UIT-T X.500 (1993) | ISO/CEI 9594-1:1994, "Technologies de l'information - Interconnexion des systèmes ouverts – L'Annuaire : Survol des concepts, modèles et services".

7. Notice de droits de propriété intellectuelle

L'IETF ne prend pas position sur la validité et la portée de tout droit de propriété intellectuelle ou autres droits qui pourrait être revendiqués au titre de la mise en œuvre ou l'utilisation de la technologie décrite dans le présent document ou sur la mesure dans laquelle toute licence sur de tels droits pourrait être ou n'être pas disponible ; pas plus qu'elle ne prétend avoir accompli aucun effort pour identifier de tels droits. Les informations sur les procédures de l'ISOC au sujet des droits dans les documents de l'ISOC figurent dans les BCP 78 et BCP 79.

Des copies des dépôts d'IPR faites au secrétariat de l'IETF et toutes assurances de disponibilité de licences, ou le résultat de tentatives faites pour obtenir une licence ou permission générale d'utilisation de tels droits de propriété par ceux qui mettent en œuvre ou utilisent la présente spécification peuvent être obtenues sur répertoire en ligne des IPR de l'IETF à http://www.ietf.org/ipr.

L'IETF invite toute partie intéressée à porter son attention sur tous copyrights, licences ou applications de licence, ou autres droits de propriété qui pourraient couvrir les technologies qui peuvent être nécessaires pour mettre en œuvre la présente norme. Prière d'adresser les informations à l'IETF à ietf-ipr@ietf.org.

8. Adresse de l'auteur

Steven Legg
Adacel Technologies Ltd.
250 Bay Street
Brighton, Victoria 3186
AUSTRALIA

téléphone : +61 3 8530 7710
Fax: +61 3 8530 7888
mél : steven.legg@adacel.com.au

9. Déclaration complète de droits de reproduction

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