Network Working Group
Request for Comments: 3642
Category: Informational

S. Legg
Adacel Technologies

October 2003

Common Elements of Generic String Encoding Rules (GSER) Encodings

Status of this Memo

Copyright Notice

Abstract

The Generic String Encoding Rules (GSER) describe a human readable text encoding for an Abstract Syntax Notation One (ASN.1) value of any ASN.1 type. Specifications making use of GSER may wish to provide an equivalent Augmented Backus-Naur Form (ABNF) description of the GSER encoding for a particular ASN.1 type as a convenience for implementors. This document supports such specifications by providing equivalent ABNF for the GSER encodings for ASN.1 types that commonly occur in Lightweight Directory Access Protocol (LDAP) syntaxes.

Table of Contents

1. Introduction

The Generic String Encoding Rules (GSER) [7] define a human readable text encoding, based on ASN.1 [8] value notation, for an ASN.1 value of any ASN.1 type. Specifications making use of GSER may wish to provide a non-normative equivalent ABNF [3] description of the GSER encoding for a particular ASN.1 type as a convenience for implementors unfamiliar with ASN.1. This document supports such specifications by providing equivalent ABNF for the GSER encodings for ASN.1 types that commonly occur in LDAP [10] or X.500 [11] attribute and assertion syntaxes, as well as equivalent ABNF for the GSER encodings for the ASN.1 built-in types.

The ABNF given in this document does not replace or alter GSER in any way. If there is a discrepancy between the ABNF specified here and the encoding defined by GSER [7], then GSER is to be taken as definitive.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", and "MAY" in this document are to be interpreted as described in BCP 14, RFC 2119 [1]. The key word "OPTIONAL" is exclusively used with its ASN.1 meaning.

3. Separators

Certain separators are commonly used in constructing equivalent ABNF for SET and SEQUENCE types.

```
sp  =  *%x20  ; zero, one or more space characters
msp = 1*%x20  ; one or more space characters

sep = [ "," ]
```

The <sep> rule is used in the ABNF description of the encoding for ASN.1 SET or SEQUENCE types where all the components are either OPTIONAL or DEFAULT. It encodes to an empty string if and only if the immediately preceding character in the encoding is "{", i.e., it is only empty for the first optional component actually present in the SET or SEQUENCE value being encoded.

4. ASN.1 Built-in Types

This section describes the GSER encoding of values of the ASN.1 built-in types, except for the restricted character string types.

The <BIT-STRING> rule describes the GSER encoding of values of the BIT STRING type without a named bit list.

```
BIT-STRING = bstring / hstring
```

If the number of bits in a BIT STRING value is a multiple of four the <hstring> form of <BIT-STRING> MAY be used. Otherwise, the <bstring> form of <BIT-STRING> is used. The <bstring> rule encodes each bit as the character "0" or "1" in order from the first bit to the last bit. The <hstring> rule encodes each group of four bits as a hexadecimal number where the first bit is the most significant. An odd number of hexadecimal digits is permitted.

```
hstring           = squote *hexadecimal-digit squote %x48 ; '...'H
hexadecimal-digit = %x30-39 /  ; "0" to "9"
                    %x41-46    ; "A" to "F"

bstring           = squote *binary-digit squote %x42  ; '...'B
binary-digit      = "0" / "1"

squote            =  %x27  ; ' (single quote)
```

The <BOOLEAN> rule describes the GSER encoding of values of the BOOLEAN type.

```
BOOLEAN = %x54.52.55.45 /   ; "TRUE"
          %x46.41.4C.53.45  ; "FALSE"
```

The <CHARACTER-STRING> rule describes the GSER encoding of values of the associated type for the unrestricted CHARACTER STRING type.

```
CHARACTER-STRING = "{" sp id-identification msp Identification ","
                       sp id-data-value msp OCTET-STRING
                       sp "}"

id-identification = %x69.64.65.6E.74.69.66.69.63.61.74.69.6F.6E
                       ; "identification"
id-data-value     = %x64.61.74.61.2D.76.61.6C.75.65 ; "data-value"

Identification = ( id-syntaxes ":" Syntaxes ) /
                 ( id-syntax ":" OBJECT-IDENTIFIER ) /
                 ( id-presentation-context-id ":" INTEGER ) /
                 ( id-context-negotiation ":"
                     ContextNegotiation ) /
                 ( id-transfer-syntax ":" OBJECT-IDENTIFIER ) /
                 ( id-fixed ":" NULL )
```

```
   id-syntaxes                = %x73.79.6E.74.61.78.65.73
                                    ; "syntaxes"
   id-syntax                  = %x73.79.6E.74.61.78 ; "syntax"
   id-presentation-context-id = %x70.72.65.73.65.6E.74.61.74.69.6F.6E
                                   %x2D.63.6F.6E.74.65.78.74.2D.69.64
                                   ; "presentation-context-id"
   id-context-negotiation     = %x63.6F.6E.74.65.78.74.2D.6E.65.67.6F
                                   %x74.69.61.74.69.6F.6E
                                   ; "context-negotiation"
   id-transfer-syntax         = %x74.72.61.6E.73.66.65.72.2D.73.79.6E
                                   %x74.61.78 ; "transfer-syntax"
   id-fixed                   = %x66.69.78.65.64 ; "fixed"

   Syntaxes = "{" sp id-abstract msp OBJECT-IDENTIFIER ","
                  sp id-transfer msp OBJECT-IDENTIFIER
                  sp "}"
   id-abstract = %x61.62.73.74.72.61.63.74 ; "abstract"
   id-transfer = %x74.72.61.6E.73.66.65.72 ; "transfer"

   ContextNegotiation = "{" sp id-presentation-context-id msp
                                  INTEGER ","
                            sp id-transfer-syntax msp
                                  OBJECT-IDENTIFIER
                            sp "}"
```

The <INTEGER> rule describes the GSER encoding of values of the INTEGER type without a named number list. The <INTEGER-0-MAX> rule describes the GSER encoding of values of the constrained type INTEGER (0..MAX). The <INTEGER-1-MAX> rule describes the GSER encoding of values of the constrained type INTEGER (1..MAX).

```
   INTEGER         = "0" / positive-number / ("-" positive-number)
   INTEGER-0-MAX   = "0" / positive-number
   INTEGER-1-MAX   = positive-number
   positive-number = non-zero-digit *decimal-digit
   decimal-digit   = %x30-39  ; "0" to "9"
   non-zero-digit  = %x31-39  ; "1" to "9"
```

The <EMBEDDED-PDV> rule describes the GSER encoding of values of the associated type for the EMBEDDED PDV type.

```
   EMBEDDED-PDV = "{" sp id-identification msp Identification ","
                      sp id-data-value msp OCTET-STRING
                      sp "}"
```

The <EXTERNAL> rule describes the GSER encoding of values of the associated type for the EXTERNAL type.

```
EXTERNAL = "{" [ sp id-direct-reference msp
                    OBJECT-IDENTIFIER "," ]
                 [ sp id-indirect-reference msp INTEGER "," ]
                 [ sp id-data-value-descriptor msp
                    ObjectDescriptor "," ]
                   sp id-encoding msp Encoding
                   sp "}"

id-direct-reference      = %x64.69.72.65.63.74.2D.72.65.66.65.72
                           %x65.6E.63.65
                           ; "direct-reference"
id-indirect-reference    = %x69.6E.64.69.72.65.63.74.2D.72.65.66
                           %x65.72.65.6E.63.65
                           ; "indirect-reference"
id-data-value-descriptor = %x64.61.74.61.2D.76.61.6C.75.65.2D.64
                           %x65.73.63.72.69.70.74.6F.72
                           ; "data-value-descriptor"
id-encoding              = %x65.6E.63.6F.64.69.6E.67
                           ; "encoding"

Encoding = ( id-single-ASN1-type ":" Value ) /
           ( id-octet-aligned ":" OCTET-STRING ) /
           ( id-arbitrary ":" BIT-STRING )

id-single-ASN1-type = %x73.69.6E.67.6C.65.2D.41.53.4E.31.2D.74.79
                      %x70.65
                      ; "single-ASN1-type"
id-octet-aligned    = %x6F.63.74.65.74.2D.61.6C.69.67.6E.65.64
                      ; "octet-aligned"

id-arbitrary        = %x61.72.62.69.74.72.61.72.79
                      ; "arbitrary"
```

The <Value> rule is defined by GSER [7]. It represents the GSER encoding of a single value of the ASN.1 type identified by the direct-reference and/or indirect-reference components.

The <NULL> rule describes the GSER encoding of values of the NULL type.

```
NULL = %x4E.55.4C.4C  ; "NULL"
```

The <OBJECT-IDENTIFIER> rule describes the GSER encoding of values of the OBJECT IDENTIFIER type.

```
OBJECT-IDENTIFIER = numeric-oid / descr
numeric-oid       = oid-component 1*( "." oid-component )
oid-component     = "0" / positive-number
```

An OBJECT IDENTIFIER value is encoded using either the dotted decimal representation or an object descriptor name, i.e., <descr>. The <descr> rule is described in RFC 2252 [4]. An object descriptor name is potentially ambiguous and should be used with care.

The <OCTET-STRING> rule describes the GSER encoding of values of the OCTET STRING type.

```
OCTET-STRING = hstring
```

The octets are encoded in order from the first octet to the last octet. Each octet is encoded as a pair of hexadecimal digits where the first digit corresponds to the four most significant bits of the octet. If the hexadecimal string does not have an even number of digits, the four least significant bits in the last octet are assumed to be zero.

The <REAL> rule describes the GSER encoding of values of the REAL type.

```
REAL = "0"                    ; zero
       / PLUS-INFINITY        ; positive infinity
       / MINUS-INFINITY       ; negative infinity
       / realnumber           ; positive base 10 REAL value
       / ( "-" realnumber )   ; negative base 10 REAL value
       / real-sequence-value  ; non-zero base 2 or 10 REAL value

PLUS-INFINITY  = %x50.4C.55.53.2D.49.4E.46.49.4E.49.54.59
                     ; "PLUS-INFINITY"

MINUS-INFINITY = %x4D.49.4E.55.53.2D.49.4E.46.49.4E.49.54.59
                     ; "MINUS-INFINITY"

realnumber = mantissa exponent
mantissa   = (positive-number [ "." *decimal-digit ])
             / ( "0." *("0") positive-number )
exponent   = "E" ( "0" / ([ "-" ] positive-number))

real-sequence-value = "{" sp id-mantissa msp INTEGER ","
                          sp id-base msp ( "2" / "10" ) ","
                          sp id-exponent msp INTEGER sp "}"
id-mantissa         = %x6D.61.6E.74.69.73.73.61 ; "mantissa"
id-base             = %x62.61.73.65             ; "base"
id-exponent         = %x65.78.70.6F.6E.65.6E.74 ; "exponent"
```

A value of the REAL type MUST be encoded as "0" if it is zero.

The <RELATIVE-OID> rule describes the GSER encoding of values of the RELATIVE-OID type.

```
RELATIVE-OID = oid-component *( "." oid-component )
```

5. ASN.1 Restricted String Types

This section describes the GSER encoding of values of the ASN.1 restricted character string types. The characters of a value of a restricted character string type are always encoded as a UTF-8 character string between double quotes. For some of the ASN.1 string types, this requires a translation to or from the UTF-8 encoding. Some of the ASN.1 string types permit only a subset of the characters representable in UTF-8. Any double quote characters in the character string, where allowed by the character set, are escaped by being repeated.

The <UTF8String> rule describes the GSER encoding of values of the UTF8String type. The characters of this string type do not require any translation before being encoded.

```
UTF8String        = StringValue
StringValue       = dquote *SafeUTF8Character dquote

dquote            = %x22 ; " (double quote)

SafeUTF8Character = %x00-21 / %x23-7F /   ; ASCII minus dquote
                    dquote dquote /       ; escaped double quote
                    %xC0-DF %x80-BF /     ; 2 byte UTF-8 character
                    %xE0-EF 2(%x80-BF) /  ; 3 byte UTF-8 character
                    %xF0-F7 3(%x80-BF)    ; 4 byte UTF-8 character
```

The <NumericString>, <PrintableString>, <VisibleString>, <ISO646String>, <IA5String>, <GeneralizedTime> and <UTCTime> rules describe the GSER encoding of values of the correspondingly named ASN.1 types. The characters of these string types are compatible with UTF-8 and do not require any translation before being encoded. The GeneralizedTime and UTCTime types use the VisibleString character set, but have a strictly defined format.

```
NumericString       = dquote *(decimal-digit / space) dquote
space               = %x20
```

```
PrintableString      = dquote *PrintableCharacter dquote
PrintableCharacter   = decimal-digit / space
                       / %x41-5A ; A to Z
                       / %x61-7A ; a to z
                       / %x27-29 ; ' ( )
                       / %x2B-2F ; + , - . /
                       / %x3A    ; :
                       / %x3D    ; =
                       / %x3F    ; ?

ISO646String         = VisibleString
VisibleString        = dquote *SafeVisibleCharacter dquote
SafeVisibleCharacter = %x20-21
                       / %x23-7E ; printable ASCII minus dquote
                       / dquote dquote   ; escaped double quote

IA5String            = dquote *SafeIA5Character dquote
SafeIA5Character     = %x00-21 / %x23-7F ; ASCII minus dquote
                       / dquote dquote   ; escaped double quote

century = 2(%x30-39) ; "00" to "99"
year    = 2(%x30-39) ; "00" to "99"
month   =   ( %x30 %x31-39 ) ; "01" (January) to "09"
          / ( %x31 %x30-32 ) ; "10" to "12"
day     =   ( %x30 %x31-39 )    ; "01" to "09"
          / ( %x31-32 %x30-39 ) ; "10" to "29"
          / ( %x32 %x30-31 )    ; "30" to "31"
hour    = ( %x30-31 %x30-39 ) / ( %x32 %x30-33 ) ; "00" to "23"
minute  = %x30-35 %x30-39                        ; "00" to "59"
second  =   ( %x30-35 %x30-39 )  ; "00" to "59"
          / ( %x36 %x30 )        ; "60" (a leap second)

UTCTime         = dquote year month day hour minute [ second ]
                      [ %x5A / u-differential ] dquote
u-differential  = ( "-" / "+" ) hour minute

GeneralizedTime = dquote century year month day hour
                      [ minute [ second ] ] [ fraction ]
                      [ %x5A / g-differential ] dquote
fraction        = ( "." / "," ) 1*(%x30-39)
g-differential  = ( "-" / "+" ) hour [ minute ]
```

The <BMPString> and <UniversalString> rules describe the GSER encoding of values of the BMPString and UniversalString types respectively. BMPString (UCS-2) and UniversalString (UCS-4) values are translated into UTF-8 [6] character strings before being encoded according to <StringValue>.

```
   BMPString       = StringValue
   UniversalString = StringValue
```

The <TeletexString>, <T61String>, <VideotexString>, <GraphicString>, <GeneralString> and <ObjectDescriptor> rules describe the GSER encoding of values of the correspondingly named ASN.1 types. Values of these string types are translated into UTF-8 character strings before being encoded according to <StringValue>. The ObjectDescriptor type uses the GraphicString character set.

```
   TeletexString    = StringValue
   T61String        = StringValue
   VideotexString   = StringValue
   GraphicString    = StringValue
   GeneralString    = StringValue
   ObjectDescriptor = GraphicString
```

6. Directory ASN.1 Types

This section describes the GSER encoding of values of selected ASN.1 types defined for LDAP and X.500. The ABNF rule names beginning with uppercase letters describe the GSER encoding of values of the ASN.1 type with the same name.

```
   AttributeType  = OBJECT-IDENTIFIER
```

The characters of a DirectoryString are translated into UTF-8 characters as required before being encoded between double quotes with any embedded double quotes escaped by being repeated.

```
   DirectoryString = StringValue /
                     ( id-teletexString   ":" TeletexString ) /
                     ( id-printableString ":" PrintableString ) /
                     ( id-bmpString       ":" BMPString ) /
                     ( id-universalString ":" UniversalString ) /
                     ( id-uTF8String      ":" UTF8String )

   id-teletexString   = %x74.65.6C.65.74.65.78.53.74.72.69.6E.67
                            ; "teletexString"
   id-printableString = %x70.72.69.6E.74.61.62.6C.65
                            %x53.74.72.69.6E.67 ; "printableString"
   id-bmpString       = %x62.6D.70.53.74.72.69.6E.67 ; "bmpString"
   id-universalString = %x75.6E.69.76.65.72.73.61.6C
                            %x53.74.72.69.6E.67 ; "universalString"
   id-uTF8String      = %x75.54.46.38.53.74.72.69.6E.67
                               ; "uTF8String"
```

The <RDNSequence> rule describes the GSER encoding of values of the RDNSequence type, which is syntactically equivalent to the DistinguishedName and LocalName types. The <RDNSequence> rule encodes a name as an LDAPDN character string between double quotes. The character string is first derived according to the <distinguishedName> rule in Section 3 of RFC 2253 [5], and then it is encoded between double quotes with any embedded double quotes escaped by being repeated.

```
DistinguishedName = RDNSequence
LocalName         = RDNSequence
RDNSequence       = dquote *SafeUTF8Character dquote
```

The <RelativeDistinguishedName> rule describes the GSER encoding of values of the RelativeDistinguishedName type that are not part of an RDNSequence value. The <RelativeDistinguishedName> rule encodes an RDN as a double quoted string containing the RDN as it would appear in an LDAPDN character string. The character string is first derived according to the <name-component> rule in Section 3 of RFC 2253 [5], and then any embedded double quote characters are escaped by being repeated. This resulting string is output between double quotes.

```
RelativeDistinguishedName = dquote *SafeUTF8Character dquote
```

The <ORAddress> rule encodes an X.400 address as an IA5 character string between double quotes. The character string is first derived according to Section 4.1 of RFC 2156 [2], and then any embedded double quotes are escaped by being repeated. This resulting string is output between double quotes.

```
ORAddress = dquote *SafeIA5Character dquote
```

7. Security Considerations

This document contains an alternative description of parts of the Generic String Encoding Rules, but does not replace or alter GSER in any way. For the full security implications of using GSER, see the Security Considerations section for GSER [7].

8. References

8.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Kille, S., "MIXER (Mime Internet X.400 Enhanced Relay): Mapping between X.400 and RFC 822/MIME", RFC 2156, January 1998.

[3] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[4] Wahl, M., Coulbeck, A., Howes, T. and S. Kille, "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions", RFC 2252, December 1997.

[5] Wahl, M., Kille, S. and T. Howes, "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names", RFC 2253, December 1997.

[6] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

[7] Legg, S., "Generic String Encoding Rules (GSER) for ASN.1 Types", RFC 3641, October 2003.

[8] ITU-T Recommendation X.680 (07/02) | ISO/IEC 8824-1:2002 Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation

8.2. Informative References

[9] Hovey, R. and S. Bradner, "The Organizations Involved in the IETF Standards Process", BCP 11, RFC 2028, October 1996.

[10] Hodges, J. and R. Morgan, "Lightweight Directory Access Protocol (v3): Technical Specification", RFC 3377, September 2002.

[11] ITU-T Recommendation X.500 (1993) | ISO/IEC 9594-1:1994, Information Technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services

9. Intellectual Property Notice

10. Author's Address

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

Phone: +61 3 8530 7710
Fax: +61 3 8530 7888
EMail: steven.legg@adacel.com.au

11. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.