          The String Representation of Standard Attribute Syntaxes

Status of this Memo

Abstract

   The Lightweight Directory Access Protocol (LDAP) [9] requires that
   the contents of AttributeValue fields in protocol elements be octet
   strings. This document defines the requirements that must be
   satisfied by encoding rules used to render X.500 Directory attribute
   syntaxes into a form suitable for use in the LDAP, then goes on to
   define the encoding rules for the standard set of attribute syntaxes
   defined in [1,2] and [3].

1.   Attribute Syntax Encoding Requirements.

   This section defines general requirements for lightweight directory
   protocol attribute syntax encodings. All documents defining attribute
   syntax encodings for use by the lightweight directory protocols are
   expected to conform to these requirements.

   The encoding rules defined for a given attribute syntax must produce
   octet strings. To the greatest extent possible, encoded octet
   strings should be usable in their native encoded form for display
   purposes. In particular, encoding rules for attribute syntaxes
   defining non-binary values should produce strings that can be
   displayed with little or no translation by clients implementing the
   lightweight directory protocols.

2. Standard Attribute Syntax Encodings

   For the purposes of defining the encoding rules for the standard
   attribute syntaxes, the following auxiliary BNF definitions will be
   used:

```
<a> ::= 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' |
        'j' | 'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' |
        's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y' | 'z' | 'A' |
        'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' |
        'K' | 'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' |
        'T' | 'U' | 'V' | 'W' | 'X' | 'Y' | 'Z'

<d> ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

<hex-digit> ::= <d> | 'a' | 'b' | 'c' | 'd' | 'e' | 'f' |
                'A' | 'B' | 'C' | 'D' | 'E' | 'F'

<k> ::= <a> | <d> | '-'

<p> ::= <a> | <d> | ''' | '(' | ')' | '+' | ',' | '-' | '.' |
        '/' | ':' | '?' | ' '

<CRLF> ::= The ASCII newline character with hexadecimal value 0x0A

<letterstring> ::= <a> | <a> <letterstring>

<numericstring> ::= <d> | <d> <numericstring>

<keystring> ::= <a> | <a> <anhstring>

<anhstring> ::= <k> | <k> <anhstring>

<printablestring> ::= <p> | <p> <printablestring>

<space> ::= ' ' | ' ' <space>
```

2.1. Undefined

   Values of type Undefined are encoded as if they were values of type
   Octet String, with the string value being the BER-encoded version of
   the value.

2.2. Case Ignore String

   A string of type caseIgnoreStringSyntax is encoded as the string
   value itself.

2.3. Case Exact String

   The encoding of a string of type caseExactStringSyntax is the string
   value itself.

2.4. Printable String

   The encoding of a string of type printableStringSyntax is the string
   value itself.

2.5. Numeric String

   The encoding of a string of type numericStringSyntax is the string
   value itself.

2.6. Octet String

   The encoding of a string of type octetStringSyntax is the string
   value itself.

2.7. Case Ignore IA5 String

   The encoding of a string of type caseIgnoreIA5String is the string
   value itself.

2.8. IA5 String

   The encoding of a string of type iA5StringSyntax is the string value
   itself.

2.9. T61 String

   The encoding of a string of type t61StringSyntax is the string value
   itself.

2.10. Case Ignore List

   Values of type caseIgnoreListSyntax are encoded according to the
   following BNF:

<caseignorelist> ::= <caseignorestring> |
                     <caseignorestring> '$' <caseignorelist>

<caseignorestring> ::= a string encoded according to the rules for Case
                     Ignore String as above.

2.11. Case Exact List

   Values of type caseExactListSyntax are encoded according to the
   following BNF:

<caseexactlist> ::= <caseexactstring> |
                    <caseexactstring> '$' <caseexactlist>

<caseexactstring> ::= a string encoded according to the rules for Case
                    Exact String as above.

2.12. Distinguished Name

   Values of type distinguishedNameSyntax are encoded to have the
   representation defined in [5].

2.13. Boolean

   Values of type booleanSyntax are encoded according to the following
   BNF:

      <boolean> ::= "TRUE" | "FALSE"

   Boolean values have an encoding of "TRUE" if they are logically true,
   and have an encoding of "FALSE" otherwise.

2.14. Integer

   Values of type integerSyntax are encoded as the decimal
   representation of their values, with each decimal digit represented
   by the its character equivalent. So the digit 1 is represented by the
   character

2.15. Object Identifier

   Values of type objectIdentifierSyntax are encoded according to the
   following BNF:

      <oid> ::= <descr> | <descr> '.' <numericoid> | <numericoid>

      <descr> ::= <keystring>

      <numericoid> ::= <numericstring> | <numericstring> '.' <numericoid>

   In the above BNF, <descr> is the syntactic representation of an
   object descriptor. When encoding values of type
   objectIdentifierSyntax, the first encoding option should be used in
   preference to the second, which should be used in preference to the

third wherever possible. That is, in encoding object identifiers,
object descriptors (where assigned and known by the implementation)
should be used in preference to numeric oids to the greatest extent
possible. For example, in encoding the object identifier representing
an organizationName, the descriptor "organizationName" is preferable
to "ds.4.10", which is in turn preferable to the string "2.5.4.10".

2.16. Telephone Number

   Values of type telephoneNumberSyntax are encoded as if they were
   Printable String types.

2.17. Telex Number

   Values of type telexNumberSyntax are encoded according to the
   following BNF:

      <telex-number> ::= <actual-number> '$' <country> '$' <answerback>

      <actual-number> ::= <printablestring>

      <country> ::= <printablestring>

      <answerback> ::= <printablestring>

   In the above, <actual-number> is the syntactic representation of the
   number portion of the TELEX number being encoded, <country> is the
   TELEX country code, and <answerback> is the answerback code of a
   TELEX terminal.

2.18. Teletex Terminal Identifier

   Values of type teletexTerminalIdentifier are encoded according to the
   following BNF:

      <teletex-id> ::= <printablestring> 0*('$' <ttx-parm>)

      <ttx-param> ::= <ttx-key> ':' <ttx-value>

      <ttx-key> ::= 'graphic' | 'control' | 'misc' | 'page' | 'private'

      <ttx-value> ::= <octetstring>

   In the above, the first <printablestring> is the encoding of the
   first portion of the teletex terminal identifier to be encoded, and
   the subsequent 0 or more <printablestrings> are subsequent portions
   of the teletex terminal identifier.

2.19. Facsimile Telephone Number

   Values of type FacsimileTelephoneNumber are encoded according to the
   following BNF:

<fax-number> ::= <printablestring> ['$' <faxparameters>]

<faxparameters> ::= <faxparm> | <faxparm> '$' <faxparameters>

<faxparm> ::= 'twoDimensional' | 'fineResolution' | 'unlimitedLength' |
              'b4Length' | 'a3Width' | 'b4Width' | 'uncompressed'

   In the above, the first <printablestring> is the actual fax number,
   and the <faxparm> tokens represent fax parameters.

2.20. Presentation Address

   Values of type PresentationAddress are encoded to have the
   representation described in [6].

2.21. UTC Time

   Values of type uTCTimeSyntax are encoded as if they were Printable
   Strings with the strings containing a UTCTime value.

2.22. Guide (search guide)

   Values of type Guide, such as values of the searchGuide attribute,
   are encoded according to the following BNF:

<guide-value> ::= [<object-class> '#'] <criteria>

<object-class> ::= an encoded value of type objectIdentifierSyntax

<criteria> ::= <criteria-item> | <criteria-set> | '!' <criteria>

<criteria-set> ::= ['('] <criteria> '&' <criteria-set> [')'] |
                   ['('] <criteria> '|' <criteria-set> [')']

<criteria-item> ::= ['('] <attributetype> '$' <match-type> [')']

<match-type> ::= "EQ" | "SUBSTR" | "GE" | "LE" | "APPROX"

2.23. Postal Address

   Values of type PostalAddress are encoded according to the following
   BNF:

      <postal-address> ::= <t61string> | <t61string> '$' <postal-address>

   In the above, each <t61string> component of a postal address value is
   encoded as a value of type t61StringSyntax.

2.24. User Password

   Values of type userPasswordSyntax are encoded as if they were of type
   octetStringSyntax.

2.25. User Certificate

   Values of type userCertificate are encoded according to the following
   BNF:

      <certificate> ::= <version> '#' <serial> '#' <signature-algorithm-id>
                        '#' <issuer> '#' <validity> '#' <subject>
                        '#' <public-key-info> '#' <encrypted-sign-value>

      <version> ::= <integervalue>

      <serial> ::= <integervalue>

      <signature-algorithm-id> ::= <algorithm-id>

      <issuer> ::= an encoded Distinguished Name

      <validity> ::= <not-before-time> '#' <not-after-time>

      <not-before-time> ::= <utc-time>

      <not-after-time> ::= <utc-time>

      <algorithm-parameters> ::= <null> | <integervalue> |
                                  '{ASN}' <hex-string>

      <subject> ::= an encoded Distinguished Name

      <public-key-info> ::= <algorithm-id> '#' <encrypted-sign-value>

      <encrypted-sign-value> ::= <hex-string> | <hex-string> '-' <d>

      <algorithm-id> ::= <oid> '#' <algorithm-parameters>

```
<utc-time> ::= an encoded UTCTime value

<hex-string> ::= <hex-digit> | <hex-digit> <hex-string>
```

2.26. CA Certificate

   Values of type cACertificate are encoded as if the values were of
   type userCertificate.

2.27. Authority Revocation List

   Values of type authorityRevocationList are encoded according to the
   following BNF:

```
<certificate-list> ::= <signature-algorithm-id> '#' <issuer> '#' <utc-time>
                       [ '#' <revoked-certificates> ]
                       '#' <signature-algorithm-id>
                       '#' <encrypted-sign-value>

<revoked-certificates> ::= 1*( '#' <revoked-certificate> )
                       <signature-algorithm-id> '#' <encrypted-sign-value>

<revoked-certificate> ::= <signature-algorithm-id> '#' <issuer> '#'
                       <serial> '#' <utc-time>
```

   The syntactic components <signature-algorithm-id>, <issuer>,
   <encrypted-sign-value>, <utc-time>, <subject> and <serial> have the
   same definitions as in the BNF for the userCertificate attribute
   syntax.

2.28. Certificate Revocation List

   Values of type certificateRevocationList are encoded as if the values
   were of type authorityRevocationList.

2.29. Cross Certificate Pair

   Values of type crossCertificatePair are encoded according to the
   following BNF:

```
   <certificate-pair> ::= <forward> '#' <reverse>
                             | <forward>
                             | <reverse>

   <forward> ::= 'forward:' <certificate>

   <reverse> ::= 'reverse:' <certificate>
```

   The syntactic component <certificate> has the same definition as in
   the BNF for the userCertificate attribute syntax.

2.30. Delivery Method

   Values of type deliveryMethod are encoded according to the following
   BNF:

      <delivery-value> ::= <pdm> | <pdm> '$' <delivery-value>

      <pdm> ::= 'any' | 'mhs' | 'physical' | 'telex' | 'teletex' |
                'g3fax' | 'g4fax' | 'ia5' | 'videotex' | 'telephone'

2.31. Other Mailbox

   Values of the type otherMailboxSyntax are encoded according to the
   following BNF:

      <otherMailbox> ::= <mailbox-type> '$' <mailbox>

      <mailbox-type> ::= an encoded Printable String

      <mailbox> ::= an encoded IA5 String

   In the above, <mailbox-type> represents the type of mail system in
   which the mailbox resides, for example "Internet" or "MCIMail"; and
   <mailbox> is the actual mailbox in the mail system defined by
   <mailbox-type>.

2.32. Mail Preference

   Values of type mailPreferenceOption are encoded according to the
   following BNF:

      <mail-preference> ::= "NO-LISTS" | "ANY-LIST" | "PROFESSIONAL-LISTS"

2.33. MHS OR Address

   Values of type MHS OR Address are encoded as strings, according to
   the format defined in [10].

2.34. Distribution List Submit Permission

   Values of type DLSubmitPermission are encoded as strings, according
   to the following BNF:

      <dlsubmit-perm> ::= <dlgroup_label> ':' <dlgroup-value>
                               | <dl-label> ':' <dl-value>

      <dlgroup-label> ::= 'group_member'

      <dlgroup-value> ::= <name>

      <name> ::= an encoded Distinguished Name

      <dl-label> ::= 'individual' | 'dl_member' | 'pattern'

      <dl-value> ::= <orname>

      <orname> ::= <address> '#' <dn>
             | <address>

      <address> ::= <add-label> ':' <oraddress>

      <dn> ::= <dn-label> ':' <name>

      <add-label> = 'X400'

      <dn-label> = 'X500'

   where <oraddress> is as defined in RFC 1327.

2.35. Photo

   Values of type Photo are encoded as if they were octet strings
   containing JPEG images in the JPEG File Interchange Format (JFIF), as
   described in [8].

2.36. Fax

   Values of type Fax are encoded as if they were octet strings
   containing Group 3 Fax images as defined in [7].

3. Security Considerations

    Security issues are not discussed in this memo.

4. Acknowledgements

    Many of the attribute syntax encodings defined in this document are
    adapted from those used in the QUIPU X.500 implementation. The
    contributions of the authors of the QUIPU implementation in the
    specification of the QUIPU syntaxes [4] are gratefully acknowledged.

5. Bibliography

    [1] The Directory: Selected Attribute Syntaxes. CCITT,
        Recommendation X.520.

    [2] Information Processing Systems -- Open Systems Interconnection --
        The Directory: Selected Attribute Syntaxes.

    [3] Barker, P., and S. Kille, "The COSINE and Internet X.500 Schema",
        RFC 1274, University College London, November 1991.

    [4] The ISO Development Environment: User's Manual -- Volume 5:
        QUIPU. Colin Robbins, Stephen E. Kille.

    [5] Kille, S., "A String Representation of Distinguished Names", RFC
        1779, ISODE Consortium, March 1995.

    [6] Kille, S., "A String Representation for Presentation Addresses",
        RFC 1278, University College London, November 1991.

    [7] Terminal Equipment and Protocols for Telematic Services -
        Standardization of Group 3 facsimile apparatus for document
        transmission. CCITT, Recommendation T.4.

    [8] JPEG File Interchange Format (Version 1.02). Eric Hamilton, C-
        Cube Microsystems, Milpitas, CA, September 1, 1992.

    [9] Yeong, W., Howes, T., and S. Kille, "Lightweight Directory Access
        Protocol", RFC 1777, Performance Systems International,
        University of Michigan, ISODE Consortium, March 1995.

   [10] Alvestrand, H., Kille, S., Miles, R., Rose, M., and S. Thompson,
        "Mapping between X.400 and RFC-822 Message Bodies", RFC 1495,
        SINTEF DELAB, ISODE Consortium, Soft*Switch, Inc., Dover Beach
        Consulting, Inc., Soft*Switch, Inc., August 1993.

6. Authors' Addresses

        Tim Howes
        University of Michigan
        ITD Research Systems
        535 W William St.
        Ann Arbor, MI 48103-4943
        USA

        Phone: +1 313 747-4454
        EMail: tim@umich.edu

        Steve Kille
        ISODE Consortium
        PO Box 505
        London
        SW11 1DX
        UK

        Phone: +44-71-223-4062
        EMail: S.Kille@isode.com

        Wengyik Yeong
        PSI Inc.
        510 Huntmar Park Drive
        Herndon, VA 22070
        USA

        Phone: +1 703-450-8001
        EMail: yeongw@psilink.com

        Colin Robbins
        NeXor Ltd
        University Park
        Nottingham
        NG7 2RD
        UK