              Augmented BNF for Syntax Specifications: ABNF

Status of this Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

TABLE OF CONTENTS

1.   INTRODUCTION

   Internet technical specifications often need to define a format
   syntax and are free to employ whatever notation their authors deem
   useful. Over the years, a modified version of Backus-Naur Form
   (BNF), called Augmented BNF (ABNF), has been popular among many
   Internet specifications. It balances compactness and simplicity,
   with reasonable representational power. In the early days of the
   Arpanet, each specification contained its own definition of ABNF.
   This included the email specifications, RFC733 and then RFC822 which
   have come to be the common citations for defining ABNF. The current
   document separates out that definition, to permit selective
   reference. Predictably, it also provides some modifications and
   enhancements.

   The differences between standard BNF and ABNF involve naming rules,
   repetition, alternatives, order-independence, and value ranges.
   Appendix A (Core) supplies rule definitions and encoding for a core
   lexical analyzer of the type common to several Internet
   specifications. It is provided as a convenience and is otherwise
   separate from the meta language defined in the body of this document,
   and separate from its formal status.

2.   RULE DEFINITION

2.1 Rule Naming

   The name of a rule is simply the name itself; that is, a sequence of
   characters, beginning with an alphabetic character, and followed by
   a combination of alphabetics, digits and hyphens (dashes).

        NOTE: Rule names are case-insensitive

   The names <rulename>, <Rulename>, <RULENAME> and <rUlENamE> all refer
   to the same rule.

   Unlike original BNF, angle brackets ("<", ">") are not required.
   However, angle brackets may be used around a rule name whenever their
   presence will facilitate discerning the use of a rule name. This is
   typically restricted to rule name references in free-form prose, or
   to distinguish partial rules that combine into a string not separated
   by white space, such as shown in the discussion about repetition,
   below.

2.2 Rule Form

   A rule is defined by the following sequence:

        name = elements crlf

   where <name> is the name of the rule, <elements> is one or more rule
   names or terminal specifications and <crlf> is the end-of- line
   indicator, carriage return followed by line feed. The equal sign
   separates the name from the definition of the rule. The elements
   form a sequence of one or more rule names and/or value definitions,
   combined according to the various operators, defined in this
   document, such as alternative and repetition.

   For visual ease, rule definitions are left aligned. When a rule
   requires multiple lines, the continuation lines are indented. The
   left alignment and indentation are relative to the first lines of the
   ABNF rules and need not match the left margin of the document.

2.3 Terminal Values

   Rules resolve into a string of terminal values, sometimes called
   characters. In ABNF a character is merely a non-negative integer.
   In certain contexts a specific mapping (encoding) of values into a
   character set (such as ASCII) will be specified.

   Terminals are specified by one or more numeric characters with the
   base interpretation of those characters indicated explicitly. The
   following bases are currently defined:

        b   = binary

        d   = decimal

        x   = hexadecimal

Hence:

        CR   = %d13

        CR   = %x0D

respectively specify the decimal and hexadecimal representation of
[US-ASCII] for carriage return.

A concatenated string of such values is specified compactly, using a
period (".") to indicate separation of characters within that value.
Hence:

        CRLF   = %d13.10

ABNF permits specifying literal text string directly, enclosed in
quotation-marks. Hence:

        command   = "command string"

Literal text strings are interpreted as a concatenated set of
printable characters.

        NOTE: ABNF strings are case-insensitive and
                   the character set for these strings is us-ascii.

Hence:

        rulename = "abc"

and:

        rulename = "aBc"

will match "abc", "Abc", "aBc", "abC", "ABc", "aBC", "AbC" and "ABC".

                To specify a rule which IS case SENSITIVE,
                    specify the characters individually.

For example:

        rulename   = %d97 %d98 %d99

    or

        rulename   = %d97.98.99

will match only the string which comprises only lowercased
characters, abc.

2.4 External Encodings

   External representations of terminal value characters will vary
   according to constraints in the storage or transmission environment.
   Hence, the same ABNF-based grammar may have multiple external
   encodings, such as one for a 7-bit US-ASCII environment, another for
   a binary octet environment and still a different one when 16-bit
   Unicode is used. Encoding details are beyond the scope of ABNF,
   although Appendix A (Core) provides definitions for a 7-bit US-ASCII
   environment as has been common to much of the Internet.

   By separating external encoding from the syntax, it is intended that
   alternate encoding environments can be used for the same syntax.

3.   OPERATORS

3.1 Concatenation   Rule1 Rule2

   A rule can define a simple, ordered string of values -- i.e., a
   concatenation of contiguous characters -- by listing a sequence of
   rule names. For example:

        foo   = %x61   ; a

        bar   = %x62   ; b

        mumble   = foo bar foo

        So that the rule <mumble> matches the lowercase string "aba".

        LINEAR WHITE SPACE: Concatenation is at the core of the ABNF
        parsing model. A string of contiguous characters (values) is
        parsed according to the rules defined in ABNF. For Internet
        specifications, there is some history of permitting linear white
        space (space and horizontal tab) to be freelyPand
        implicitlyPinterspersed around major constructs, such as
        delimiting special characters or atomic strings.

        NOTE: This specification for ABNF does not
                   provide for implicit specification of linear white
                   space.

   Any grammar which wishes to permit linear white space around
   delimiters or string segments must specify it explicitly. It is
   often useful to provide for such white space in "core" rules that are

then used variously among higher-level rules. The "core" rules might
be formed into a lexical analyzer or simply be part of the main
ruleset.

3.2 Alternatives   Rule1 / Rule2

Elements separated by forward slash ("/") are alternatives.
Therefore,

        foo / bar

will accept <foo> or <bar>.

            NOTE: A quoted string containing alphabetic
                       characters is special form for specifying alternative
                       characters and is interpreted as a non-terminal
                       representing the set of combinatorial strings with the
                       contained characters, in the specified order but with
                       any mixture of upper and lower case..

3.3 Incremental Alternatives   Rule1 =/ Rule2

It is sometimes convenient to specify a list of alternatives in
fragments. That is, an initial rule may match one or more
alternatives, with later rule definitions adding to the set of
alternatives. This is particularly useful for otherwise- independent
specifications which derive from the same parent rule set, such as
often occurs with parameter lists. ABNF permits this incremental
definition through the construct:

        oldrule   =/ additional-alternatives

So that the rule set

        ruleset   = alt1 / alt2

        ruleset   =/ alt3

        ruleset   =/ alt4 / alt5

is the same as specifying

        ruleset   = alt1 / alt2 / alt3 / alt4 / alt5

3.4 Value Range Alternatives   %c##-##

   A range of alternative numeric values can be specified compactly,
   using dash ("-") to indicate the range of alternative values. Hence:

        DIGIT   = %x30-39

   is equivalent to:

        DIGIT   = "0" / "1" / "2" / "3" / "4" / "5" / "6" /

                      "7" / "8" / "9"

   Concatenated numeric values and numeric value ranges can not be
   specified in the same string. A numeric value may use the dotted
   notation for concatenation or it may use the dash notation to specify
   one value range. Hence, to specify one printable character, between
   end of line sequences, the specification could be:

        char-line = %x0D.0A %x20-7E %x0D.0A

3.5 Sequence Group   (Rule1 Rule2)

   Elements enclosed in parentheses are treated as a single element,
   whose contents are STRICTLY ORDERED.   Thus,

          elem (foo / bar) blat

   which matches (elem foo blat) or (elem bar blat).

          elem foo / bar blat

   matches (elem foo) or (bar blat).

        NOTE: It is strongly advised to use grouping
                    notation, rather than to rely on proper reading of
                    "bare" alternations, when alternatives consist of
                    multiple rule names or literals.

   Hence it is recommended that instead of the above form, the form:

          (elem foo) / (bar blat)

   be used. It will avoid misinterpretation by casual readers.

   The sequence group notation is also used within free text to set off
   an element sequence from the prose.

3.6 Variable Repetition   *Rule

   The operator "*" preceding an element indicates repetition. The full
   form is:

        <a>*<b>element

   where <a> and <b> are optional decimal values, indicating at least
   <a> and at most <b> occurrences of element.

   Default values are 0 and infinity so that *<element> allows any
   number, including zero; 1*<element> requires at least one;
   3*3<element> allows exactly 3 and 1*2<element> allows one or two.

3.7 Specific Repetition   nRule

   A rule of the form:

        <n>element

   is equivalent to

        <n>*<n>element

   That is, exactly <N> occurrences of <element>. Thus 2DIGIT is a
   2-digit number, and 3ALPHA is a string of three alphabetic
   characters.

3.8 Optional Sequence   [RULE]

   Square brackets enclose an optional element sequence:

        [foo bar]

   is equivalent to

        *1(foo bar).

3.9 ; Comment

   A semi-colon starts a comment that continues to the end of line.
   This is a simple way of including useful notes in parallel with the
   specifications.

3.10 Operator Precedence

   The various mechanisms described above have the following precedence,
   from highest (binding tightest) at the top, to lowest and loosest at
   the bottom:

         Strings, Names formation
         Comment
         Value range
         Repetition
         Grouping, Optional
         Concatenation
         Alternative

   Use of the alternative operator, freely mixed with concatenations can
   be confusing.

         Again, it is recommended that the grouping operator be used to
         make explicit concatenation groups.

4.   ABNF DEFINITION OF ABNF

   This syntax uses the rules provided in Appendix A (Core).

         rulelist   = 1*(rule / (*c-wsp c-nl))

         rule   = rulename defined-as elements c-nl
                              ; continues if next line starts
                              ; with white space

         rulename   = ALPHA *(ALPHA / DIGIT / "-")

         defined-as   = *c-wsp ("=" / "=/") *c-wsp
                              ; basic rules definition and
                              ; incremental alternatives

         elements   = alternation *c-wsp

         c-wsp   = WSP / (c-nl WSP)

         c-nl   = comment / CRLF
                              ; comment or newline

         comment   = ";" *(WSP / VCHAR) CRLF

         alternation   = concatenation
                           *(*c-wsp "/" *c-wsp concatenation)

```
        concatenation  =  repetition *(1*c-wsp repetition)

        repetition     =  [repeat] element

        repeat         =  1*DIGIT / (*DIGIT "*" *DIGIT)

        element        =  rulename / group / option /
                          char-val / num-val / prose-val

        group          =  "(" *c-wsp alternation *c-wsp ")"

        option         =  "[" *c-wsp alternation *c-wsp "]"

        char-val       =  DQUOTE *(%x20-21 / %x23-7E) DQUOTE
                              ; quoted string of SP and VCHAR
                                 without DQUOTE

        num-val        =  "%" (bin-val / dec-val / hex-val)

        bin-val        =  "b" 1*BIT
                          [ 1*("." 1*BIT) / ("-" 1*BIT) ]
                              ; series of concatenated bit values
                              ; or single ONEOF range

        dec-val        =  "d" 1*DIGIT
                          [ 1*("." 1*DIGIT) / ("-" 1*DIGIT) ]

        hex-val        =  "x" 1*HEXDIG
                          [ 1*("." 1*HEXDIG) / ("-" 1*HEXDIG) ]

        prose-val      =  "<" *(%x20-3D / %x3F-7E) ">"
                              ; bracketed string of SP and VCHAR
                                 without angles
                              ; prose description, to be used as
                                 last resort
```

5.   SECURITY CONSIDERATIONS

   Security is truly believed to be irrelevant to this document.

6.   APPENDIX A - CORE

   This Appendix is provided as a convenient core for specific grammars.
   The definitions may be used as a core set of rules.

6.1 Core Rules

   Certain basic rules are in uppercase, such as SP, HTAB, CRLF,
   DIGIT, ALPHA, etc.

```
        ALPHA          =  %x41-5A / %x61-7A   ; A-Z / a-z

        BIT            =  "0" / "1"

        CHAR           =  %x01-7F
                             ; any 7-bit US-ASCII character,
                              excluding NUL

        CR             =  %x0D
                             ; carriage return

        CRLF           =  CR LF
                             ; Internet standard newline

        CTL            =  %x00-1F / %x7F
                             ; controls

        DIGIT          =  %x30-39
                             ; 0-9

        DQUOTE         =  %x22
                             ; " (Double Quote)

        HEXDIG         =  DIGIT / "A" / "B" / "C" / "D" / "E" / "F"

        HTAB           =  %x09
                             ; horizontal tab

        LF             =  %x0A
                             ; linefeed

        LWSP           =  *(WSP / CRLF WSP)
                             ; linear white space (past newline)

        OCTET          =  %x00-FF
                             ; 8 bits of data

        SP             =  %x20
```

```
                         ; space

    VCHAR             = %x21-7E
                         ; visible (printing) characters

    WSP               = SP / HTAB
                         ; white space
```

6.2 Common Encoding

   Externally, data are represented as "network virtual ASCII", namely
   7-bit US-ASCII in an 8-bit field, with the high (8th) bit set to
   zero. A string of values is in "network byte order" with the
   higher-valued bytes represented on the left-hand side and being sent
   over the network first.

7.   ACKNOWLEDGMENTS

   The syntax for ABNF was originally specified in RFC 733. Ken L.
   Harrenstien, of SRI International, was responsible for re-coding the
   BNF into an augmented BNF that makes the representation smaller and
   easier to understand.

   This recent project began as a simple effort to cull out the portion
   of RFC 822 which has been repeatedly cited by non-email specification
   writers, namely the description of augmented BNF. Rather than simply
   and blindly converting the existing text into a separate document,
   the working group chose to give careful consideration to the
   deficiencies, as well as benefits, of the existing specification and
   related specifications available over the last 15 years and therefore
   to pursue enhancement. This turned the project into something rather
   more ambitious than first intended. Interestingly the result is not
   massively different from that original, although decisions such as
   removing the list notation came as a surprise.

   The current round of specification was part of the DRUMS working
   group, with significant contributions from Jerome Abela , Harald
   Alvestrand, Robert Elz, Roger Fajman, Aviva Garrett, Tom Harsch, Dan
   Kohn, Bill McQuillan, Keith Moore, Chris Newman , Pete Resnick and
   Henning Schulzrinne.

8.   REFERENCES

   [US-ASCII]   Coded Character Set--7-Bit American Standard Code for
   Information Interchange, ANSI X3.4-1986.

   [RFC733] Crocker, D., Vittal, J., Pogran, K., and D. Henderson,
   "Standard for the Format of ARPA Network Text Message," RFC 733,
   November 1977.

   [RFC822] Crocker, D., "Standard for the Format of ARPA Internet Text
   Messages", STD 11, RFC 822, August 1982.

9.   CONTACT

   David H. Crocker   Paul Overell

   Internet Mail Consortium   Demon Internet Ltd
   675 Spruce Dr.   Dorking Business Park
   Sunnyvale, CA 94086 USA   Dorking
                                 Surrey, RH4 1HN
                                 UK

   Phone: +1 408 246 8253
   Fax: +1 408 249 6205
   EMail: dcrocker@imc.org   paulo@turnpike.com

10. Full Copyright Statement