|  |
| --- |
| Title: LOON –  Line Oriented Object Notation |

**Contents**

1. LOON – Line Oriented Object Notation 1

1.1. String values 5

1.2. Comment Directives 6

|  |
| --- |
|  |

# LOON – Line Oriented Object Notation

LOON is a simple file format for configuration data. It is intended to be easy for both humans and machines to read and write. It is a stripped-down form of JSON, that ends up looking similar to the format used by HTTP, SMTP etc.

An example LOON message is as follows:

# Some fake details about me

com.codalogic.aboutme {

Name: Pete

Height: 178

DoB: 1969-04-18

Children [

{

Name: Sarah

Height: 170

}

{

Name: Jenny

Height: 144

}

]

Grades [

A

B

C

]

PlaceOfBirth: " string with leading spaces! "

History <<END

Born a long time again

in a galaxy far, far away.

<<END

MoreHistory <<END " "

Multiline string with preamble to

be ignored.

<<END

}

The ABNF is as follows (note that LOON is encoded in UTF-8 or US-ASCII. This ABNF is written in terms of bytes, not Unicode codepoints):

loon = preamble [ object-body / object / array ]

[ eol ]

preamble = \*( ( ows / comment ) eol )

comment = ows "#" \*not-eol

object-body = object-line \*( eol object-line )

object-line = comment / object-member

object-member = ows full-name ows value

full-name = [ realm "." ] [ "@" ] name

realm = name \*( "." name )

name = ALPHA \*( ALPHA / DIGIT / "-" / "\_" )

value = object / array / multiline-string /

primitive-spec / null1

object = "{" eol [ object-body eol ] ows "}"

array = "[" eol [ array-body eol ] ows "]"

array-body = array-line \*( eol array-line )

array-line = array-member ; Comments not allowed

array-member = comment / ows value

primitive-spec = ":" ows primitive-value

primitive-value = null2 / true / false / number /

inline-string

null1 = "" ; Empty member value field indicates null

null2 = "\0"

true = true-kw

false = false-kw

; From RFC8259

number = [ minus ] int [ frac ] [ exp ]

decimal-point = %x2E ; .

digit1-9 = %x31-39 ; 1-9

e = %x65 / %x45 ; e E

exp = e [ minus / plus ] 1\*DIGIT

frac = decimal-point 1\*DIGIT

int = zero / ( digit1-9 \*DIGIT )

minus = %x2D ; -

plus = %x2B ; +

zero = %x30 ; 0

inline-string = naked-string / quoted-string

naked-string = \*char ; See notes on strings

quoted-string = quotation-mark \*char quotation-mark

multiline-string = "<<" name [ ows preamble ] eol

\*( \*not-eol eol )

[ \*not-eol ] "<<" name

preamble = quoted-string

char = unescaped / escaped

unescaped = HTAB / %x20-5B / %x5D-FF

; not controls except TAB nor "\"

; N.B: quotation mark is NOT escaped

escaped = escape (

escape / ; \ i.e.: \\ -> \

; N.B. quotation-mark is NOT escaped

%x62 / ; b i.e.: \b -> backspace

%x66 / ; f i.e.: \f -> form feed

%x6E / ; n i.e.: \n -> line feed

%x72 / ; r i.e.: \r -> carriage return

%x74 / ; t i.e.: \t -> tab

%x75 (4HEXDIG / "{" 1\*6HEXDIG "}")

; \uXXXX or \u{XXXXXX} -> U+XXXX

escape = %x5C ; \

quotation-mark = %x22 ; "

eol = ows ( CR [ LF ] / LF )

not-eol = HTAB / %x20-FF

ows = \*WSP ; Optional white space

;; Keywords

true-kw = %x74.72.75.65 ; "true"

false-kw = %x66.61.6C.73.65 ; "false"

;; Referenced RFC 5234 Core Rules

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

CR = %x0D ; carriage return

DIGIT = %x30-39 ; 0-9

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

HTAB = %x09 ; horizontal tab

LF = %x0A ; linefeed

SP = %x20 ; space

WSP = SP / HTAB ; white space

## String values

LOON string values need special treatment.

Leading and trailing whitespace of a string value will be automatically removed on parsing. If that whitespace is significant, make the string a quoted string by wrapping it in quotation marks, e.g.:

Description: " A string with leading whitespace "

Quotation marks within a quoted string are not escaped in any way:

Description: "A string with " marks in it"

A string must also be quoted if, after removing any whitespace from both ends of the string, any of the following apply:

* The string is a value in an array and consists solely of a single '{', '[' or ']' character
* The string is a value in an array and begins with a '<' character and matches the ABNF ("<<" name)
* The string is a value in an array and begins with a '#' character

For example:

# An object start

Example1 {

}

# A string consisting only of '{'

Example2: {

# The string "["

Example3: [

Example4 [

# A comment in an array

"# A string that starts with a comment marker"

] A string, not an array end

# The following is the string ] in an array

"]"

]

# Not an array start. It is an object-member value

Example5: [

Example6: << A simple-string, not a multiline-string

A multiline string begins with the "<<" HEREDOC marker followed by a name used to mark the end of the multiline string, e.g.:

LongMessage <<END

A message

that is long

goes here.<<END

Note that, unlike other HEREDOC formats, the end marker doesn't have to appear on its own line. It just needs to appear at the end of a line.

Multiline strings can also specify a line preamble that is to be removed from each line in the multiline string before supplying the string to the application. This is specified using a quoted string following the HEREDOC marker and the name used to mark the end of the multiline string. Typically, the preamble would be whitespace to enable more aesthetic display of the string but it needn't be. For example, the following

LongMessage <<END " "

A message

that is long

goes here.<<END

Would yield:

A message

that is long

goes here.

instead of:

A message

that is long

goes here.

## Comment Directives

Comments that have the pling character ("!") immediately following the opening comments marker ("#") are "Comment Directives". These are comments that may potentially be processed by a machine. Their effect may change the way subsequent LOON content is processed. The ABNF of a Comment Directive is:

comment-directive = "#!" directive-name ows \*not-eol

directive-name = full-name

For example:

#!org.example.extensions Mode1 Mode2

No Comment Directives are defined at this time.

|  |
| --- |
|  |

**History**

|  |  |  |
| --- | --- | --- |
| Issue | Date | Change |
| A | 2 May 19 | Creation |
| B | 19 Nov 21 | Enable comments in arrays, clarify use of quoted strings for escaping, allow @ in object member names and add comment directives |
| C | 11 May 22 | Allow specifying preamble characters in multiline strings |