Regular Expression Notation

https://campus.barracuda.com/doc/17629564/

The Barracuda Load Balancer ADC employs a regular expression (regex) engine to evaluate regular expressions (as defined in POSIX 1003.2) used as values in various parameters. Regular expressions allow you to specify complex relationships. The following table describes syntax rules that apply when creating a regular expression for a parameter value.

Regular expressions use raw bytes/characters for everything except for NUL(0x00 that gets escaped to %00) and LF(0x0a that gets escaped to %0a).

Value	Meaning
x	Match the character x.
.	Match any character (byte) except newline.
[xyz]	Match the pattern (character class) among x, y, or z. Matching is case dependent.
[abj-oZ]	Match the pattern (character class with a range) among a, b, any letter from j through o, or Z. Matching is case dependent.
[^A-Z]	Match anything except the pattern (negated character class), that is, any character but those in the class, which in this case is any character except an uppercase letter.
[^A-Z\n]	Match anything except the pattern (negated character class), which in this case is any character except an uppercase letter or a newline.
r+	Match zero or more of r, where r is any regular expression.
r?	Match zero or one of r (that is, an optional r), where r is any regular expression.
r{2,5}	Match two to five of r.
r{2,}	Match two or more of r.
r{4}	Match exactly 4 of r.
"*[xyz]\\"foo*"	Match the literal string: [xyz]"foo
\X	If X is an a, b, f, n, r, t, or v, then match the ANSI-C interpretation of \x applies. Otherwise, it is a literal X (used to escape operators such as an asterisk [*]).
\0	Match a NULL character (ASCII code 0).
\123	Match the character with octal value 123.
\x2a	Match the character with hexadecimal value 2a.
(r)	Match the r. Parentheses are used to override precedence; expressions in parentheses are evaluated first.
rs	Match the regular expression r followed by the regular expression s. This type of pattern is called *concatenation*.
r|s	Match either an r or an s. This type of pattern is called *alternation*.

r/s	Match an r if it is followed by an s. The text matched by s is included when determining whether this rule is the "longest match," but it is then returned to the input before the action is executed, so the action only sees the text matched by r. This type of pattern is called *trailing context*.
^r	Match an r at the beginning of a line (that is, when starting to scan or immediately after a newline has been scanned). **Note**: The circumflex (^) character means beginning of the input string when it appear at the beginning of a pattern. If it appears elsewhere, it is treated as a character.
r$	Match an r at the end of a line (that is, just before a newline). This is equivalent to r/\n. **Note**: The dollar sign ($) character means end of the input string when it appear at the end of a pattern. If it appears elsewhere, it is treated as a character.

The following are special characters (that is, have special meaning as described in the above table) and must be escaped by prefixing a back-slash (\) in order to be recognized as the character itself:

```
. [ ] ( ) ^ $ / * + ? { } \ |
```

The following characters must be escaped or quoted during header rule configuration for proper rule matching:
White spaces[' ', '\t', '\n'], the brackets '[' '(' and ')' ']] and ';'
Precede each character with the "\" character to escape it, or quote the entire string.

The regular expressions listed in Regular Expression Values table are grouped according to precedence, from highest precedence at the top to lowest at the bottom. For example, the following two expressions are identical because the asterisk (*) operator has higher precedence than concatenation, and concatenation has higher precedence than alternation (|):

```
foo|bar*
(foo)|(ba(r*))
```

This pattern matches either the string foo or the string ba followed by zero or more r strings.

Inside a character class, all regular expression operators lose their special meaning except escape (\) and the character class operators dash (-), right bracket (]), and circumflex (^) at the beginning of the class.

Valid character class expressions are the following:

```
[:alnum:] [:alpha:] [:blank:]
[:cntrl:] [:digit:] [:graph:]
[:lower:] [:print:] [:punct:]
[:space:] [:upper:] [:xdigit:]
```

These expressions are equivalent to the corresponding standard C is XXX function. If used in caseinsensitive mode, [:upper:] and [:lower:] are equivalent to [:alpha:].

A rule can have at most one instance of the / or $ operators. The start condition (^) can only occur at the beginning of a pattern, none of these operators can be grouped inside parentheses. A ^ character that does not occur at the beginning of a rule or a $ character that does not occur at the end of a rule loses its special properties and is treated as a normal character.

If more than one match is found, the rule matching the most text is used. If two or more matches are of the same length, the first rule is chosen.

Usage Examples:

- `^r`: Match the beginning of an input string only. For example, ^[a-z]+ matches foo but does not match 1foo because the latter does not begin with an alphabetic character.
- `[^a-z]`: Negate character class. This form matches anything other than a lower case alphabetic character. For example, ^[^a-z] matches 1foo but does not match foo.
- ^ anywhere else: Literal character. For example, ^(^|[a-z]) matches foo and ^1foo but does not match 1foo.

Usage Examples: $

- `r$`: Match the end of an input string only. For example, [a-z]+$ matches foo and 1foo but does not match foo1.
- $ anywhere else: Literal character. For example, ([a-z]+|$) matches foo, 1foo, foo1, and foo$.

Usage Examples: Combinations

- `^r$`: Match the pattern exactly. There can be no additional characters.
- `(r1|r2$)`: The dollar sign is treated as a literal character.
- `(^r1|r2)`: The circumflex is treated as a literal character.