



Regular Expressions

This appendix contains information about the regular expression syntax used by the application appliance. The syntax is the GNU POSIX regular expression syntax.

- [Regular Expression Reference, page F-1](#)
- [Regular Expression Pattern Examples, page F-2](#)
- [Additional Information, page F-3](#)

Regular Expression Reference

This section contains regular expression reference information. The following table contains the most common regular expression metacharacters, but is not comprehensive.

Table F-1 *Regular Expression Syntax*

Metacharacter	Description
.	Matches any single character. For example, the regular expression <code>r.t</code> matches the strings <code>rat</code> , <code>rut</code> , <code>r t</code> , but not <code>root</code> .
\$	Matches the end of a line. For example, the regular expression <code>you\$</code> matches the end of the string “Thank you” but not the string “Thank you very much.”
^	Matches the beginning of a line. For example, the regular expression <code>^when in</code> matches the beginning of the string “When in the course of human events” but not the string “What and When in the”
*	Matches zero or more occurrences of the character immediately preceding. For example, the regular expression <code>.*</code> means match any number of any characters.
\	This is the quoting character; use it to treat the following metacharacter as an ordinary character. For example, <code>\\$</code> is used to match the dollar sign character (\$) rather than the end of a line. Similarly, the expression <code>\.</code> is used to match the period character rather than any single character.
[] [c1-c2] [^c1-c2]	Matches any one of the characters between the brackets. For example, the regular expression <code>r[ao]t</code> matches <code>rat</code> , <code>rot</code> , and <code>rut</code> , but not <code>ret</code> . Ranges of characters are specified by a beginning character (<i>c1</i>), a hyphen, and an ending character (<i>c2</i>). For example, the regular expression <code>[0-9]</code> means match any digit. Multiple ranges can be specified as well. The regular expression <code>[A-Za-z]</code> means match any upper or lower case letter. To match any character except those in the range (that is, the complement range), use the caret as the first character after the opening bracket. For example, the expression <code>[^269A-Z]</code> matches any characters except 2, 6, 9, and uppercase letters.
\< \>	Matches the beginning (\<) or end (\>) of a word. For example, the expression <code>\<the</code> matches “the” in the string “for the wise” but does not match “the” in “otherwise.”

Table F-1 Regular Expression Syntax (continued)

Metacharacter	Description
()	Treat the expression between (and) as a group. Also, saves the characters matched by the expression into temporary holding areas. Up to nine pattern matches can be saved by a single regular expression. They can be referenced as \1 through \9 (on the same line only).
	Logical OR two conditions together. For example (him her) matches the line “it belongs to him” and matches the line “it belongs to her” but does not match the line “it belongs to them.”
+	Matches one or more occurrences of the character or regular expression immediately preceding. For example, the regular expression 9+ matches 9, 99, and 999.
?	Matches 0 or 1 occurrence of the character or regular expression immediately preceding.
{i}	Matches a specific number (i) or range (i through j) of instances of the preceding character. For example, the expression A[0-9]{3} matches “A” followed by exactly 3 digits. That is, it matches A123 but not A1234. The expression [0-9]{4,6} matches any sequence of 4, 5, or 6 digits.
{i,j}	

Note that the regular expression evaluator matches the **longest** pattern possible.

For example, say you want to parse a URL. You might use a regular expression like this:

```
(http)://(.*)/(.*)
```

expecting to parse a URL such as “http://www.example.com/images/kournikova.jpg” into these strings:

group 0: “http://www.example.com/images/kournikova.jpg”

group 1: “http”

group 2: “www.example.com”

group 3: “/images/kournikova.jpg”

However, this attempt yields unexpected results for groups 2 and 3:

group 2: “www.example.com/images/”

group 3: “kournikova.jpg”

This occurs because the group 2 string that the regular expression evaluator found represents the longest match for the given pattern. The matched string doesn’t end with the first / found, it ends with the last one found.

To achieve the desired result, use this regular expression:

```
http://([^\/]*)/(.*)$
```

This works because the first group (the first expression in parenthesis) matches 0 or more characters up to any that is not in the set that includes /.

Regular Expression Pattern Examples

Here are some examples of regular expression patterns.

Table F-2 Regular Expression Pattern Examples

Pattern	Regular Expression
IP Address	(\d{1,3})\.\d{1,3})\.\d{1,3})\.\d{1,3})
Domain Name	^[a-zA-Z]([a-zA-Z0-9-][a-zA-Z0-9])?\.[a-zA-Z]([a-zA-Z0-9-][a-zA-Z0-9])?(\.[a-zA-Z]([a-zA-Z0-9-][a-zA-Z0-9])?)?\$
E-mail addresses	{^[A-Za-z0-9._-]+@[A-Za-z0-9.-]+\$}
URL	("http://" mailto:" ftp://")[^ \n\r"\<\>\+]

Additional Information

For more information about regular expressions, refer to a POSIX regular expression reference book or one of these Web references:

- http://www.gnu.org/software/grep/doc/grep_7.html
- http://www.ugcs.caltech.edu/info/regex/regex_2.html

