

12. Regular expressions

Overview

[Perl Compatible Regular Expressions](#) (PCRE) are supported in Zabbix.

There are two ways of using regular expressions in Zabbix:

- manually entering a regular expression
- using a global regular expression created in Zabbix

Regular expressions

You may manually enter a regular expression in supported places. Note that the expression may not start with @ because that symbol is used in Zabbix for referencing global regular expressions.

Global regular expressions

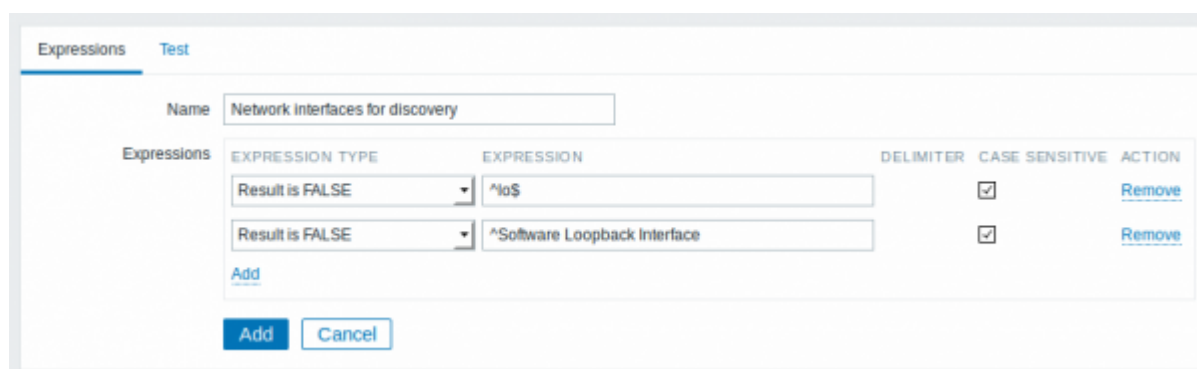
There is an advanced editor for creating and testing complex regular expressions in Zabbix frontend.

Once a regular expression has been created this way, it can be used in several places in the frontend by referring to its name, prefixed with @, for example, @mycustomregexp.

To create a global regular expression:

- Go to: *Administration* → *General*
- Select *Regular expressions* from the dropdown
- Click on *New regular expression*

The **Expressions** tab allows to set the regular expression name and add subexpressions.



Parameter	Description
Name	Set the regular expression name. Any Unicode characters are allowed.
Expressions	Click on <i>Add</i> in the Expressions block to add a new subexpression.

Parameter	Description
Expression type	Select expression type: Character string included - match the substring Any character string included - match any substring from a delimited list. The delimited list includes a comma (,), a dot (.) or a forward slash (/). Character string not included - match any string except the substring Result is TRUE - match the regular expression Result is FALSE - do not match the regular expression
Expression	Enter substring/regular expression.
Delimiter	A comma (,), a dot (.) or a forward slash (/) to separate text strings in a regular expression. This parameter is active only when "Any character string included" expression type is selected.
Case sensitive	A checkbox to specify whether a regular expression is sensitive to capitalization of letters.

Since Zabbix 2.4.0, a forward slash (/) in the expression is treated literally, rather than a delimiter. This way it is possible to save expressions containing a slash, whereas previously it would produce an error.

A custom regular expression name in Zabbix may contain commas, spaces, etc. In those cases where that may lead to misinterpretation when referencing (for example, a comma in the parameter of an item key) the whole reference may be put in quotes like this: "@My custom regexp for purpose1, purpose2".

Regular expression names must not be quoted in other locations (for example, in LLD rule properties).

Example

Use of the following regular expression in LLD to discover databases not taking into consideration a database with a specific name:

```
^TESTDATABASE$
```

Chosen *Expression type*: "Result is FALSE". Doesn't match name, containing string "TESTDATABASE".

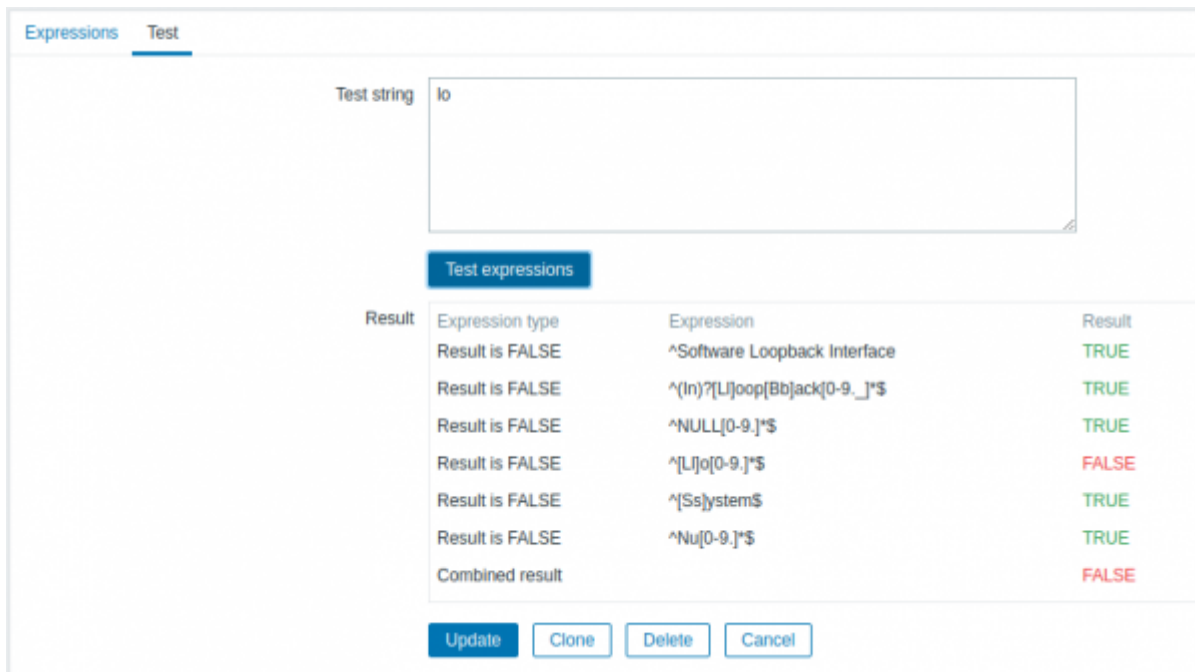
Test string

Test expressions

Result	Expression type	Expression	Result
	Result is FALSE	^TESTDATABASE	FALSE
	Combined result		FALSE

More complex example

A custom regular expression may consist of multiple subexpressions, and it can be tested in the **Test** tab by providing a test string.



Results show the status of each subexpression and total custom expression status.

Total custom expression status is defined as *Combined result*. If several sub expressions are defined Zabbix uses AND logical operator to calculate *Combined result*. It means that if at least one Result is False *Combined result* has also False status.

Explanation of global regular expressions

Global regexp	Expression	Description
File systems for discovery	^(btrfs ext2 ext3 ext4 jfs reiser xfs ffs ufs jfs jfs2 vxfs hfs refs ntfs fat32 zfs)\$	Matches "btrfs" or "ext2" or "ext3" or "ext4" or "jfs" or "reiser" or "xfs" or "ffs" or "ufs" or "jfs" or "jfs2" or "vxfs" or "hfs" or "refs" or "ntfs" or "fat32" or "zfs"
Network interfaces for discovery	^Software Loopback Interface	Matches strings starting with "Software Loopback Interface"
	^lo\$	Matches "lo"
	^(In)?[Ll]oop[Bb]ack[0-9._]*\$	Matches strings that optionally start with "In", then have "L" or "l", then "oop", then "B" or "b", then "ack", which can be optionally followed by any number of digits, dots or underscores
	^NULL[0-9.]*\$	Matches strings starting with "NULL" optionally followed by any number of digits or dots
	^[Ll]o[0-9.]*\$	Matches strings starting with "Lo" or "lo" and optionally followed by any number of digits or dots
	^[Ss]ystem\$	Matches "System" or "system"
	^Nu[0-9.]*\$	Matches strings starting with "Nu" optionally followed by any number of digits or dots
Storage devices for SNMP discovery	^(Physical memory Virtual memory Memory buffers Cached memory Swap space)\$	Matches "Physical memory" or "Virtual memory" or "Memory buffers" or "Cached memory" or "Swap space"

Global regexp	Expression	Description
Windows service names for discovery	<code>^(MMCSS gupdate SysmonLog clr_optimization_v2.0.50727_32 clr_optimization_v4.0.30319_32)\$</code>	Matches "MMCSS" or "gupdate" or "SysmonLog" or strings like "clr_optimization_v2.0.50727_32" and "clr_optimization_v4.0.30319_32" where instead of dots you can put any character except newline.
Windows service startup states for discovery	<code>^(automatic automatic delayed)\$</code>	Matches "automatic" or "automatic delayed".

Regular expression support by location

Location	Regex support	Global regexp support	Comments
Macro functions			
<code>regexsub()</code> <code>iregsub()</code>	Yes	No	pattern parameter
Trigger functions			
<code>count()</code>	Yes	Yes	pattern parameter if operator parameter is <i>regexp</i> or <i>iregexp</i>
<code>logeventid()</code>			pattern parameter
<code>iregexp()</code>			
<code>regexp()</code>			
Low-level discovery			
	Yes	Yes	<i>Filter</i> field
Web monitoring			
	Yes	No	<i>Variables</i> with a regex: prefix <i>Required string</i> field
Zabbix agent items			
<code>eventlog[]</code>	Yes	Yes	regexp , severity , source , eventid parameters
<code>log[]</code> <code>log.count[]</code>			regexp parameter
<code>logrt[]</code> <code>logrt.count[]</code>			regexp parameter supports both, file_regexp parameter supports non-global expressions only
<code>proc.cpu.util[]</code> <code>proc.mem[]</code> <code>proc.num[]</code> <code>sensor[]</code> <code>system.hw.macaddr[]</code> <code>system.sw.packages[]</code> <code>vfs.dir.size[]</code> <code>vfs.file.regexp[]</code> <code>vfs.file.regmatch[]</code> <code>web.page.regexp[]</code>	Yes	No	cmdline parameter device and sensor parameters on Linux 2.4 interface parameter package parameter regex_incl and regex_excl parameters regexp parameter

Location	Regex support	Global regex support	Comments
SNMP traps			
snmptrap[]	Yes	Yes	regex parameter
Icon mapping			
	Yes	Yes	<i>Expression</i> field
Item value preprocessing			
	Yes	No	pattern parameter

From:

<https://www.zabbix.com/documentation/3.4/> - **Zabbix Documentation 3.4**

Permanent link:

https://www.zabbix.com/documentation/3.4/manual/regular_expressions?rev=1513600168

Last update: **2017/12/18 12:29**

