

# 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

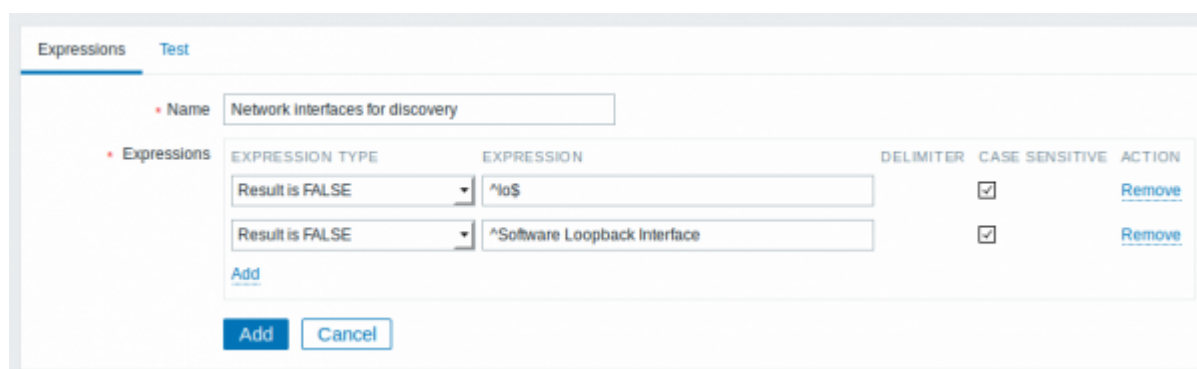
There is an advanced editor for creating and testing complex regular expressions in Zabbix frontend.

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.



All mandatory input fields are marked with a red asterisk.

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: <b>Character string included</b> - match the substring <b>Any character string included</b> - match any substring from a delimited list. The delimited list includes a comma (,), a dot (.) or a forward slash (/). <b>Character string not included</b> - match any string except the substring <b>Result is TRUE</b> - match the regular expression <b>Result is FALSE</b> - 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$
```

Test string

Test expressions

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

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

### Example with an inline regex modifier

Use of the following regular expression including an inline modifier (?i) to match the characters "error":

```
(?i)error
```

Test string: `Sometexthere1345Error1357`

Test expressions

Result	Expression type	Expression	Result
	Result is TRUE	(?i)error	TRUE
	Combined result		TRUE

Chosen Expression type: "Result is TRUE". Characters "error" are matched.

### Another example with an inline regex modifier

Use of the following regular expression including multiple inline modifiers to match the characters after a specific line:

```
(?<=match (?i)everything(?-i) after this line\n)(?sx).*# we add s modifier to allow . match newline characters
```

Test string: `Some text here for your consideration  
1235kfd345  
match eveRything after this line  
Continuation`

Test expressions

Result	Expression type	Expression	Result
	Result is TRUE	(?<=match (?i)everything(?-i) after this line\n)(?sx).*# we add s modifier to allow . match newline characters	TRUE
	Combined result		TRUE

Chosen Expression type: "Result is TRUE". Characters after a specific line are matched.

**g** modifier can't be specified in line. The list of available modifiers can be found in [pcresyntax man page](#). For more information about PCRE syntax please refer to [PCRE HTML documentation](#).

### 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.

The screenshot shows the 'Test' tab in the Zabbix Expressions configuration. A text input field labeled 'Test string' contains the value 'lo'. Below this, a blue button labeled 'Test expressions' is visible. Underneath, a table displays the results of testing various expressions against the test string. The table has three columns: 'Result', 'Expression type', 'Expression', and 'Result'. The 'Result' column shows 'FALSE' for each individual expression, and the 'Combined result' is also 'FALSE'. At the bottom of the table, there are four buttons: 'Update', 'Clone', 'Delete', and 'Cancel'.

Result	Expression type	Expression	Result
Result is FALSE	Expression type	^Software Loopback Interface	TRUE
Result is FALSE	Expression type	^(In)?[Ll]oop[Bb]ack[0-9._]*\$	TRUE
Result is FALSE	Expression type	^NULL[0-9.]*\$	TRUE
Result is FALSE	Expression type	^[Ll]o[0-9.]*\$	FALSE
Result is FALSE	Expression type	^[Ss]ystem\$	TRUE
Result is FALSE	Expression type	^Nu[0-9.]*\$	TRUE
	Combined result		FALSE

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	Regular expression	Global regular expression	Comments			
<b>Agent items</b>						
<ul style="list-style-type: none"> <li>eventlog[]</li> <li>log[]</li> <li>log.count[]</li> <li>logrt[]</li> <li>logrt.count[]</li> <li>proc.cpu.util[]</li> <li>proc.mem[]</li> <li>proc.num[]</li> <li>sensor[]</li> <li>system.hw.macaddr[]</li> <li>system.sw.packages[]</li> <li>vfs.dir.count[]</li> <li>vfs.dir.size[]</li> <li>vfs.file.regexp[]</li> <li>vfs.file.regmatch[]</li> <li>web.page.regexp[]</li> </ul>	Yes	Yes	regexp, severity, source, eventid parameters			
			Yes/No	regexp parameter		
				regexp parameter supports both, file_regexp parameter supports non-global expressions only		
		No	cmdline parameter			
			device and sensor parameters on Linux 2.4			
			interface parameter			
			package parameter			
			regex_incl and regex_excl parameters			
			regex_incl and regex_excl parameters			
			regexp parameter			
			<b>SNMP traps</b>			
			snmptrap[]	Yes	Yes	regexp parameter
			<b>Item value preprocessing</b>	Yes	No	pattern parameter
		<b>Trigger functions</b>				
<ul style="list-style-type: none"> <li>count()</li> <li>logeventid()</li> <li>iregexp()</li> <li>regexp()</li> </ul>	Yes	Yes	pattern parameter if operator parameter is <i>regexp</i> or <i>iregexp</i>			
			pattern parameter			
			<b>Low-level discovery</b>			
			Filter field			
<b>Web monitoring</b>	Yes	No	Variables with a <b>regex:</b> prefix Required string field			

Location	Regular expression	Global regular expression	Comments
<b>Macro functions</b>			
regsub() iregsub()	Yes	No	pattern parameter
<b>Icon mapping</b>	Yes	Yes	<i>Expression</i> field

From: <https://www.zabbix.com/documentation/current/> - **Zabbix Documentation 5.0**

Permanent link: [https://www.zabbix.com/documentation/current/manual/regular\\_expressions?rev=1521025508](https://www.zabbix.com/documentation/current/manual/regular_expressions?rev=1521025508)

Last update: **2019/10/07 06:35**

