



## Zabbix Blog

The Future of Monitoring

# Zabbix Log File Monitoring



Watch the [video](#) now.

## Contents

- I. [Introduction](#) (0:01)
- II. [Log file parameter overview](#) (2:55)
- III. [Log file monitoring](#) (7:44)

## Introduction

Previously, we talked about quite a lot of stuff – the installation of [Zabbix server](#) and [proxy](#), [Docker](#), [Timescale](#), [Prometheus](#), [XPath](#), [inventory](#), [templates](#), and [item agent configurations](#).

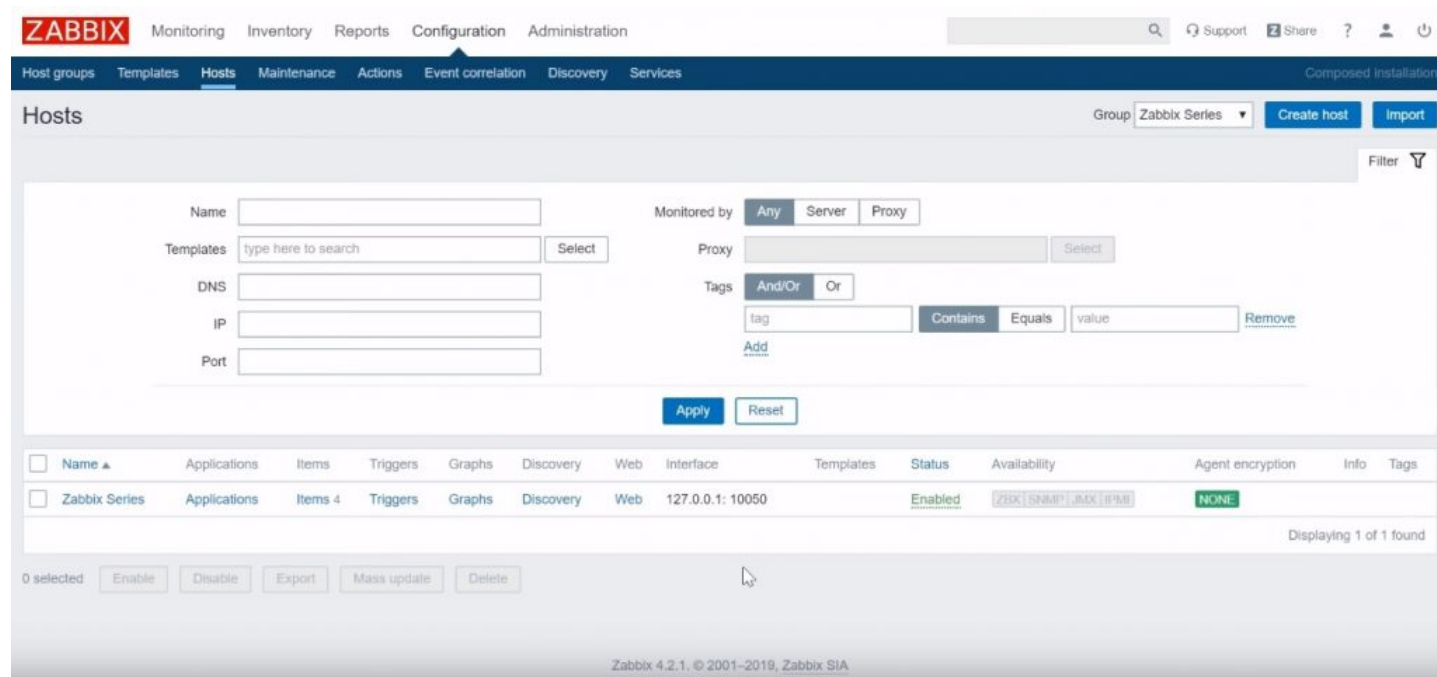
The topic for today will be log file monitoring on Windows or Linux machines.

Before we start, remember that native log file monitoring is achieved with Zabbix agent. But what's most important is that you must use Zabbix **agent active mode**. It is not possible to use the log items and do log file monitoring with Zabbix agent passive item types.

Let's quickly remind ourselves what we need to keep in mind for our agent active checks to actually work.

Things to keep in mind

**Note.** *I already have my Zabbix Series host in the front end. The version is the recent one — 4.2.1, and I have CLI.*



Front end

Remember, in the agent config file which is located in **/etc/zabbix/zabbix\_agentd.conf** we basically need to use two parameters:

1) **ServerActive** which is the IP or DNS name of your Zabbix server or the proxy if this agent is monitored by the proxy.

```
ServerActive=127.0.0.1,172.16.238.3
```

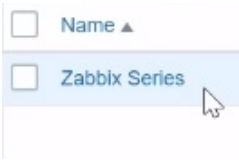
IP or DNS name of the Zabbix server

This will be the IP or the DNS name to which the agent, first of all, will connect and request the configuration like “*What do I need to monitor?*” It will be a log file, in this case. And then, IP or DNS name to which agent will send the data.

2) **Host name** (must match the **Host name** in the front end).

Hostname=Zabbix Series

Host name in the configuration file



Host name in the front end

It is case-sensitive: if you have an upper case S in the front end, then it must be also upper case in the configuration file. Otherwise, you will have your items in the front end, Zabbix server, Zabbix agent up and running without any strange error messages but still, you will not see the latest data in the front end. So remember, the Host names must match and they are case sensitive. And don't mess up – we are talking about a **Host name**, not about a **Visible name**.

\* Host name

Visible name

Host name and Visible name

That would be a quick review of what we need to keep in mind for our active checks to actually work. Now let's get into the actual items.

## Log file parameter overview

I've already prepared 4 log monitoring items, and each of them is doing different things.

<input type="checkbox"/>	Wizard	Name ▲	Triggers	Key	Interval	History	Trends	Type	Applications	Status	Info
<input type="checkbox"/>	***	Log file monitoring		log[/var/log/zabbix/zabbix_proxy.log,,,skip,,]	1s	90d		Zabbix agent (active)		Enabled	
<input type="checkbox"/>	***	Log file monitoring ( Regexp )		log[/var/log/zabbix/zabbix_proxy.log,"processed","[0-9]+ values in. "[0-9]+""...,skip,,]	1s	90d		Zabbix agent (active)		Enabled	
<input type="checkbox"/>	***	Log file monitoring ( Regexp and Output )		log[/var/log/zabbix/zabbix_proxy.log,"processed","[0-9]+ values in. ([0-9]+. "[0-9]+""...,skip,\1,]	1s	90d		Zabbix agent (active)		Enabled	
<input type="checkbox"/>	***	Log file monitoring ( Regexp and Output and Timestamp )		log[/var/log/zabbix/zabbix_proxy.log,"processed","[0-9]+ values in. ([0-9]+. "[0-9]+""...,skip,\0,]	1s	90d		Zabbix agent (active)		Enabled	

Displaying 4 of 4 found

You can collect all new lines in the log file where you can do some filtering, matching based on the regular expression, extracting some specific output of your extracted log line, display it in the graphs, etc.

Let's go to [documentation](#). Just like any other item, the log item has a full explanation with examples — explanation of the parameters, what does what in our documentation. To find this, go to **Configuration > Items > Item types > Zabbix agent** and here you will find the log item.



Zabbix agent documentation

log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>]				
Log file monitoring.	Log	<b>file</b> - full path and name of log file <b>regexp</b> - regular expression <sup>4</sup> describing the required pattern <b>encoding</b> - code page <b>identifier</b> <b>maxlines</b> - maximum number of new lines per second the agent will send to Zabbix server or proxy. This parameter overrides the value of 'MaxLinesPerSecond' in <b>zabbix_agentd.conf</b> <b>mode</b> - possible values: <i>all</i> (default), <i>skip</i> - skip processing of older data (affects only newly created items). <b>output</b> - an optional output formatting template. The <b>\0</b> escape sequence is replaced with the matched part of text (from the first	The item must be configured as an <b>active check</b> . If file is missing or permissions do not allow access, item turns unsupported.  If <b>output</b> is left empty - the whole line containing the matched text is returned. Note that all global regular expression types except 'Result is TRUE' always return the whole matched line and the <b>output</b> parameter is ignored.  Content extraction using the <b>output</b> parameter takes place on the agent.  Examples: → log[/var/log/evlog]	

Documentation of log file

**Note.** Very similar is **logrt**. The only difference is that it is for the logs that are rotating — changing their names based on some conditions, for example, the log file size.

logrt[file_regexp,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>]			
Log file monitoring with log rotation support.	Log	<p><b>file_regexp</b> - absolute path to file and regular expression<sup>4</sup> describing the file name pattern</p> <p><b>regexp</b> - regular expression<sup>4</sup> describing the required content pattern</p> <p><b>encoding</b> - code page <b>identifier</b></p> <p><b>maxlines</b> - maximum number of new lines per second the agent will send to Zabbix server or proxy. This parameter overrides the value of 'MaxLinesPerSecond' in <b>zabbix_agentd.conf</b></p> <p><b>mode</b> - possible values:  <i>all</i> (default), <i>skip</i> - skip processing of older data (affects only newly created items).</p> <p><b>output</b> - an optional output formatting template. The <b>\0</b> escape sequence is replaced with the matched part of text (from the first character where match begins until the</p>	<p>The item must be configured as an <b>active check</b>.  Log rotation is based on the last modification time of files.</p> <p>Note that logrt is designed to work with one currently active log file, with several other matching inactive files rotated. If, for example, a directory has many active log files, a separate logrt item should be created for each one. Otherwise if one logrt item picks up too many files it may lead to exhausted memory and a crash of monitoring.</p> <p>If <b>output</b> is left empty - the whole line containing the matched text is returned. Note that all global regular expression types except 'Result is TRUE' always return the whole matched line and the <b>output</b> parameter is ignored.</p> <p>Content extraction using the <b>output</b> parameter takes place</p>

## Documentation of logrt file

But today we are talking just about a log. So, in the log file monitoring you can see quite a lot of parameters which we can use but it's mostly optional, not mandatory.

log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>]			
Log file monitoring.	Log	<p><b>file</b> - full path and name of log file</p> <p><b>regexp</b> - regular expression<sup>4</sup> describing the required pattern</p> <p><b>encoding</b> - code page <b>identifier</b></p> <p><b>maxlines</b> - maximum number of new lines per second the agent will send to Zabbix server or proxy. This parameter overrides the value of 'MaxLinesPerSecond' in <b>zabbix_agentd.conf</b></p> <p><b>mode</b> - possible values:  <i>all</i> (default), <i>skip</i> - skip processing of older data (affects only newly created items).</p> <p><b>output</b> - an optional output formatting template. The <b>\0</b> escape sequence is replaced with the matched part of text (from the first character where match begins until the character where match ends) while an <b>\N</b> (where N=1...9) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups).</p> <p><b>maxdelay</b> - maximum delay in seconds. Type: float. Values: 0 - (default) never ignore log file lines; &gt; 0.0 - ignore older lines in order to get the most recent lines analyzed within "maxdelay" seconds. Read the <b>maxdelay</b> notes before using it!</p>	<p>The item must be configured as an <b>active check</b>.  If file is missing or permissions do not allow access, item turns unsupported.</p> <p>If <b>output</b> is left empty - the whole line containing the matched text is returned. Note that all global regular expression types except 'Result is TRUE' always return the whole matched line and the <b>output</b> parameter is ignored.</p> <p>Content extraction using the <b>output</b> parameter takes place on the agent.</p> <p>Examples:  ⇒ log[/var/log/syslog]  ⇒ log[/var/log/syslog,error]  ⇒ log[/home/zabbix/logs/logfile,,,100]</p> <p>Using <b>output</b> parameter for extracting a number from log record:  ⇒ log[/app1/app.log,"task run [0-9.]+ sec, processed ([0-9.]+ records, [0-9.]+ errors",,,,1) → will match a log record "2015-11-13 10:08:26 task run 6.08 sec, processed 6080 records, 0 errors" and send only '6080' to server. Because a numeric value is being sent, the "Type of information" for this item can be set to "Numeric (unsigned)" and the value can be used in graphs, triggers etc.</p>

## Parameters of log file

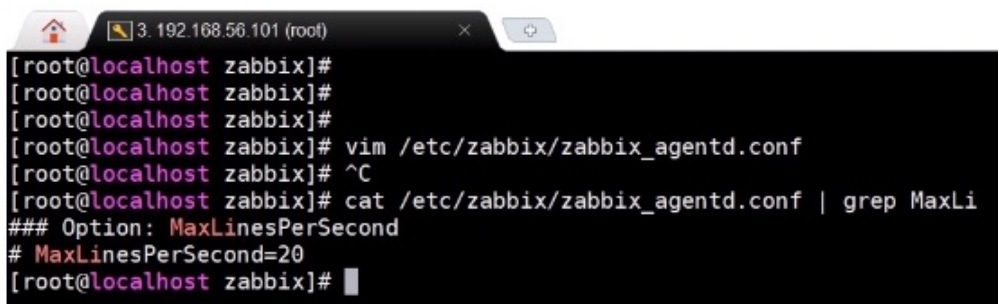
The only mandatory parameter is the '**file**' because we need to specify: *"file is the full path and name of the log file that you want to monitor"*. So it will be, for example, **/var/log/httpd/error\_log**.



The next one is the **'regular expression' (<regexp>)** parameter. As you see it is inside the brackets which means that it is optional. But if we want to use it then it will be a *"regular expression describing the required pattern"*. So, we will be filtering something in our log file.

**'Encoding' (<encoding>)** — pretty straightforward.

**'Maxlines'(<maxlines>)** — "maximum number of new lines per second the agent will send to the Zabbix server or proxy. This parameter overrides the value of **'MaxLinesPerSecond'** in **'zabbix\_agentd.conf'**. So, this is already mentioned in the agent config file and the default value is **'20'**.



```
[root@localhost zabbix]#  
[root@localhost zabbix]#  
[root@localhost zabbix]#  
[root@localhost zabbix]# vim /etc/zabbix/zabbix_agentd.conf  
[root@localhost zabbix]# ^C  
[root@localhost zabbix]# cat /etc/zabbix/zabbix_agentd.conf | grep MaxLi  
### Option: MaxLinesPerSecond  
# MaxLinesPerSecond=20  
[root@localhost zabbix]#
```

'Maxlines' parameter in the agent configuration file

So each second the agent can send 20 lines of the log to the Zabbix server.

**Note.** *I might be wrong on this one — it is not documented — but I'm almost sure that it was multiplied by 4, and that will be the amount of the lines that the agent will read per second from the log file. So, each second it can read 88 zero lines and it will send only 20 lines to the Zabbix server.*

Usually, there is no need to change this parameter, there is no need to use **'maxlines'** in the item **'Key'** parameters and even more, to change it in the Zabbix agent config file. Default is absolutely okay.

Next one — **'mode' (<mode>)**. This is pretty serious. There are only two options — **'all'** and **'skip'**. Usually, in 99% of the cases you will have to use the **'skip'** because, let's say you have some kind of error log, which is of course already existing on your virtual machine or physical server for months or even years and it already has a lot of lines, a lot of error messages.

Then, you decide to start using Zabbix — you've installed the agent, configured log monitoring items. And if you do not specify this mode parameter to the **'skip'**, the first time agent will check the log file it will check for all the file from the beginning.

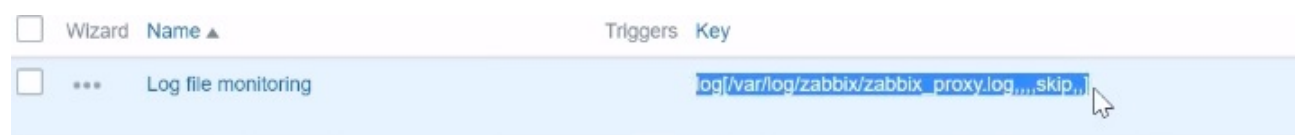
So, most likely, you will receive a lot of lines that might be up to a year old. And if you have some triggers configured then you will also receive a lot of false-positive trigger alarms about the lines that actually are from the past. When you specify the parameter '**skip**', it means that the agent will ignore all the log lines from the history and start monitoring only from this moment.

Next one – '**output**' (<output>). It allows you to extract some specific lines from the matched lines in the log file. We will see that in the examples in my demo items.

'**Maxdelay**' (<maxdelay>) is for ignoring the lines from the log file that are older than X from the moment when we are reading. Again, I don't know any use case why you would want to use this – usually, this is simply ignored.

## Log file monitoring

Let's get to the front end. I have multiple things here. First of all, this one – **Log file monitoring**:



First log file monitoring item

Let's check the parameters I have – I have the location of the file. And in all of these four examples, I am monitoring my **/zabbix\_proxy.log**. Here I have even increased debug level so it is producing quite a lot of new lines per second.

Since I don't have any additional parameter here except of the '**skip**', I just specify location of the log. I don't do any filtering based on the regular expression, I don't extract any output. Just '**skip**'. So, this item will collect absolutely all new incoming lines to the log file. Is it reasonable? Not really.

Zabbix is not a syslog server. While it is possible to collect absolutely all data coming inside the log file, the amount of new lines it produces each second makes it unreasonable.

```

2019-06-04 16:28:13 23101:20190604:075610.422 zbx_setproctitle() title:'icmp pinger #1 [getting values]'
2019-06-04 16:28:13 23094:20190604:075610.422 zbx_setproctitle() title:'unreachable poller #1 [got 0 values in 0.000261 sec, idle 5 sec]'
2019-06-04 16:28:13 23094:20190604:075610.422 End of get_values():0
2019-06-04 16:28:13 23094:20190604:075610.422 End of DCoonfig_get_poller_nextcheck():-1
2019-06-04 16:28:13 23094:20190604:075610.422 In DCoonfig_get_poller_nextcheck() poller_type:1
2019-06-04 16:28:13 23094:20190604:075610.422 End of DCoonfig_get_poller_items():0
2019-06-04 16:28:13 23094:20190604:075610.422 In DCoonfig_get_poller_items() poller_type:1
2019-06-04 16:28:13 23094:20190604:075610.422 In get_values()
2019-06-04 16:28:13 23094:20190604:075610.422 zbx_setproctitle() title:'unreachable poller #1 [got 0 values in 0.000230 sec, getting values]'
2019-06-04 16:28:13 23080:20190604:075610.390 zbx_setproctitle() title:'self-monitoring [processed data in 0.000178 sec, idle 1 sec]'
2019-06-04 16:28:13 23080:20190604:075610.390 End of collect_selfmon_stats()
2019-06-04 16:28:13 23080:20190604:075610.390 In collect_selfmon_stats()
2019-06-04 16:28:13 23080:20190604:075610.390 zbx_setproctitle() title:'self-monitoring [processing data]'
2019-06-04 16:28:13 23072:20190604:075610.144 zbx_setproctitle() title:'data sender [sent 0 values in 0.002559 sec, idle 1 sec]'
2019-06-04 16:28:13 23072:20190604:075610.144 End of proxy_data_sender():FAIL more:0 flags:0x8000
2019-06-04 16:28:13 23072:20190604:075610.144 cannot send proxy data to server at "127.0.0.1": proxy "Sabbix proxy" not found
2019-06-04 16:28:13 23072:20190604:075610.144 End of put_data_to_server():FAIL
2019-06-04 16:28:13 23090:20190604:075610.442 End of DCoonfig_get_poller_nextcheck():-1
2019-06-04 16:28:13 23090:20190604:075610.442 In DCoonfig_get_poller_nextcheck() poller_type:0

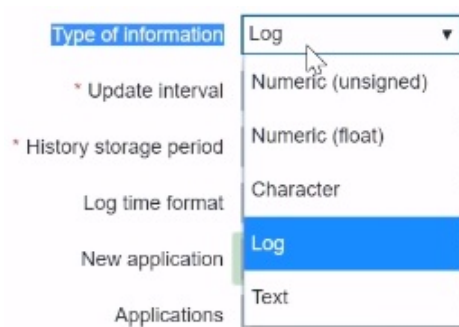
```

Log file lines of the first item

It is absolutely not wise to collect everything from this log in our database because of performance reasons — it will take a lot of the disk space and it is simply not reasonable.

So, what are other options? Let's go back to our **Items**.

**Note.** In the log file monitoring that collects everything, the “Type of information” will be “Log”. Don't mess up, it's not “Numeric (unsigned)”, it's not “Numeric (float)”. You can use it as a “Text” but normally the type of information is “Log”.



Type of information in the first log file monitoring

And **'Update interval'** — remember, it is best practice to use **one second (1s)**.

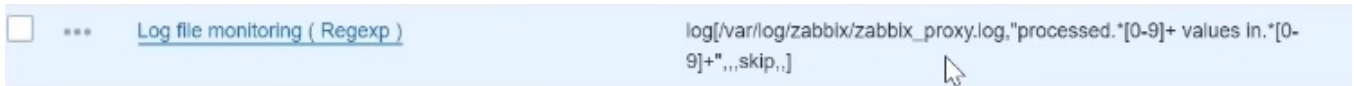
\* Update interval

Recommended update interval



You might be thinking — “I don’t want to put a big stress on my Zabbix server, I don’t want to check it each second, I will be doing that each five minutes”. But remember, during those five minutes the log continues to grow and if it produced some new lines that should be captured by Zabbix every five seconds, after five minutes there will be already a big chunk of data to send to the Zabbix server. Instead of receiving two or three lines each five seconds you’ll be receiving 200 log file lines every five minutes. That single moment, those 200 incoming lines can and most likely will affect the performance of your Zabbix server. So remember – one second update interval.

In the second item, I am using a regular expression.



Second log file monitoring item

Let’s check the monitoring latest data of the **Log file monitoring (Regexp)**.

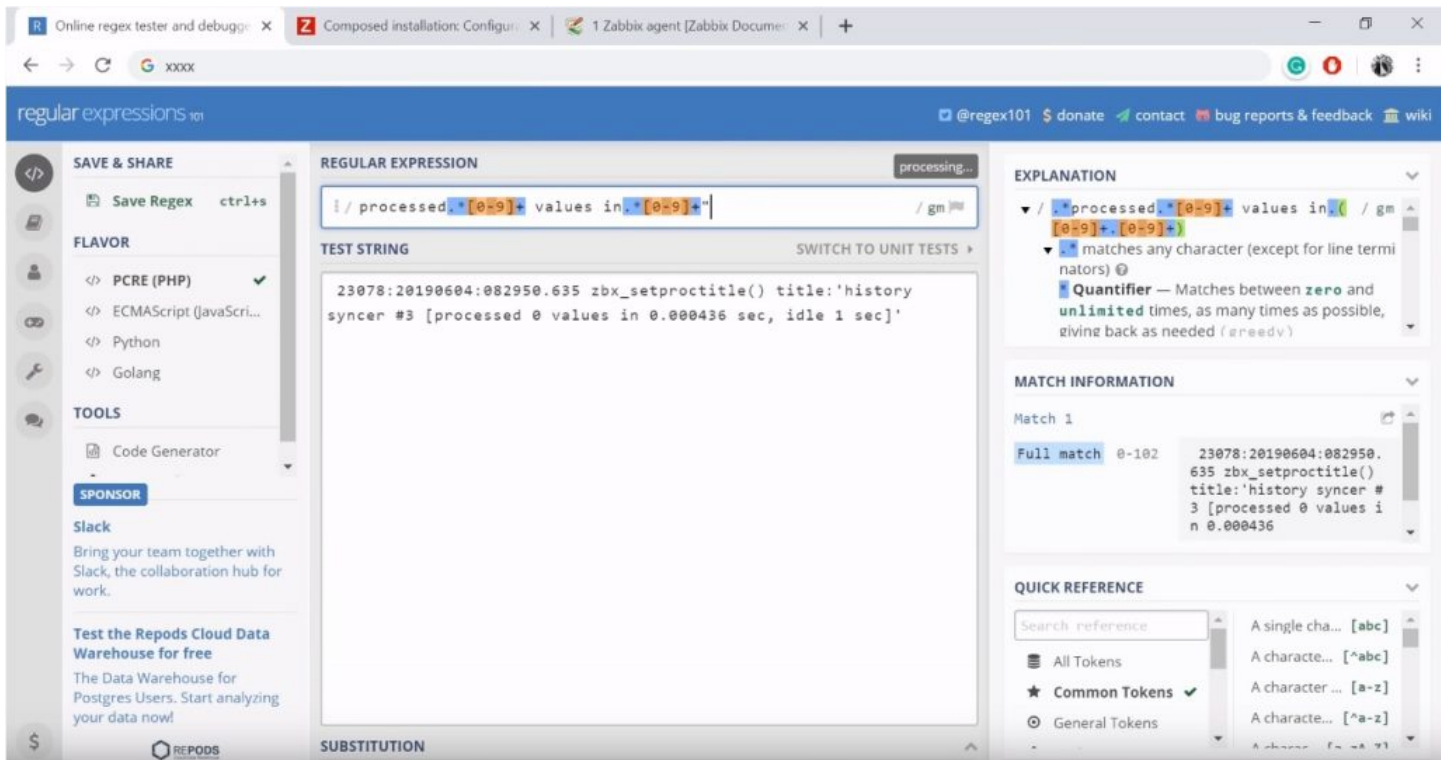
2019-06-04 16:30:01	23077:20190604:082950.655 zbx_setproctitle() title:'history syncer #2 [processed 0 values in 0.000444 sec, syncing history]'
2019-06-04 16:30:01	23078:20190604:082950.635 zbx_setproctitle() title:'history syncer #3 [processed 0 values in 0.000436 sec, idle 1 sec]'
2019-06-04 16:30:01	23078:20190604:082950.634 zbx_setproctitle() title:'history syncer #3 [processed 0 values in 0.000243 sec, syncing history]'
2019-06-04 16:30:01	23076:20190604:082950.508 zbx_setproctitle() title:'history syncer #1 [processed 0 values in 0.000263 sec, idle 1 sec]'
2019-06-04 16:30:01	23076:20190604:082950.508 zbx_setproctitle() title:'history syncer #1 [processed 0 values in 0.000243 sec, syncing history]'
2019-06-04 16:30:01	23079:20190604:082950.263 zbx_setproctitle() title:'history syncer #4 [processed 0 values in 0.000164 sec, idle 1 sec]'
2019-06-04 16:30:01	23079:20190604:082950.263 zbx_setproctitle() title:'history syncer #4 [processed 0 values in 0.000157 sec, syncing history]'
2019-06-04 16:30:01	23077:20190604:082949.655 zbx_setproctitle() title:'history syncer #2 [processed 0 values in 0.000444 sec, idle 1 sec]'
2019-06-04 16:30:01	23077:20190604:082949.654 zbx_setproctitle() title:'history syncer #2 [processed 0 values in 0.000193 sec, syncing history]'
2019-06-04 16:30:01	23078:20190604:082949.630 zbx_setproctitle() title:'history syncer #3 [processed 0 values in 0.000243 sec, idle 1 sec]'

Latest data of the second log file item

I will copy-paste this line:

```
23078:20190604:082950.635 zbx_setproctitle() title: 'history syncer #3 [processed 0 values in 0.000436 sec, idle 1 sec]'
```

If you are configuring some kind of the regular expressions in the front end or any other places online, i.e. [Regex tester](#), just type a string and then you can try your regular expression and see if it works or not.



Regex tester

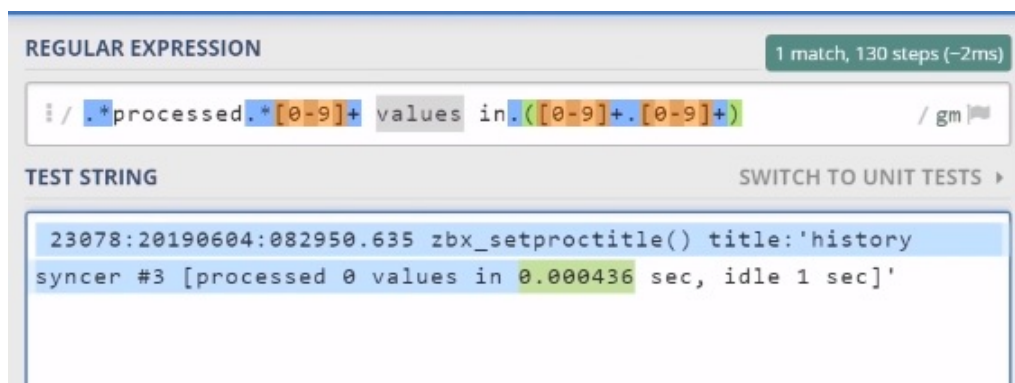
Let's go back to the **front end > items > regular expression** filtering item. The only additional parameter I have here is still the location of my proxy log:

\* Key `log[/va/ log/zabbix/zabbix_proxy.log,"processed.*[0-9]+ values in.*[0-9]+",,skip,,]` Select

Parameters of the second log file item

I still have the **'skip'**. There is quite a lot of commas because I'm skipping those optional parameters. And just one new single parameter — the **'regular expression'**. I can test it in the **Regex tester**. Again, just copy-paste, delete the quotes and there we go.

So, I am capturing and will be getting all new lines from the **Zabbix proxy log lines** that will contain a string **'processed'**, then there will be anything, then there might be any digit because of the **plus** (an unlimited amount), **values in**, anything, and again any digit (an unlimited amount).



Any line that will contain this string will be captured in the **Zabbix server** by the **Zabbix agent**. You can see that if in the **Latest data** of the **first item** I had quite a lot of values (since this is capturing absolutely everything I have in the log), on **the next one** where I use my latest regular expression I am capturing just a single line:

Timestamp	Local time	Value
2019-06-04 16:31:42	23076:20190604:083350.897	zbx_setproctitle() title:'history syncer #1 [processed 0 values in 0.000242 sec, idle 1 sec]'
2019-06-04 16:31:42	23076:20190604:083350.897	zbx_setproctitle() title:'history syncer #1 [processed 0 values in 0.000278 sec, syncing history]'
2019-06-04 16:31:42	23079:20190604:083350.652	zbx_setproctitle() title:'history syncer #4 [processed 0 values in 0.000366 sec, idle 1 sec]'
2019-06-04 16:31:42	23079:20190604:083350.651	zbx_setproctitle() title:'history syncer #4 [processed 0 values in 0.000465 sec, syncing history]'
2019-06-04 16:31:42	23077:20190604:083350.068	zbx_setproctitle() title:'history syncer #2 [processed 0 values in 0.000164 sec, idle 1 sec]'
2019-06-04 16:31:42	23077:20190604:083350.068	zbx_setproctitle() title:'history syncer #2 [processed 0 values in 0.000078 sec, syncing history]'
2019-06-04 16:31:42	23078:20190604:083350.009	zbx_setproctitle() title:'history syncer #3 [processed 0 values in 0.000206 sec, idle 1 sec]'
2019-06-04 16:31:42	23078:20190604:083350.009	zbx_setproctitle() title:'history syncer #3 [processed 0 values in 0.000321 sec, syncing history]'
2019-06-04 16:31:42	23076:20190604:083349.897	zbx_setproctitle() title:'history syncer #1 [processed 0 values in 0.000278 sec, idle 1 sec]'

Latest data of the second log file item

Now, I am filtering. I am not capturing all incoming log lines, I am filtering based on the regular expression.

Back to the **Items**. The next one is '**Log file monitoring (regular expression and output)**'.

<input type="checkbox"/>	***	Log file monitoring ( Regexp and Output )	log[/var/log/zabbix/zabbix_proxy.log,"processed."[0-9]+ values in.([0-9]+.[0-9]+)"",skip,1,]
--------------------------	-----	---	--

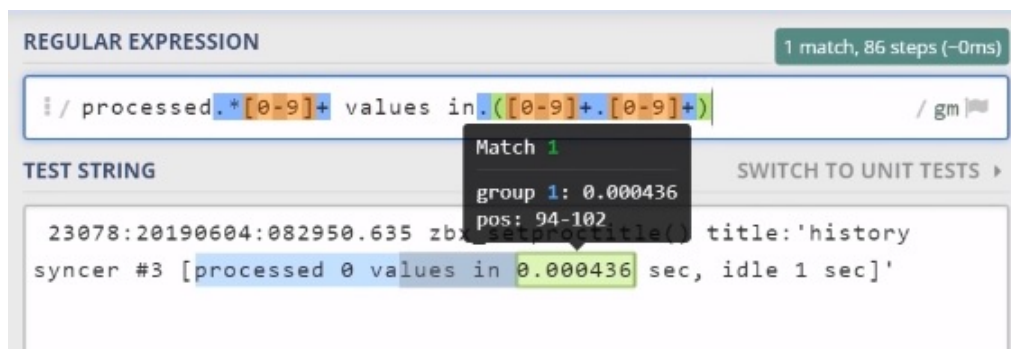
Third log file item

In this case, I am using another output parameter which specifies the capturing groups of my regular expression parameter. Again, I have the same line but in the '**Key**' the regular expression is slightly different in this case.

\* Key

Parameters of the third log file item

In the Regex tester '**processed**' and '**values**' are the same and the new one is '**capturing groups**'. So, I am capturing the same lines but I am also adding in a '**capturing group**' this digit because I wanna extract the amount of time spent for syncing the history:



Specifying the capturing group

This means that in my **Item parameters**, I specify which capturing group I want to display in the output (**\1**).

If I added **\0**, in the latest data it would display everything that matches from my regular expression, meaning this part:

```
processed 0 values in 0.000436
```

Since I've displayed here **\1** (the first capturing group), I am getting this one:

```
0.000436
```

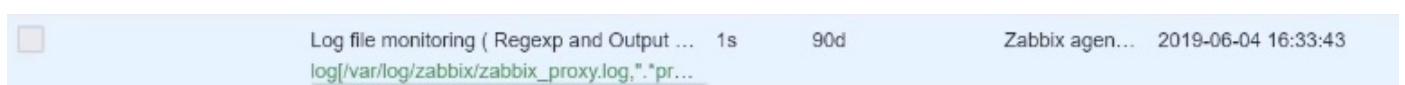
And we can verify that in the monitoring latest data:



Last value of the third log file item

**Note.** Because there is 0.000 in this one, I still have to use the **'Text'** or the **'Log'** type of information because it rounds to the zero. But normally you can extract integers and even draw the graphs from your log file monitoring items.

The last one:



Fourth log file item

First of all, in the parameters we have the same **log Zabbix agent (active)**, the **location of the proxy log file**, the **regular expression** that we are capturing, **'skip'** and **capturing group '0'**.

\* Key

\* Key

Parameters of the fourth log file item

So, the last one will capture everything from my regular expression:

Timestamp	Local time	Value
2019-06-04 16:34:28	2019-06-04 15:40:50	23078:20190604:084050.794 zbx_setproctitle() title:'history syncer #3 [processed 0 values in 0.000199
2019-06-04 16:34:28	2019-06-04 15:40:50	23076:20190604:084050.769 zbx_setproctitle() title:'history syncer #1 [processed 0 values in 0.000210
2019-06-04 16:34:28	2019-06-04 15:40:50	23076:20190604:084050.768 zbx_setproctitle() title:'history syncer #1 [processed 0 values in 0.000195
2019-06-04 16:34:28	2019-06-04 15:40:50	23079:20190604:084050.263 zbx_setproctitle() title:'history syncer #4 [processed 0 values in 0.000206
2019-06-04 16:34:28	2019-06-04 15:40:50	23079:20190604:084050.263 zbx_setproctitle() title:'history syncer #4 [processed 0 values in 0.000247
2019-06-04 16:34:27	2019-06-04 15:40:54	23077:20190604:084054.942 zbx_setproctitle() title:'history syncer #2 [processed 0 values in 0.000508
2019-06-04 16:34:27	2019-06-04 15:40:54	23077:20190604:084054.942 zbx_setproctitle() title:'history syncer #2 [processed 0 values in 0.000272
2019-06-04 16:34:27	2019-06-04 15:40:54	23078:20190604:084054.862 zbx_setproctitle() title:'history syncer #3 [processed 0 values in 0.000266
2019-06-04 16:34:27	2019-06-04 15:40:54	23078:20190604:084054.862 zbx_setproctitle() title:'history syncer #3 [processed 0 values in 0.000254
2019-06-04 16:34:27	2019-06-04 15:40:54	23076:20190604:084054.773 zbx_setproctitle() title:'history syncer #1 [processed 0 values in 0.000252
2019-06-04 16:34:27	2019-06-04 15:40:54	23076:20190604:084054.772 zbx_setproctitle() title:'history syncer #1 [processed 0 values in 0.000225

Latest data of the fourth log item

But the additional thing here is that I have added the **'Log time format'**:

Log time format

So, it is possible to not only capture the log line itself but also extract the timestamp from the log.

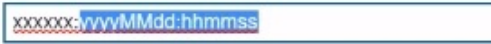
```
23077:20190604:092831.551 In zbx_sync_history_cache() history_num:0
23077:20190604:092831.551 End of zbx_sync_history_cache()
23077:20190604:092831.551 zbx_setproctitle() title:'history syncer #2 [proce
```

Log line in the agent configuration file

Here you see that this first part (23077) is the **process ID** which wrote the mentioned log line. The next one (:) is just a separator, then there is a year (2019), month (06), date (04), separator (:), hours (09), minutes (28), seconds (31) and nanoseconds (551). Here it is not relevant for us.



So if we want to extract it we also need to use the placeholders. You can check the example here:



Placeholders and the format of the timestamp

The format is years (yyyy), months (MM), days (dd), hours (hh), minutes (mm), seconds (ss). Everything else serves as a placeholder. So, the first **five** digits and the separator in the process ID are not the timestamp so I need to use a placeholder (xxxxxx) till I get to the actual beginning of the year. Then, I use “yyyy”, “MM”, “dd”, a separator (:), “hh”, “mm”, “ss”.

That is all you have to do. And then, in the monitoring latest data I see additional column — the **Local time**. You can see there is a **Timestamp** of when the value was collected and the actual **Local time** which is the time of the log line written to the log:

Timestamp	Local time	Value
2019-06-04 16:36:15	2019-06-04 15:45:11	23076:20190604:084511.201 zbx_se
2019-06-04 16:36:15	2019-06-04 15:45:11	23076:20190604:084511.201 zbx_se
2019-06-04 16:36:15	2019-06-04 15:45:10	23079:20190604:084510.603 zbx_se
2019-06-04 16:36:15	2019-06-04 15:45:10	23079:20190604:084510.603 zbx_se
2019-06-04 16:36:14	2019-06-04 15:45:05	23077:20190604:084505.306 zbx_se
2019-06-04 16:36:14	2019-06-04 15:45:05	23077:20190604:084505.305 zbx_se
2019-06-04 16:36:14	2019-06-04 15:45:05	23078:20190604:084505.231 zbx_se
2019-06-04 16:36:14	2019-06-04 15:45:05	23078:20190604:084505.231 zbx_se
2019-06-04 16:36:14	2019-06-04 15:45:05	23076:20190604:084505.186 zbx_se
2019-06-04 16:36:14	2019-06-04 15:45:05	23076:20190604:084505.186 zbx_se
2019-06-04 16:36:14	2019-06-04 15:45:04	23079:20190604:084504.593 zbx_se

Local time column in Latest data of the fourth log file

And you can see that in the previous items I have not used it and that is why the log **Local time** column is empty:



Timestamp	Local time	Value
2019-06-04 16:36:41		23076:20190604:084553.270 zbx_setprocti
2019-06-04 16:36:41		23076:20190604:084553.269 zbx_setprocti
2019-06-04 16:36:41		23079:20190604:084552.691 zbx_setprocti
2019-06-04 16:36:41		23079:20190604:084552.691 zbx_setprocti
2019-06-04 16:36:41		23077:20190604:084552.368 zbx_setprocti
2019-06-04 16:36:41		23077:20190604:084552.368 zbx_setprocti
2019-06-04 16:36:41		23078:20190604:084552.311 zbx_setprocti
2019-06-04 16:36:41		23078:20190604:084552.311 zbx_setprocti
2019-06-04 16:36:41		23076:20190604:084552.269 zbx_setprocti
2019-06-04 16:36:41		23076:20190604:084552.269 zbx_setprocti

Local time column in Latest data of the previous log file

So, long story short – that would be a quick review of how you can monitor your logs in **Windows** and **Linux** environments through **Zabbix agents**. Just remember that all the monitoring is done by default by the **Zabbix user**. So make sure that the **Zabbix user** can read your log file and that there are at least read permissions. Also, if you are using a **Linux** machine with **SELinux** turned on, then by default 99% it will not allow the **Zabbix user** to read the log file, so you will have to make an exception in **SELinux** policies.

And that's about it. I hope you liked this article. Be sure to check out other videos in the [Zabbix series](#).

Tags: [log file](#), [log monitoring](#), [logmonitoring](#), [Zabbix agent](#)



**Author: Dmitry Lambert**

Zabbix Certified Expert & Trainer [View all posts by Dmitry Lambert](#)



Dmitry Lambert / September 26, 2019 / How To, Technical / log file, log monitoring, logmonitoring, Zabbix agent

## 2 thoughts on “Zabbix Log File Monitoring”



**decebal traian**

February 28, 2020 at 18:38