Configuring NXLog to Send Multi-Line Log Files in Nagios Log Server 2024R2

Purpose

This document describes how you can configure NXLog to send multi-line logs to Nagios Log Server.

Requirements

It is assumed you have already installed NXLog on your Windows server and configured it to send logs to Nagios Log Server. This is covered in the following article:

Monitoring A New Log Source

Multi-Line Logs

What is a multi-line log? This is when the data that encompasses the entire event is spread across multiple lines in the log file. For example:

```
2016-10-07T09:59:45.806 INFO  SOME_SERVER This is the header row
This is the second line
This is the third line
```

Normally when you configure NXLog to send a custom log file to Nagios Log Server it is sent on a line-by-line basis. This can make it complicated to review the logs on NLS as it will be displayed as multiple events.

In the example above, you can see the first line starts with the date time format **ISO8601**. Every entry recorded in this log file will always have this first line formatted this way. NXLog can be configured to identify the ISO8601 string and then send the entire data to NLS as a multi-line log. NLS will also have an extra configuration input added to handle the incoming multi-line data.

Scenario Details

To properly demonstrate how this works, the following KB article will use the log file `C:\Logs\Important_File.log` to send to Nagios Log Server.

To simulate a multiple line log entry being added to the log, first create a file here:

```
C:\Logs\test.log
```

…with the following contents:

```
2016-10-07T09:59:45.806 INFO  SOME_SERVER This is the header row
This is the second line
This is the third line
```

The following command in Command Prompt will append the data to a new file `C:\Logs\Important_File.log`:

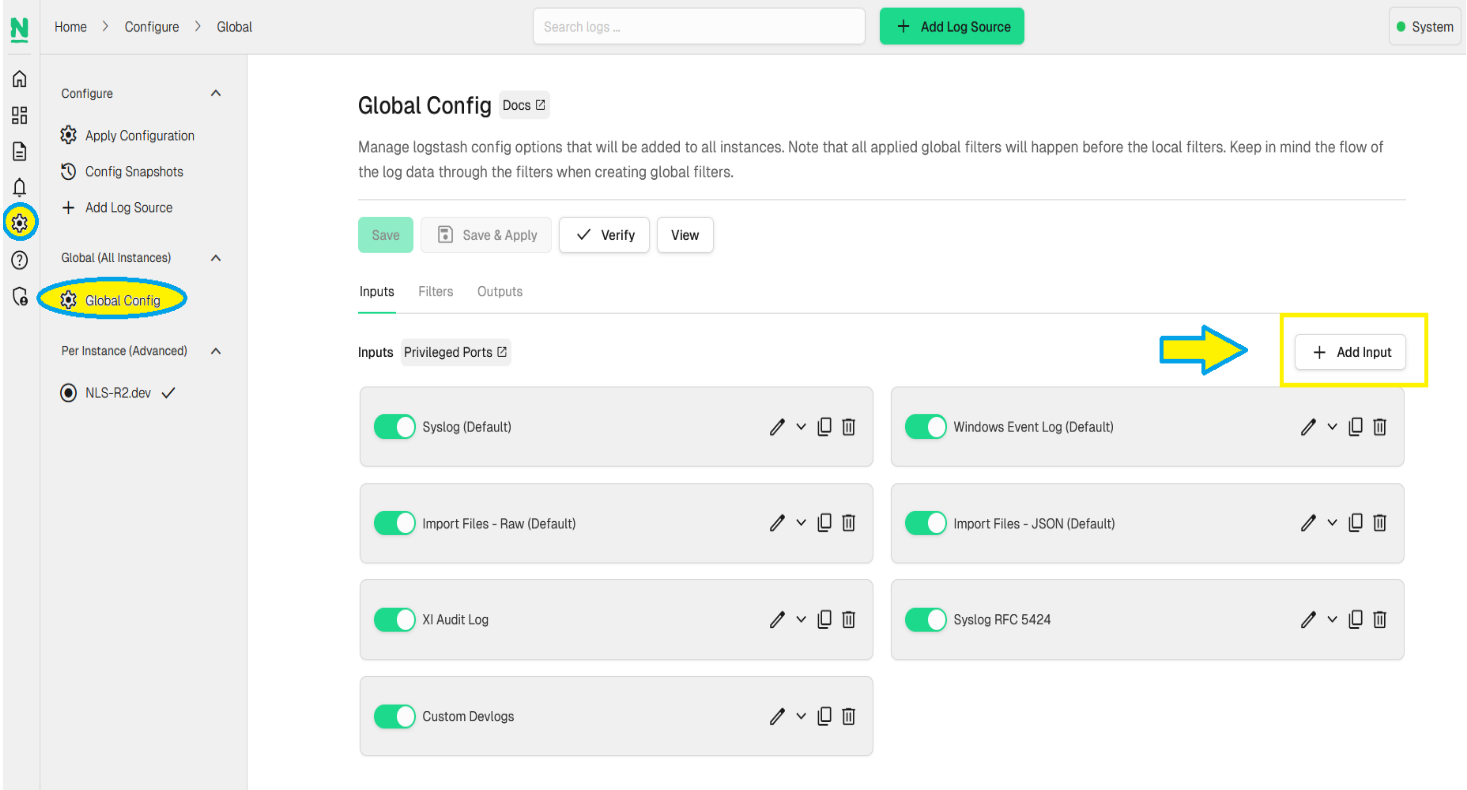
```
more C:\Logs\test.log >> C:\Logs\Important_File.log
```

Using those steps you will be able to successfully follow this KB article and confirm the functionality works. Every time the **`more`** command above is executed, a new multi-line entry is added to the `C:\Logs\Important_File.log` log file. Even though technically the date is incorrect this will not matter, it is simply an example.

Create Input On Nagios Log Server

The first step is to create a new Input to identify multi-line logs. Login to one of your Nagios Log Server instances as an Admin user click the **Configure** icon on the lefthand navigation bar: ⚙

In the left pane under **Global (All Instances)** click **Global Config**, then click “+Add Input”

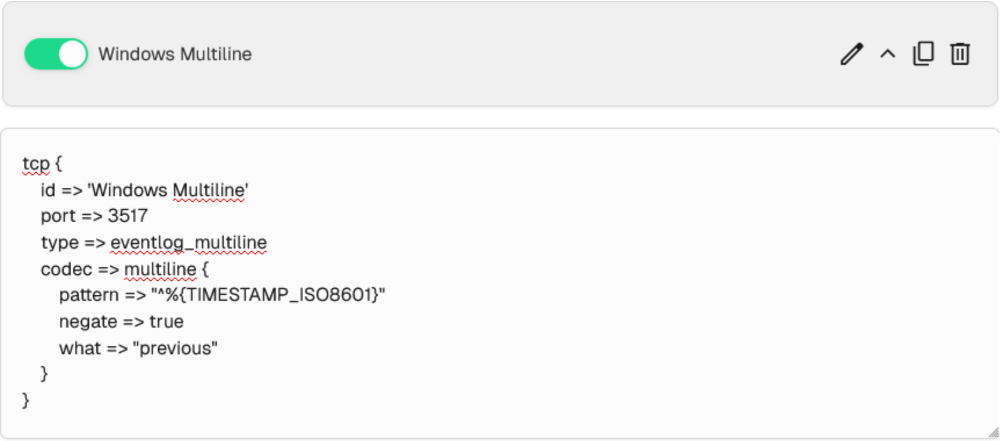


Manage logstash config options that will be added to all instances. Note that all applied global filters will happen before the local filters. Keep in mind the flow of the log data through the filters when creating global filters.

Give your new Input a meaningful name, and add the following to the code section:

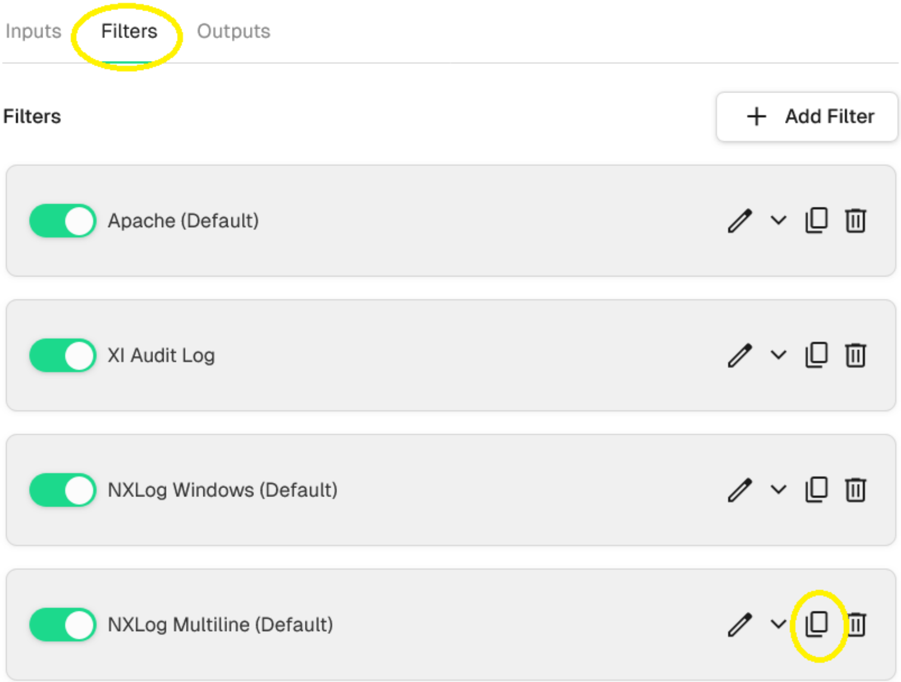
```
tcp {
    id => 'Windows Multiline'
    port => 3517
    type => eventlog_multiline
    codec => multiline {
        pattern => "^%{TIMESTAMP_ISO8601}"
        negate => true
        what => "previous"
    }
}
```

The input will look like this:



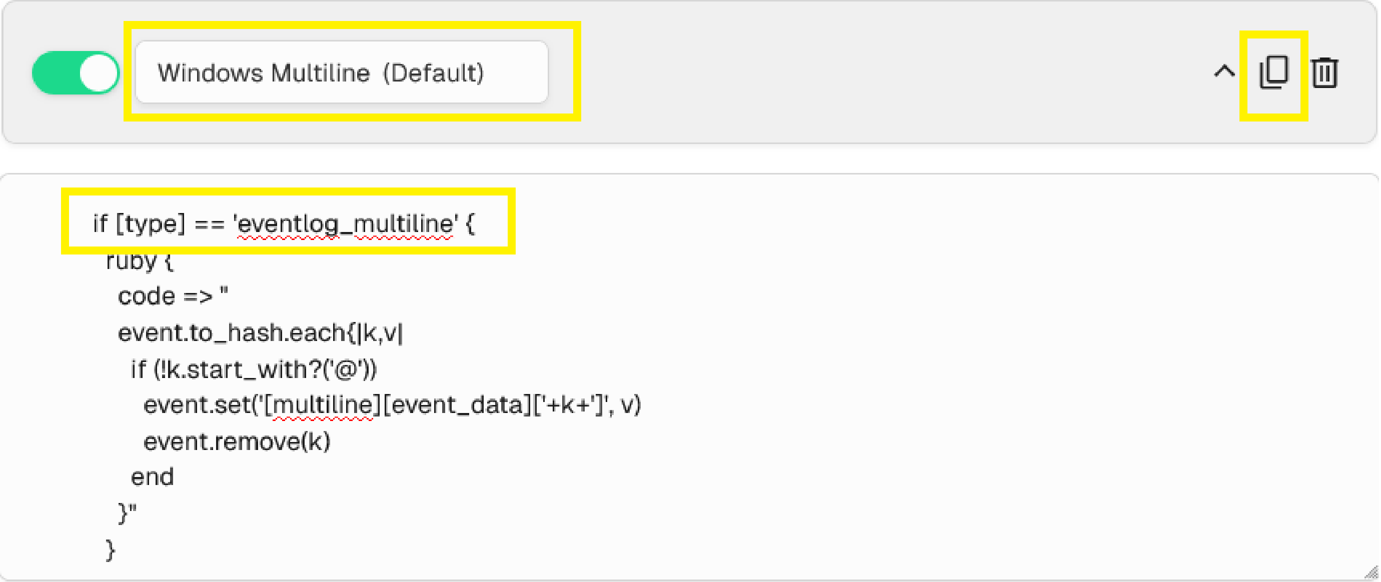
```
tcp {
   id => 'Windows Multiline'
   port => 3517
   type => eventlog_multiline
   codec => multiline {
      pattern => "^%{TIMESTAMP_ISO8601}"
      negate => true
      what => "previous"
   }
}
```

Next click **Filters**, and make a copy of the **NXLog Multiline (Default)** filter:



Change the type check in the first line to:

```
if [type] == eventlog_multiline{
```



```
if [type] == 'eventlog_multiline' {
 ruby {
  code => "
  event.to_hash.each{|k,v|
   if (!k.start_with?('@'))
    event.set('[multiline][event_data]['+k+']', v)
    event.remove(k)
   end
  }"
 }
```

Click the **Save** button.

Then click the **Verify** button above to ensure this is a valid configuration.

Once the verification process is OK, in the left pane under **Configure** click **Apply Configuration**.

Click the **Apply** button.

Click **Yes, Apply Now**.

Once this process has finished you can continue onto the next step. You will return to Nagios Log Server once NXLog has been configured.

Configure NXLog

Login to your Windows server and open the file *C:\Program Files (x86)\nxlog\conf\**nxlog.conf*** in Notepad.

Add the following to the end of the file:

```
  <Extension multiline_header>
     Module xm_multiline
     HeaderLine /^\d\d\d\d-\d\d-\d\d \d\d:\d\d:\d\d,\d\d\d\s+/
  </Extension>

  <Input Important_File>
     Module im_file
     InputType multiline_header
     Exec $type = 'multiline';
     File 'C:\Logs\Important_File.log'
     SavePos TRUE
     Exec $Message = $raw_event;
  </Input>
```

The location that you place the content in the file is not important.

Add a output for the multiline file:

```
<Output out_multiline>
    Module om_tcp
    Host 192.168.56.127
    Port 3517

    Exec $raw_event = $raw_event + $MessageSourceAddress;
    Exec $raw_event = $raw_event + $source_ip;
    Exec  $tmpmessage = $Message; delete($Message); rename_field("tmpmessage","message");
</Output>
```

You will also need to add a **route** section to include the `Important_File` input. Find this section:

```
<Route 1>
   Path internal, file1, eventlog => out
</Route>
```

Add below it:

```
<Route 2>
    Path Important_File  => out_multiline
</Route>
```

Save the changes you just made, then restart NXlog. To do so:

Open *services.msc* and **restart** the **nxlog** service

Or, from the command prompt, you can issue the following commands:

```
net stop nxlog
net start nxlog
```

What does all of that mean?

```
<Extension multiline_header>
   Module xm_multiline
   HeaderLine /^\d\d\d\d-\d\d-\d\d \d\d:\d\d:\d\d\.\d\d\d\s+/
</Extension>
```

The line Module `xm_multiline` tells NXLog to use the "Multi-line message parser" module.

This `HeaderLine` tells NXLog that the following sting format is first line of a log entry:

```
HeaderLine /^\d\d\d\d-\d\d-\d\d \d\d:\d\d:\d\d,\d\d\d\s+/
```

- This is a regular expression (regex)
- The first **/** is the beginning of the regex
- The **^** means that the line begins with this pattern
- Every **\d** represents a digit
- The **\.** means to use a period. Periods in regexes must be escaped with a ****
- The **-** and **:** are all characters present at that location in the string
- The **\s** is a whitespace character
- The + means one or more occurrences of the whitespace character
- The last **/** is the end of the regex

Basically it's saying this is the format of the string which needs to be matched:

```
dddd-dd-dd dd:dd:dd.ddd
```

Remember the example we have, you can see the line begins with format:

```
2016-10-07 09:59:45.806 INFO  SOME_SERVER This is the header row
```

In the next part of the configuration:

```
<Input Important_File>
   Module im_file
   InputType multiline_header
   Exec $type = 'multiline';
   File 'C:\Logs\Important_File.log'
   SavePos TRUE
   Exec $Message = $raw_event;
</Input>
```

- The first line ***<Input Important_File>*** identifies an input that is labeled ***Important_File***
- The line ***Module im_file*** tells NXLog to use the "File" module
- The line **InputType multiline_header** tells NXLog to use the multiline_header extension previously defined, this is how it will identify the beginning of a new entry
- The line ***Exec $type = 'multiline’;*** tells NXLog to send the entry to NLS with the type defined as 'multiline'
    - This is purely for ease of identification in NLS, it can be anything you want it to be
    - It can make querying easier in NLS later on
- The line ***File 'C:\Logs\Important_File.log'*** tells NXLog the location of the file it will be watching
- The line ***SavePos TRUE*** tells NXLog to remember where it is up to in the log file when the nxlog service is stopped *(this prevents the entire log being re-sent to NLS)*
- The line ***Exec $Message = $raw_event;*** is how NXLog sends the entire message to NLS

The final part of the configuration simply tells NXLog to use the Important_File input that you defined.

```
<Route 3>
   Path internal, file1, eventlog, Important_File => out
</Route>
```

Test

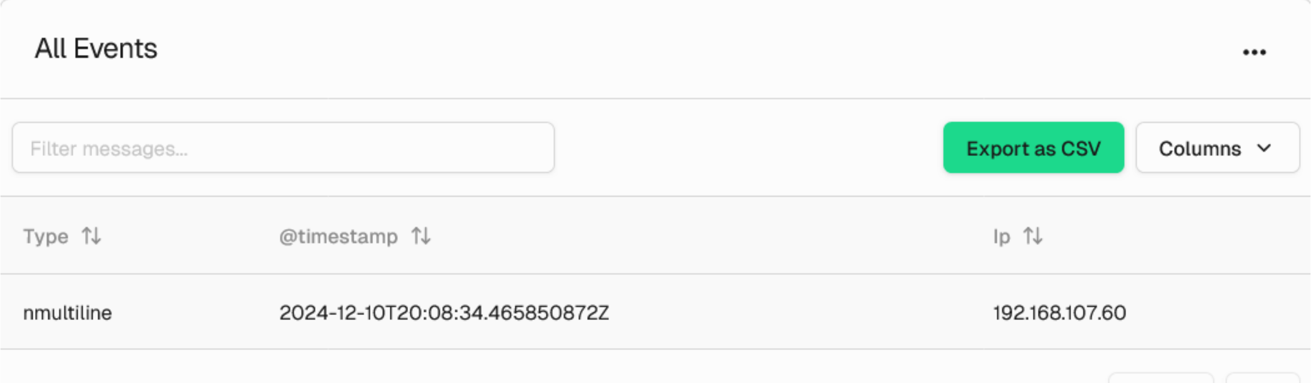
Now you can test that it is working by executing the following command on your Windows machine:

```
more C:\Logs\test.log >> C:\Logs\Important_File.log
```

On your Nagios Log Server:

- Login to Nagios Log Server
- Open the Default dashboard
- Perform a search for `multiline`

Once you see it appear in the dashboard search results, the log is being successfully imported. The screenshot on the following page demonstrates this:



All Events

Type	@timestamp	Ip
nmultiline	2024-12-10T20:08:34.465850872Z	192.168.107.60

It is worth mentioning that the type field is the name of the input you created in the `nxlog.conf` file, this allows you to differentiate between separate log files.

For more information on configuring Nagios Log Server with Windows, see the NXLog User Guide.

Finishing Up

This completes the documentation on configuring NXLog to send multi-line log files. Specifically this was for log files with the date time format **ISO8601**. Other methods can be used as NXLog and Nagios Log Server are flexible in their configurations.

If you have additional questions or other support related questions, please visit us at our Nagios Support Forums or Knowledgebase:

Visit Nagios Support Forum   Visit Nagios Knowledgebase

Finishing Up

This completes the documentation on **ADD SUBJECT HERE**. If you have additional questions or other support-related questions, please visit us at our Nagios Support Forum, Nagios Knowledge Base, or Nagios Library:

Visit Nagios Support Forum Visit Nagios Knowledge Base Visit Nagios Library