Configuring NXLog to Send MultiLine Log Files in Nagios Log Server 2024

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:

```
2016-10-07 09:59:45,806 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, a second file will be created called C:\Logs\test.log with the following contents:

```
2016-10-07 09:59:45,806 INFO  SOME_SERVER This is the header row
```

```
This is the second line
```

```
This is the third line
```

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

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

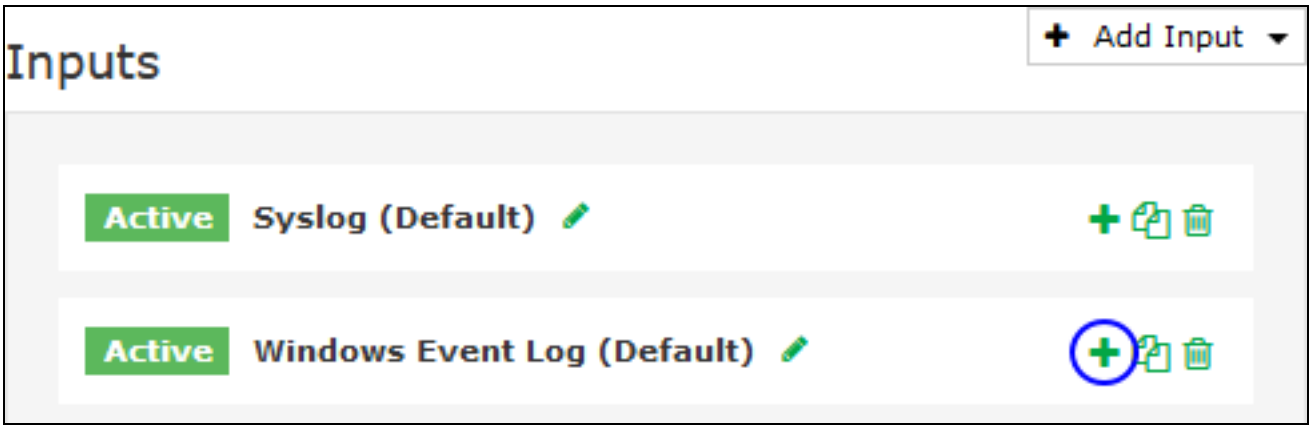
Using those steps you will be able to successfully follow this KB article and confirm the functionality works. Every time the command is executed above, a multi-line entry is added to the C:\Logs\Important_File.log log file. Even though technically the date is incorrect this will not matter, it is simply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

Create Input On Nagios Log Server

The first step is to configure the NLS input to identify multi-line logs.

1. Login to one of your Nagios Log Server instances as an Admin user Click Configure on the navigation bar.

2. In the left pane under Global (All Instances) click Global Config.



3. Under Inputs find the Windows Event Log (Default) input. Click the + sign and you should be presented with the following:

```
tcp {
```

```
type => 'eventlog'
```

```
port => 3515
```

```
codec => json
```

```
}
```

You need to modify it so it looks like the following:

```
tcp {
```

```
type => 'eventlog'
```

```
port => 3515
```

```
codec => json
```

```
codec => multiline {
```

```
pattern => "^%{TIMESTAMP_ISO8601}"
```

```
negate => true
```

```
what => "previous"
```

```
}
```

```
}
```

You can see that the following was added:

```
codec => multiline {
```

```
pattern => "^%{TIMESTAMP_ISO8601}"
```

```
negate => true
```

```
what => "previous"
```

```
}
```

4. Click the Save button.

5. Then click the Verfiy button above to ensure this is a valid configuration.

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

7. Click the Apply button.

8. Click Yes, Apply Now.

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

Configure NXLog

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

2. 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 = 'Important_File';
```

```
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.

You will also need to modify the route section to include the Important_File input. Find this section:

```
<Route 1>
```

```
Path internal, file1, eventlog => out
```

```
</Route>
```

On the Path line you need to add , Important_File after eventlog:

```
<Route 1>
```

```
Path internal, file1, eventlog, Important_File => out
```

```
</Route>
```

3. Save the changes you just made, the next step will be to restart NXLog.

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

What was 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 - : , are all character 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 = 'Important_File';
```

```
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 = 'Important_File'; tells NXLog to send the entry to NLS with the type defined as 'Important_File'

    - 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 1>
```

```
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 Important_File

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

@timestamp	host	type	message	Actions
2019-02-14T11:48:13.924+11:00	10.25.14.5	Important_File	2016-10-07 09:59:45,806 INFO SOME_SERVER This is the header row This is the second line This is the third line	

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.