



SC4S Customization

 When building custom parsers for SC4S, a single typo or syntax issue will send SC4S into a boot-loop. This won't cause the SC4S service to show as failed, so you need to be extra diligent. When restarting SC4S after making any customizations, always do so in the following manner:

sudo systemctl restart sc4s && sudo journalctl -f -u sc4s

If the parser works, you should see the service restart and end with **starting syslog-ng**

If you see pages of text with parser errors, your parser failed. Use **ctrl + c** to stop the journal and investigate what went wrong with your parser.

 Information contained within this article may need review. Always attempt to test changes in a lab setting before making changes to production environments.

- [Overview](#)
 - [Flow Diagram](#)
- [Assigning/Changing Host Names](#)
 - [Procedure](#)
 - [Example](#)
- [Simple Port](#)
 - [Procedure](#)
 - [Example](#)
- [Custom Host/IP filter](#)
 - [Procedure](#)
 - [Example 1](#)
 - [Example 2](#)
- [Custom MSG Parser](#)
 - [Procedure](#)
 - [Example](#)
- [Null Queue](#)
 - [Reference](#)
 - [Procedure:](#)
 - [Example](#)
 - [Example 2](#)
- [Adjust Timezone](#)
 - [Example 1](#)
 - [Example 2](#)
- [Troubleshooting](#)

Overview

SC4S is very extensible, meaning it allows you multiple layers of customizability from adding simple port definitions, to custom host definitions, to complete msg parsing additions.

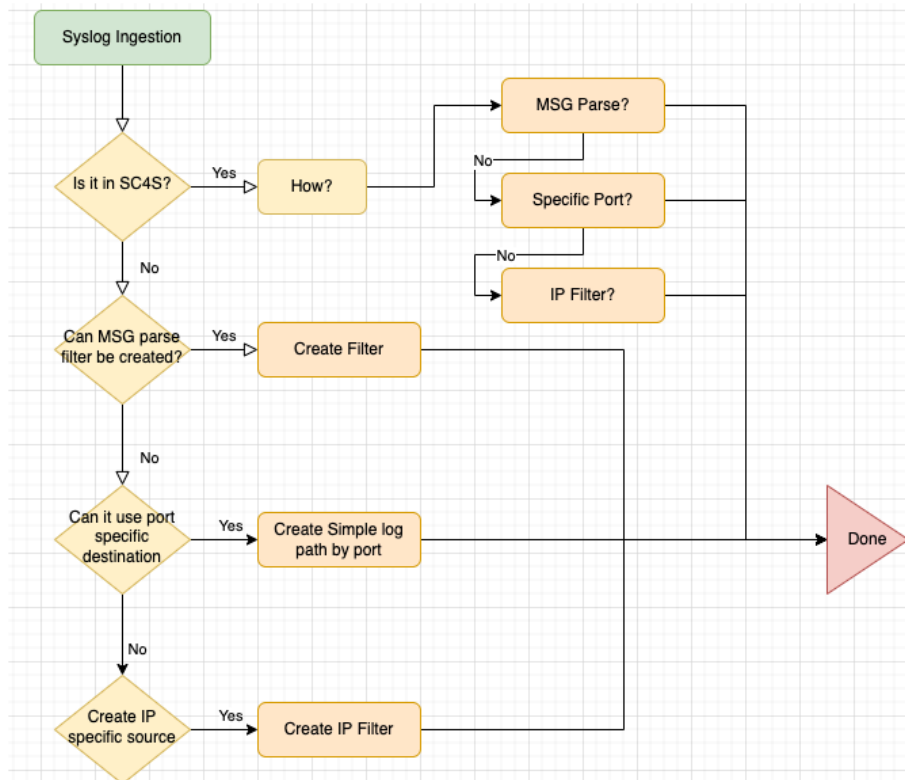
i SC4S, at its core, is a containerized `syslog-ng` instance that accepts incoming syslog, and outputs via the Splunk HTTP Event Collector. The distribution of `syslog-ng` used is named `AxoSyslog`. Their documentation can be quite valuable when customizing SC4S.

[▲ AxoSyslog documentation](#)

Flow Diagram

The below diagram shows the thought process for new syslog data that should be ingested.

⚠ IP FILTERS ARE LAST RESORT



Assigning/Changing Host Names

Procedure

1. Open the `$SC4S_Home/local/context/hosts.csv`
2. Add a newline
3. Enter the source ip and desired hostname following below convention
 - a. `<src ip>,HOST,<desired hostname>`
4. save

Example

- 1 `169.254.0.2,H0ST,foo.example`
- 2 `172.17.1.12,H0ST,LF1-P-SFTP01`

Simple Port

Procedure

Example

Custom Host/IP filter

Three files are required to be updated to create a custom host/IP filter.

Procedure

1. Create the dir and any required parent directories for your app parser.
 - a. `$SC4S_Home/local/config/app_parsers/net_source`
2. Create a .conf file for your app parser following the naming convention of `app-<vendor name>_<product>.conf`
 - a. i.e. `app-cerberus_ftp.conf`
3. Paste in the app parsing config based on the example below, changing the **block** & **application** definitions and any relevant source/index/sourcetype configs.
 - a. Block and application definition names should follow the naming convention of `<vendor>_<product>-parser`
4. restart SC4S `sudo systemctl restart sc4s`

Example 1

```
1 # $SC4S_Home/local/config/app_parsers/net_source/app-netapp_netapp_syslog.conf
2 # Netapp syslog by host filter
3 block parser netapp_syslog-parser() {
4     channel {
5         rewrite {
6             r_set_splunk_dest_update(
7                 index('netapp')
8                 sourcetype('netapp:syslog')
9                 vendor('netapp')
10                product('netapp:syslog')
11                template('t_msg_only')
12            );
13        };
14    };
15 };
16 };
17
18 application app-netapp_netapp_syslog-parser[sc4s-syslog] {
19     filter {
20         netmask(###.###.###.###/##)
21         or netmask (###.###.###.###/##)
22         or netmask (###.###.###.###/##)
23     };
24     parser { netapp_syslog-parser();
25     };
26 };
```

Example 2

```
1 # $SC4S_Home/local/config/app_parsers/net_source
2 block parser cerberus_ftp-parser() {
3     channel {
4         rewrite {
5             r_set_splunk_dest_default(
6                 index('web')
7                 sourcetype('cerberus')
8                 vendor('cerberus')
9                 product('ftp')
10                template('t_msg_only')
11            );
12        };
13    };
14
15
16 };
17 };
18
19 application app-vps-cerberus_ftp-parser[sc4s-syslog] {
20     filter {
21         host('^cerberus-*' type(glob))
22     };
23     parser { cerberus_ftp-parser();
24     };
25 };
```

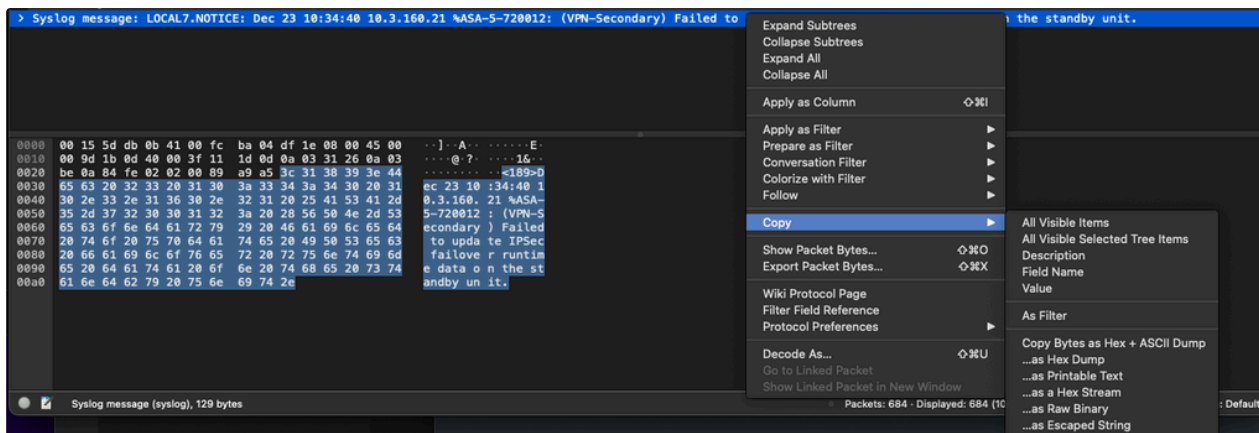
Custom MSG Parser

Creating a custom MSG parsing app parser can be fairly complex depending on sending host and requires the message to have differentiating characteristics. For example the message includes a vendor name, model, or specific pattern that can be matched against.

This procedure is a general guideline not all encompassing, feel free to reachout to [@Sean Elliott](#) with specific questions.

Procedure

1. Collect sample logs.
 - a. Grab them straight from the network interface using tcpdump. ie. `sudo tcpdump -X -i eth0 src ###.###.###.### -w /tmp/test.pcap`
 - b. Export these from the host and use an application like [Wireshark](#) to read the file.
 - c. Copy out the raw syslog message by right clicking on the syslog section of the message, selecting copy, then `...as Printable Text`



2. From here you can use a service like [regex101](#) to create a custom regex expression to filter out the messages.
3. Create a new file location depends on the type of sending format.
 - a. Proper syslog formatted messages will be in `$SC4S_home/local/config/app_parser/syslog`
 - b. Messages that do not conform to the RFC5424 standard would go to `SC4S_home/local/config/app_parser/raw`
4. Once the folder has been identified add a file following the naming convention of `app-<vendor name>_<product>.conf`
 - a. ie. `app-ibm_zsecure.conf`
5. Paste in the app parsing config based on the example below, changing the **block** & **application** definitions and any relevant source/index/sourtype configs.
 - a. Block and application definition names should follow the naming convention of `<vendor>_<product>-parser`

Example

```

1 # IBM zSecure Parser
2 block parser ibm_zsecure-parser() {
3     channel {
4         rewrite {
5             r_set_splunk_dest_update(
6                 sourcetype("zSecure:Syslog")
7                 index("db")
8                 vendor("ibm")
9                 product("zsecure")
10            );
11        };
12    };
13 };
14
15 application ibm_zsecure[sc4s-postfilter] {
16     filter {
17         message('LEEF\:\1\0\|IBM\|')
18     };
19     parser {
20         ibm_zsecure-parser();
21     };
22 };

```

Null Queue

The most efficient way to drop events from being ingested by Splunk is to filter them out as close to the source as possible. This means if it is possible for the device to “simply not send” them that is the preferred method. But in many cases we will have to configure the event to be routed to a “null queue” and performing this in syslog-ng (SC4S) is the preferred method.

Reference

 [Read First - Splunk Connect for Syslog](#)


Procedure:

1. Create an app_parser file in `/opt/sc4s/local/config/app-parsers/`
2. Update the `filter` under the `application` section as required to match the events needing dropped.
3. Restart SC4S `sudo systemctl restart sc4s`

Example

```
1 # Null queue cisco debug messages
2 block parser cisco_ios_debug-postfilter() {
3     channel {
4         rewrite(r_set_dest_splunk_null_queue);
5     };
6 };
7 application cisco_ios_debug-postfilter[sc4s-postfilter] {
8     filter {
9         ( "${.netsource.sc4s_vendor}" eq "cisco" and "${.netsource.sc4s_product}" eq "ios")
10        and message('^[^\-]+-7-');
11    };
12    parser { cisco_ios_debug-postfilter(); };
13 };
```

Example 2


 Method used to drop Cisco ISE messages from a specific source IP

```
1 block parser cisco_ise_debug-postfilter() {
2     channel {
3         rewrite(r_set_dest_splunk_null_queue);
4     };
5 };
6 application cisco_ise_debug-postfilter[sc4s-postfilter] {
7     filter {
8         ( "${fields.sc4s_vendor}" eq "cisco" and "${fields.sc4s_product}" eq "ise")
9         and "${fields.sc4s_fromhostip}" eq "172.25.128.97"
10    };
11    parser { cisco_ise_debug-postfilter(); };
12 };
```

Adjust Timezone

Some times depending on log source it is required to specifically set a timezone for a host. The below examples shows how to perform this using an app parser.

Example 1

 As of 27 April 2023, this was implemented successfully.

```
1 block parser app-dest-rewrite-panos_tz_fix() {
```

```

2     channel {
3         rewrite { fix-time-zone("America/Halifax"); };
4     };
5 };
6 application app-dest-rewrite-panos_tz_fix[sc4s-lp-dest-format-d_hec_fmt] {
7     filter {
8         match('pan' value('fields.sc4s_vendor') type(string))
9         and match('panos' value('fields.sc4s_product') type(string))
10    };
11
12    parser { app-dest-rewrite-panos_tz_fix(); };
13 };

```

Example 2

```

1 #filename: /opt/sc4s/local/app_parsers/postfilter/app-dest-rewrite-fix_tz_something.conf
2
3 block parser app-dest-rewrite-checkpoint_drop-d_fmt_hec_default() {
4     channel {
5         rewrite { fix-time-zone("EST5EDT"); };
6     };
7 };
8
9 application app-dest-rewrite-fix_tz_something-d_fmt_hec_default[sc4s-lp-dest-format-d_hec_fmt] {
10    filter {
11        match('checkpoint' value('fields.sc4s_vendor') type(string))
12        and match('syslog' value('fields.sc4s_product') type(string))
13
14        and match('Drop' value('.SDATA.sc4s@2620.action') type(string))
15        and match('12.' value('.SDATA.sc4s@2620.src') type(string) flags(prefix) );
16
17    };
18    parser { app-dest-rewrite-fix_tz_something-d_fmt_hec_default(); };
19 };

```

Troubleshooting

Useful tcpdump command:

```

1 tcpdump -i any -nnvAs0

```

You can add the below two parameters in the env_file followed by SC4S service restart. This will populate the "archive" directory with all information/metadata that SC4S is assigning to incoming logs. This is useful to identify the values of the extra fields such as program/etc which can be used in custom parsers.

```

1 SC4S_DEST_GLOBAL_ALTERNATES=d_hec_debug
2 SC4S_ARCHIVE_GLOBAL=yes

```