

# Exporting Splunk Data at Scale with Scribl

## scribl.py

Exporting Splunk Data at Scale with [Scribl](#). This is a python script that can be run on each Splunk Indexer for the purpose of exporting historical bucket data (raw events + metadata) at scale by balancing the work across multiple CPUs then forwarding to Cribl.

## Background


Splunk to Cribl = [scribl](#) (#thanksKam)

Exporting large amounts of previously indexed data from Splunk is challenging via the Splunk-supported approaches detailed here: <https://docs.splunk.com/Documentation/Splunk/8.2.6/Search/Exportsearchresults>.


The core Splunk binary in every install provides a switch (cmd exporttool) that allows you to export the data from the compressed buckets on the indexers back into their original raw events. You can dump them to very large local csv files or stream them to stdout so a script can redirect over the network to a receiver such as Cribl Stream. This switch has been used by others for quite a while but it isn't well documented.

Assuming that Splunk is installed in /opt/splunk/, the below commands can be applied to a particular bucket in an index called "bots" to export it.

Exporting to stdout:

```
 /opt/splunk/bin/splunk cmd exporttool /opt/splunk/var/lib/splunk/bots/db/db_1564739504_1564732800_2394 /dev/stdout -csv
```

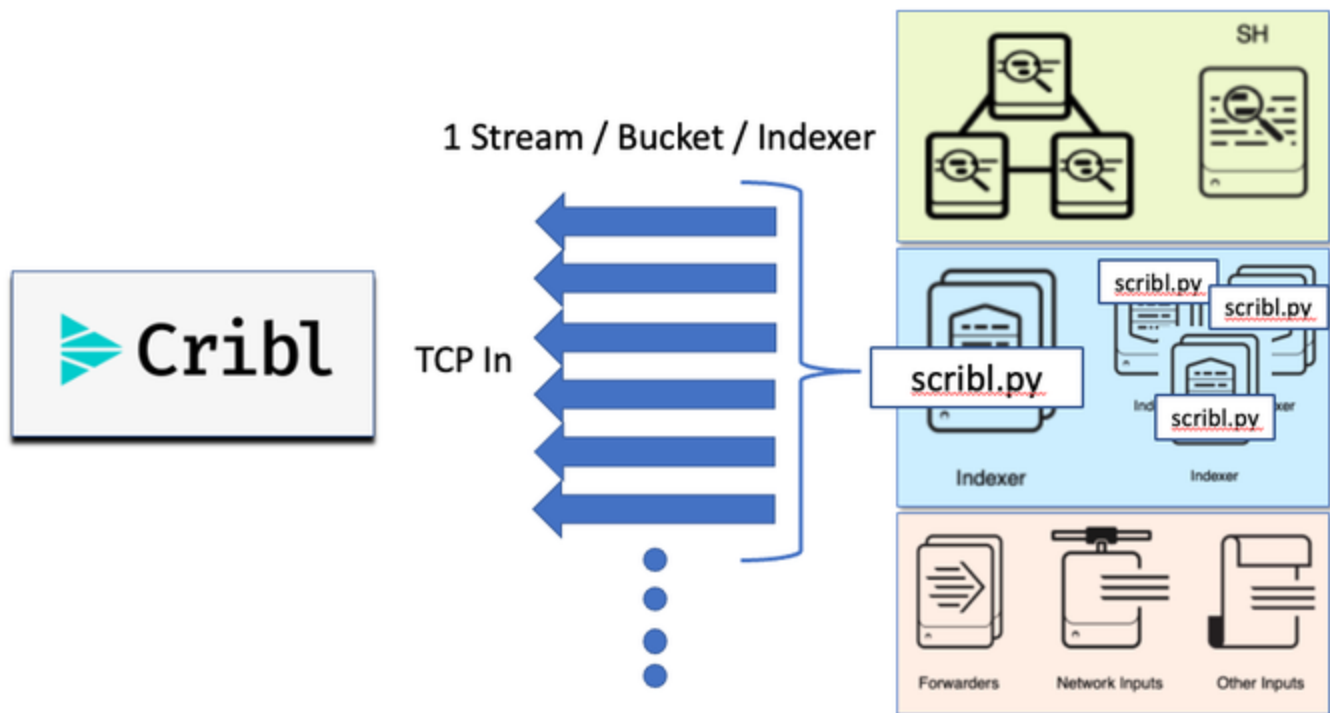
Exporting to a local csv file:

```
 /opt/splunk/bin/splunk cmd exporttool /opt/splunk/var/lib/splunk/bots/db/db_1564739504_1564732800_2394 /exports/bots/db_1564739504_1564732800_2394.csv -csv
```

There will be many buckets so some poor soul will need to build a script to export all or some of the buckets and some sort of parallelization should be used to speed the process up. The exported data will be very large (uncompressed, 3-20x) compared to the size of the individual buckets that make up the index!

## Requirements

Splunk stores its collected data on the indexers within the "Indexing Tier" as detailed below. The data is compressed and stored in a collection of time series buckets that reside on each indexer. Each bucket contains a rawdata journal, along with associated tsidx, and metadata files. The search heads access these buckets and it's very rare for someone to access them directly from the indexer CLI unless there is a need to export data to retrieve the original raw events. We will use the indexer CLI to export the original raw events (per bucket and in parallel) as well as a few other pieces of important metadata as detailed below.



For a deeper dive into how Splunk indexes data, see this: <https://docs.splunk.com/Documentation/Splunk/latest/Indexer/HowSplunkstoresindexes>

You will need:

- CLI access to each Linux indexer with the index/buckets that need to be exported. This process only applies to on-prem or non-SplunkCloud deployments.
- To install nc (netcat) on each indexer to act as the transport mechanism until we have enough demand to build the transport into the script.
- To make sure outbound communication from each indexer to the Cribl Worker TCP port is open.

## Technical

### Scale

We achieve scale for large volumes of data by processing buckets in parallel across as many CPUs as you would like to dedicate AND by streaming the data directly from disk with a single read to Cribl without ever having to write extracted/uncompressed event data to disk. Extracting/uncompressing the event data to disk would result in enormous disk IO bottlenecks and disk space consumption.

Disk speed (IOPS) and the number of CPUs are generally your limiting factors on the indexers. While disk speed is a factor, it's usually not a factor in the overall scale picture because Splunk indexers will usually have high IOPs capabilities. You can monitor the linux processes to get a feel for whether scribl processes are in SLEEP or RUN mode. If they spend the majority of their time in SLEEP mode, they are being throttled by disk, network, Cribl workers, etc and adding more CPUs will probably not buy you more speed.

The scribl script running on your indexers and Cribl Stream workers are built to scale and will usually not be your bottleneck. Your bottlenecks will almost certainly be bandwidth constraints between your indexers and your final destination. Depending on where you deploy your Cribl Stream workers, that bandwidth bottleneck might exist between the indexers and Cribl workers or between your Cribl workers and the final destination. If you happen to have unlimited bandwidth, you will likely find your next bottleneck to be the ingest rate at your destination platform.

### Exported Data Format

The exported data will be csv formatted with a header followed by the individual events. It's important to call out that these events are often multiline events with the most common example being windows logs. The below events are examples that are generated by Splunk and then passed via stdin to the scribl.py script.

The `_raw` field contains the original event and the other fields were captured/created during ingest. `_time` is the time extracted from the event which will be the primary time reference used by the destination analytics platform. The `sourcetype` field will likely be what is used by the destination to determine how to parse and where to route the event.

Example:



Host Version = 4.0

Engine Version = 4.0

Pipeline ID = 1

Command Type = Cmdlet


Command Path =

User = THIRSTYBERNER\SYSTEM

```
</Data><Data Name='UserData'></Data><Data Name='Payload'>ParameterBinding(Out-Null): name=""InputObject""; value=""True""
```

```
</Data></EventData></Event>","_indextime::1564734908 punct::<_='://..////'><><_='--'_='{----}'/><></><></><><"
```

Example Usage:

 `./scribl.py -h`  
usage: [scribl.py](#) [-h] [-t] [-n NUMSTREAMS] [-l LOGFILE] [-et EARLIEST]  
[-lt LATEST] [-d DIRECTORY] [-r REMOTEIP] [-p REMOTEPORT]

This is to be run on a Splunk Indexer for the purpose of exporting buckets and streaming their contents to Cribl Stream

optional arguments:

-h, --help show this help message and exit

-t, --TLS Send with TLS enabled

-n NUMSTREAMS, --numstreams NUMSTREAMS

the number of parallel stream to utilize

-l LOGFILE, --logfile LOGFILE

specify the location to write/append the logging

-et EARLIEST, --earliest EARLIEST

specify the earliest epoch time for bucket selection

-lt LATEST, --latest LATEST

specify the latest epoch time for bucket selection

required named arguments:

-d DIRECTORY, --directory DIRECTORY


Source directory containing the buckets

-r REMOTEIP, --remoteIP REMOTEIP

Remote address to send the exported data to

-p REMOTEPORT, --remotePort REMOTEPORT

Remote TCP port to be used

 `scribl.py -d /opt/splunk/var/lib/splunk/bots/db/ -r 14.2.39.121 -p 20000 -t -n4 -l /tmp/scribl.log -et 1564819155 -lt 1566429310`

## On No! My Splunk license expired!

Worry not, my friend. When the enterprise license expires, Splunk customers are free to use the 60-day trial or even the free version of Splunk to perform the export. Sanity check my claim here: <https://docs.splunk.com/Documentation/Splunk/9.0.0/Admin/MoreaboutSplunkFree> .

We don't care about indexing new data and we don't care about distributed search since we will use the trial/free Splunk binary in a standalone manner on each of the indexers that have data we need to migrate. Just install trial/free Splunk on top of or alongside the existing install and point scribl.py at your splunk binary and the directory containing the buckets you need to export.

## Cribl Stream Config

You can get started instantly with Cribl Cloud or even using the Cribl Free [license option](#) but keep in mind daily ingest limits (very generous) and # of cores (also very generous at 10) that can be used may factor into a full scale data export. If you choose to install Cribl Stream on-prem on in your own cloud, the [documentation](#) is your friend and will get you going quickly.

Once you have satisfied the above requirements (CLI, nc, and firewall) on your Splunk indexers, grab the [scribl.py script from the github repo](#) and copy it over to each indexer. The only thing in the script that is hard coded is the default install location of Splunk (/opt/splunk) which you can easily modify if you are running a non-default config. Keep in mind that we are running the script directly on the Splunk indexers and a python binary is kept under \$SPLUNK\_HOME/bin.

Add a new Source:

Stream > Sources > TCP > scribl X

Configure Status Charts Live Data Logs Help ?

General Settings

TLS Settings (Server Side)

Persistent Queue Settings

Processing Settings ^

Custom Command

Event Breakers

Fields

Pre-Processing

Advanced Settings

Connected Destinations

Input ID\* ? Enabled Yes

scribl

\_\_inputId=='tcp:scribl' ?

Address\* ?

0.0.0.0

Port\* ?

20000

Enable Header ? No

Tags ?

scribl x

Enable and Config TLS (recommended):

Stream > Groups > default > Sources > TCP > scribl

Configure

Status

Charts

Live Data

Logs

General Settings

TLS Settings (Server Side)

Processing Settings

Event Breakers

Fields

Pre-Processing

Advanced Settings

Connected Destinations

Enabled ?

Yes ☒ Autofill?

Certificate name ?

Select one

Private key path\* ?

/opt/criblcerts/criblcloud.key

Passphrase ?

Enter passphrase

Certificate path\* ?

/opt/criblcerts/criblcloud.crt

CA certificate path ?

Enter CA certificate path

Authenticate client (mutual auth) ? ☐ No

Minimum TLS version ?

TLSv1.2

Maximum TLS version ?

Select one



Create a new Event Breaker and set it to "[\n\r]+\\d{10}," as detailed below:

Rule Name\* ?

scribl

Filter Condition\* ?

true

EVENT BREAKER SETTINGS

Enabled ?

Yes



Event Breaker Type\* ?

Regex

▼

Event Breaker\* ?

/ [\n\r]+d{10},


/  



Max Event Bytes ?

51200

TIMESTAMP SETTINGS

Timestamp Anchor\* ?

/ 

/  

Timestamp Format\* ?

☒ Autotimestamp Scan Depth ? 150

☐ Manual Format ?

☐ Current Time ?

Default Timezone ?

Local

▼

Earliest timestamp allowed ?

-420weeks

Future timestamp allowed ?

+1week

ADD FIELDS TO EVENTS ?

+ Add Field

In

Out

Upload Sample File

Paste your events here or upload a sample file

Timestamp Anchor

Event Breaker

Timestamp

Cancel

OK

Validate:

Stream > Groups > default > Knowledge > Event Breaker Rules > exporttool > Rules

Rule Name\*

Filter Condition\*

EVENT BREAKER SETTINGS

Enabled ☒ Yes

Event Breaker Type\*

Event Breaker\*

Max Event Bytes

In **Out**

1

2022-05-25 15:42:08.572 -05:00 # \_time: 1653511328.572

2

2022-05-25 15:42:08.574 -05:00 # \_time: 1653511328.574

3

Pipeline\*

Enable Expression ☐ No

Output

Description

Final ☒ Yes

Attach functions to your pipeline to transform the inbound data as detailed below:





1

Parser

true

On



Filter ?

Help ?

true



Description ?

Extract the initial fields from \_raw

Final ?

No

Operation Mode\* ?

Extract



Type\* ?

CSV



Library ?

Select from Library



Source Field ?

\_raw

Destination Field ?

Destination field name

List of Fields ?

source x

host x

sourcetype x

\_raw x

meta x





2

Regex Extract

true

On



Filter ?

Help

true



Description ?

Capture the original index time and use that as \_time

Final ?

No

Regex\* ?

/ ^\_indextime::(?<\_time>\d{10})

/ Rb

Additional Regex

+ Add Regex

Source Field ?

meta

ADVANCED SETTINGS

Max Exec ?

100

Field Name Format Expression ?

Enter field name format expression.



Overwrite Existing Fields ?

Yes

2MasktrueOn

Filter ?Help ?

true

Description ?

Enter a description

Final ?

No

Masking Rules\*

Match Regex ?	Replace Expression ?
/ (\w+::)	' '

+ Add Rule

Apply to Fields ?

host x source x sourcetype x

> ADVANCED SETTINGS

3Regex ExtracttrueOff

Filter ?Help ?

true

Description ?

Extract \_indextime, \_subsecond, and punt fields from meta

Final ?

No

Regex\* ?

/ ^\_indextime::(?<\_indextime>\d{10})(\s\_subsecond::(?<\_subsecond>\S{4})

Additional Regex

+ Add Regex

Source Field ?

meta

4

Eval

true

On

...

Filter ?

Help ?

true

Description ?

Remove the meta field

Final ?

No

Evaluate Fields ?

+ Add Field

Keep Fields ?

Enter field names

Remove Fields ?

meta x

5

Eval

true

On

...

Filter ?

Help ?

true

Description ?

Remove the cribl\_breaker field

Final ?

No

Evaluate Fields ?

+ Add Field

Keep Fields ?

Enter field names

Remove Fields ?

cribl\_breaker x

Validate the transformation then attach your destination(s) to the route:

```
1 2022-06-15 09:44:25.632 -05:00
  _raw: <Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'><System><Provider Name='Microsoft-Windows-PowerShell' Guid='{A0C1853B-5C40-4B15-8766-3CF1C58F985A}' /><EventID>4103</EventID><Version>1</Version><Level>4</Level><Task>106</Task><Opcode>20</Opcode><Keywords>0x0</Keywords><TimeCreated SystemTime='2019-08-02T09:38:04.096934100Z' /><EventRecordID>6165455</EventRecordID><Correlation ActivityID='{135BC459-4718-0000-F407-7B131847D501}' /><Execution ProcessID='5080' ThreadID='932' /><Channel>Microsoft-Windows-PowerShell/Operational</Channel><Computer>titan.thirstyberner.com</Computer><Security UserID='S-1-5-18' /></System><EventData><Data Name='ContextInfo'>Severity = Informational
    Host Name = Default Host
    Host Version = 4.0
    Host ID = bc05089d-06d9-49cd-8c05-5672caf6aace
    Engine Version = 4.0
    Runspace ID = 5443b262-5049-49b7-9db1-f7995525117e
    Pipeline ID = 1
    Command Name = Out-Null
    Command Type = Cmdlet
    Script Name =
    Command Path =
    Sequence Number = 258134
    User = THIRSTYBERNER\SYSTEM
    Shell ID = Microsoft.PowerShell
  </Data><Data Name='UserData'></Data><Data Name='Payload'>ParameterBinding(Out-Null): name="InputObject"; value="True"
</Data></EventData></Event> Show less
# _time: 1655304265.632
  cribl_breaker: scribl:scribl
  cribl_pipe: scribl
  host: titan
  source: WinEventLog:Microsoft-Windows-PowerShell/Operational
  sourcetype: XmlWinEventLog:Microsoft-Windows-PowerShell/Operational
```

## Caveats:

### Splunk Event Sizes

You need to pay attention to event sizes in Splunk as it pertains to the Event breaking in Cribl. As noted above in the Event Breaker screenshot, the max event size has a default setting of 51200 bytes. If you use scribl to send events into Cribl Stream larger than that, things break. Either increase your event breaking max event size, use the Cribl Stream Pipeline to drop the large events (example: by sourcetype), or do not use scribl to export the buckets containing the large events.

Here is a quick Splunk search highlighting the large events that need to be dealt with:

New Search

Save As Create Table View Close

index=bots|eval l=len(\_raw)|where 25000>l|stats count values(sourcetype) by l|sort - l

All time

31,304,827 events (before 6/23/22 4:06:41.000 PM) No Event Sampling

Job

Smart Mode

Events

Patterns

Statistics (10,000)

Visualization

20 Per Page

Format

Preview

< Prev

1

2

3

4

5

6

7

8

...

Next >

	count	values(sourcetype)
2400680	31	stream:http
174623	1	stoq
159433	31	stoq
156462	31	stoq
156297	31	stoq
153023	31	stoq

## Bottlenecks

As mentioned above, the bottleneck you will most likely run into will be bndwidth in your data path or ingest rate at the final destination. Anything you can do to parallelize that final write will pay dividends. For example, you may want to use Cribl Stream's Output Router to write to multiple S3 buckets based on the original Splunk Index or Sourcetype if bandwidth is not your bottleneck.

## To Do:

- ~~Add min and max times to determine which buckets should be exported (complete 6/2022)~~
- Applies this to a smartstore config (auth, encryption, data format, etc)
- Sync with Spico and other partners who would be interested
- How about this as a collector?