

Splunk Cheat Sheet: Search and Query Commands

January 25, 2023 / By Cassandra Lee

SPLUNK CHEAT SHEET



 Listen to the article

Whether you're a cyber security professional, data scientist, or system administrator when you mine large volumes of data for insights using Splunk, having a list of Splunk query commands at hand helps you focus on your work and solve problems faster than studying the official documentation.

This article is the convenient list you need. It provides several lists organized by the type of queries you would like to conduct on your data: basic pattern search on keywords, basic filtering using regular expressions, mathematical computations, and statistical and graphing functionalities.

The following Splunk cheat sheet assumes you have Splunk installed. It is a refresher on useful Splunk query commands. Download a PDF of this Splunk cheat sheet [here](#).

Search cheats here



Brief Introduction of Splunk

The Internet of Things (IoT) and Internet of Bodies (IoB) generate much data, and searching for a needle of datum in such a haystack can be daunting.

Splunk is a Big Data mining tool. With Splunk, not only is it easier for users to excavate and analyze machine-generated data, but it also visualizes and creates reports on such data.

Splunk Enterprise search results on **sample data**

Splunk contains three processing components:

- The **Indexer** parses and indexes data **added** to Splunk.
- The **Forwarder** (optional) sends data from a source.
- The **Search Head** is for searching, analyzing, visualizing, and summarizing your data.

Search Language in Splunk

Splunk uses what's called Search Processing Language (SPL), which consists of keywords, quoted phrases, Boolean expressions, wildcards (*), parameter/value pairs, and comparison expressions.

Unless you're joining two explicit Boolean expressions, omit the AND operator because Splunk assumes the space between any two search terms to be AND.

Basic Search offers a shorthand for simple keyword searches in a body of indexed data myIndex without further processing:

```
index=myIndex keyword
```

An event is an entry of data representing a set of values associated with a timestamp. It can be a text document, configuration file, or entire stack trace. Here is an example of an event in a web activity log:

```
[10/Aug/2022:18:23:46] userID=176 country=US paymentID=30495
```

Search commands help filter unwanted events, extract additional information, calculate values, transform data, and statistically analyze the indexed data. It is a process of narrowing the data down to your focus. Note the decreasing number of results below:

Finding entries without IPv4 address on [sample data](#)

Common Search Commands

COMMAND	DESCRIPTION
chart, timechart	Returns results in a tabular output for (time-series) charting
dedup X	Removes duplicate results on a field X

COMMAND	DESCRIPTION
eval	Calculates an expression (see Calculations)
fields	Removes fields from search results
head/tail N	Returns the first/last N results, where N is a positive integer
lookup	Adds field values from an external source
rename	Renames a field. Use wildcards (*) to specify multiple fields.
rex	Extract fields according to specified regular expression(s)
search	Filters results to those that match the search expression
sort X	Sorts the search results by the specified fields X
stats	Provides statistics, grouped optionally by fields
mstats	Similar to stats but used on metrics instead of events
table	Displays data fields in table format.
top/rare	Displays the most/least common values of a field
transaction	Groups search results into transactions
where	Filters search results using eval expressions. For comparing two different fields.

SPL SEARCH TERMS	DESCRIPTION
"White Black+Hat"	Find the exact phrase with the given special characters, irrespective of capitalization
Filter by fields	
source="/var/log/myapp/access.log" status=404	All lines where the field status has value 404 from the file /var/log/myapp/access.log
source="bigdata.rar:*" index="data_tutorial" Code=RED	All entries where the field Code has value RED in the archive bigdata.rar indexed as data_tutorial
index="customer_feedback" _raw="*excellent*"	All entries whose text contains the keyword “excellent” in the indexed data set customer_feedback
Filter by host	
host="myblog" source="/var/log/syslog" Fatal	Show all Fatal entries from /var/log/syslog belonging to the blog host myblog
Selecting an index	
index="myIndex" password	Access the index called myIndex and text matching password.
source="test_data.zip:*"	Access the data archive called test_data.zip and parse all its entries (*).
sourcetype="datasource01"	(Optional) Search data sources whose type is datasource01.

This syntax also applies to the arguments following the search keyword. Here is an example of a longer SPL search string:

```
index=* OR index=_* sourcetype=generic_logs | search Cybersecurity | head 10000
```

In this example, `index=* OR index=_* sourcetype=generic_logs` is the data body on which Splunk performs `search Cybersecurity`, and then `head 10000` causes Splunk to show only the

first (up to) 10,000 entries.

Basic Filtering

You can filter your data using regular expressions and the Splunk keywords `rex` and `regex`. An example of finding deprecation warnings in the logs of an app would be:

```
index="app_logs" | regex error="Deprecation Warning"
```

SPL FILTERS	DESCRIPTION	EXAMPLES
search	Find keywords and/or fields with given values	<ul style="list-style-type: none">• <code>index=names search Chris</code>• <code>index=emails search emailAddr="*mysite.com"</code>
regex	Find expressions matching a given regular expression	Find logs not containing IPv4 addresses: <code>index=syslogs regex !="^\d{1,3}.\d{1,3}\.\d{1,3}\.\d{1,3}"</code>
<u>rex</u>	Extract fields according to specified regular expression(s) into a new field for further processing	Extract email addresses: <code>source="email_dump.txt" rex field=_raw "From: <(?(from>.*)> To: <(?(to>.*)>"</code>

The biggest difference between `search` and `regex` is that you can only exclude query strings with `regex`. These two are equivalent:

- `source="access.log" Fatal`
- `source="access.log" | regex _raw=".*Fatal.*"`

But you can only use `regex` to find events that do not include your desired search term:

- `source="access.log" | regex _raw!=".*Fatal.*"`

The Splunk keyword `rex` helps determine the alphabetical codes involved in this dataset:

Alphabetical codes in sample data

Calculations

Combine the following with eval to do computations on your data, such as finding the mean, longest and shortest comments in the following example:

```
index=comments | eval cmt_len=len(comment) | stats  
  
avg(cmt_len), max(cmt_len), min(cmt_len) by index
```


FUNCTION	RETURN VALUE / ACTION	USAGE:EVAL FOO=...
abs (X)	absolute value of X	abs (number)
case (X, "Y", ...)	Takes pairs of arguments X and Y, where X arguments are Boolean expressions. When evaluated to TRUE, the arguments return the corresponding Y argument	case(id == 0, "Amy", id == 1, "Brad", id == 2, "Chris")
ceil (X)	Ceiling of a number X	ceil(1.9)
cidrmatch ("X", Y)	Identifies IP addresses that belong to a particular subnet	cidrmatch("123.132.32.0/25", ip)
coalesce (X, ...)	The first value that is not NULL	coalesce(null(), "Returned val", null())
cos (X)	Cosine of X	n=cos(60) #1/2
exact (X)	Evaluates an expression X using double precision floating point arithmetic	exact(3.14*num)
exp (X)	e (natural number) to the power X (e ^X)	exp(3)
if (X, Y, Z)	If X evaluates to TRUE, the result is the second argument Y. If X evaluates to FALSE, the result evaluates to the third argument Z	if(error==200, "OK", "Error")
in (field, valuelist)	TRUE if a value in valuelist matches a value in field. You must use the in() function embedded inside the if() function	if(in(status, "404", "500", "503"), "true", "false")
isbool (X)	TRUE if X is Boolean	isbool(field)
isint (X)	TRUE if X is an integer	isint(field)
isnull (X)	TRUE if X is NULL	isnull(field)
isstr (X)	TRUE if X is a string	isstr(field)
len (X)	Character length of string X	len(field)

FUNCTION	RETURN VALUE / ACTION	USAGE:EVAL FOO=...
<code>like (X, "Y")</code>	TRUE if and only if X is like the SQLite pattern in Y	<code>like (field, "address%")</code>
<code>log (X, Y)</code>	Logarithm of the first argument X where the second argument Y is the base. Y defaults to 10 (base-10 logarithm)	<code>log (number, 2)</code>
<code>lower (X)</code>	Lowercase of string X	<code>lower (username)</code>
<code>ltrim (X, Y)</code>	X with the characters in Y trimmed from the left side. Y defaults to spaces and tabs	<code>ltrim (" ZZZabcZZ", " Z")</code>
<code>match (X, Y)</code>	TRUE if X matches the regular expression pattern Y	<code>match (field, "^\d{1,3}\.\d\$")</code>
<code>max (X, ...)</code>	The maximum value in a series of data X,...	<code>max (delay, mydelay)</code>
<code>md5 (X)</code>	MD5 hash of a string value X	<code>md5 (field)</code>
<code>min (X, ...)</code>	The minimum value in a series of data X,...	<code>min (delay, mydelay)</code>
<code>mvcount (X)</code>	Number of values of X	<code>mvcount (multifield)</code>
<code>mvfilter (X)</code>	Filters a multi-valued field based on the Boolean expression X	<code>mvfilter (match (email, "net\$"))</code>
<code>mvindex (X, Y, Z)</code>	Returns a subset of the multi-valued field X from start position (zero-based) Y to Z (optional)	<code>mvindex (multifield, 2)</code>
<code>mvjoin (X, Y)</code>	Joins the individual values of a multi-valued field X using string delimiter Y	<code>mvjoin (address, ";")</code>
<code>now ()</code>	Current time as Unix timestamp	<code>now ()</code>
<code>null ()</code>	NULL value. This function takes no arguments.	<code>null ()</code>
<code>nullif (X, Y)</code>	X if the two arguments, fields X and Y, are different. Otherwise returns NULL.	<code>nullif (fieldX, fieldY)</code>

FUNCTION	RETURN VALUE / ACTION	USAGE: EVAL FOO=...
<code>random()</code>	Pseudo-random number ranging from 0 to 2147483647	<code>random()</code>
<code>relative_time (X,Y)</code>	Unix timestamp value of relative time specifier Y applied to Unix timestamp X	<code>relative_time(now(), "-1d@d")</code>
<code>replace(X,Y,Z)</code>	A string formed by substituting string Z for every occurrence of regex string Y in string X The example swaps the month and day numbers of a date.	<code>replace(date, "^(\d{1,2})/(\d{1,2})/", "\2/\1/")</code>
<code>round(X,Y)</code>	X rounded to the number of decimal places specified by Y, or to an integer for omitted Y	<code>round(3.5)</code>
<code>rtrim(X,Y)</code>	X with the characters in (optional) Y trimmed from the right side. Trim spaces and tabs for unspecified Y	<code>rtrim(" ZZZZabcZZ", " Z")</code>
<code>split(X,"Y")</code>	X as a multi-valued field, split by delimiter Y	<code>split(address, ";")</code>
<code>sqrt(X)</code>	Square root of X	<code>sqrt(9) # 3</code>
<code>strftime(X,Y)</code>	Unix timestamp value X rendered using the format specified by Y	<code>strftime(time, "%H:%M")</code>
<code>strptime(X,Y)</code>	Value of Unix timestamp X as a string parsed from format Y	<code>strptime(timeStr, "%H:%M")</code>
<code>substr(X,Y,Z)</code>	Substring of X from start position (1-based) Y for (optional) Z	<code>substr("string",</code>



00:00

00:00



<code>tonumber(X,Y)</code>	Converts input string X to a number of numerical base Y (optional, defaults to 10)	<code>tonumber("FF",16)</code>
<code>tostring(X,Y)</code>	Field value of X as a string. If X is a number, it reformats it as a string. If X is a Boolean value, it reformats to “True” or “False” strings. If X is a number, the optional second argument Y is one of: “hex”: convert X to hexadecimal, “commas”: formats X with commas	This example returns <code>bar=00:08:20: makeresults eval bar = tostring(500, "duration")</code>

FUNCTION	RETURN VALUE / ACTION	USAGE: EVAL FOO=...
	and two decimal places, or "duration": converts seconds X to readable time format HH:MM:SS.	
typeof (X)	String representation of the field type	<pre>This example returns "NumberBool": makeresults eval n=typeof(12) + typeof(1==2)</pre>
urldecode (X)	URL X, decoded.	<pre>urldecode("http%3A%2F%2Fwww.site.com%2Fview%3Fr%3Dabout")</pre>
validate (X, Y, ...)	For pairs of Boolean expressions X and strings Y, returns the string Y corresponding to the first expression X which evaluates to False, and defaults to NULL if all X are True.	<pre>validate(isint(N), "Not an integer", N>0, "Not positive")</pre>



Want to Download All Our Premium Cheat Sheets?

No Problem! Just enter your email address



00:00

00:00




Email Address

DOWNLOAD

Statistical and Graphing Functions

Common statistical functions used with the `chart`, `stats`, and `timechart` commands. Field names can contain wildcards (`*`), so `avg(*delay)` might calculate the average of the `delay` and `*delay` fields.

FUNCTION	RETURN VALUE USAGE: STATS FOO=... / CHART BAR=... / TIMECHART T=...
avg (X)	average of the values of field X
count (X)	number of occurrences of the field X. To indicate a specific field value to match, format X as <code>eval (field="desired_value")</code> .
dc (X)	count of distinct values of the field X
earliest (X) latest (X)	chronologically earliest/latest seen value of X
max (X)	maximum value of the field X. For non-numeric values of X, compute the max using alphabetical ordering.
median (X)	middle-most value of the field X
min (X)	minimum value of the field X. For non-numeric values of X, compute the min using alphabetical ordering.
mode (X)	most frequent value of the field X
percN (Y)	N-th percentile value of the field Y. N is a non-negative integer < 100.Example: <code>perc50 (total)</code> = 50th percentile value of the field <code>total</code> .
range (X)	difference between the max and min values of the field X
<div><div></div><div>00:00</div><div>00:00</div><div>1⚡</div><div>✕</div></div>	
stdevp (X)	population standard deviation of the field X
sum (X)	sum of the values of the field X
sumsq (X)	sum of the squares of the values of the field X
values (X)	list of all distinct values of the field X as a multi-value entry. The order of the values is alphabetical
var (X)	sample variance of the field X

Index Statistics

Compute index-related statistics.

From this point onward, `splunk` refers to the partial or full path of the Splunk app on your device `$$SPLUNK_HOME/bin/splunk`, such as `/Applications/Splunk/bin/splunk` on macOS, or, if you have performed `cd` and entered `/Applications/Splunk/bin/`, simply `./splunk`.

FUNCTION	DESCRIPTION
<pre> eventcount summarize=false index=* dedup index fields index</pre>	List all indexes on your Splunk instance. On the command line, use this instead: <code>splunk list index</code>
<pre> eventcount summarize=false report_size=true index=* eval size_MB = round(size_bytes/1024/1024,2)</pre>	Show the number of events in your indexes and their sizes in MB and bytes
<pre> REST /services/data/indexes table title currentDBSizeMB</pre>	List the titles and current database sizes in MB of the indexes on your Indexers
<pre>index=_internal source=*metrics.log group=per_index_thruput series=* eval MB = round(kb/1024,2) timechart sum(MB) as MB by series</pre>	Query write amount in MB per index from <code>metrics.log</code>
<pre>index=_internal metrics kb series!="_*" "group=per_index_thruput" timechart fixedrange=span=1d sum(kb) by series</pre>	Query write amount in KB per day per Indexer by each index

Reload apps

To reload Splunk, enter the following in the address bar or command line interface.

ADDRESS BAR	DESCRIPTION
<code>http://localhost:8000/debug/refresh</code>	Reload Splunk. Replace <code>localhost:8000</code> with the base URL of your Splunk Web server if you're not running it on your local machine.
COMMAND LINE	DESCRIPTION
<code>splunk _internal call /data/inputs/monitor/_reload</code>	Reload Splunk file input configuration
<code>splunk stop</code> <code>splunk enable webserver</code> <code>splunk start</code>	These three lines in succession restart Splunk.

Debug Traces


You can enable traces listed in `$SPLUNK_HOME/var/log/splunk/splunkd.log`.

To change trace topics permanently, go to `$SPLUNK_HOME/bin/splunk/etc/log.cfg` and change the trace level, for example, from INFO to DEBUG: `category.TcpInputProc=DEBUG`


Then

```
08-10-2022 05:20:18.653 -0400 INFO  ServerConfig [0 MainThread] - Will
generate GUID, as none found on this server.
```

becomes


00:00

00:00
1
⚡



To change the trace settings only for the current instance of Splunk, go to Settings > Server Settings > Server Logging:

Filter the log channels as above.

Select your new log trace topic and click Save. This persists until you stop the server.



00:00

00:00

1 ⚡





Want to Download All Our Premium Cheat Sheets?

No Problem! Just enter your email address, and we'll send you the PDF versions of all our top cheat sheets.

Email Address

DOWNLOAD

Configuration

The following changes Splunk settings. Where necessary, append `-auth user:pass` to the end of your command to authenticate with your Splunk web server credentials.

COMMAND LINE	DESCRIPTION
<u>Troubleshooting</u>	
<code>splunk btool inputs list</code>	List Splunk configurations
<code>splunk btool check</code>	Check Splunk configuration syntax

00:00

00:00 1 ⚡

✕

<code>splunk _internal call /data/inputs/tcp/raw -get:search sourcetype=foo</code>	Restrict listing of TCP inputs to only those with a source type of foo
License details of your current Splunk instance	
<code>splunk list licenses</code>	Show your current license
User management	

COMMAND LINE	DESCRIPTION
splunk _internal call /authentication/providers/services/_reload	Reload authentication configurations for Splunk 6.x
splunk _internal call /services/authentication/users -get:search admin	Search for all users who are admins
splunk _internal call /services/authentication/users -get:search indexes_edit	See which users could edit indexes
splunk _internal call /services/authentication/users/helpdesk -method DELETE	Use the remove link in the returned XML output to delete the user helpdesk

Capacity Planning

Importing large volumes of data takes much time. If you’re using Splunk in-house, the software installation of Splunk Enterprise alone requires ~2GB of disk space. You can find an excellent online calculator at splunk-sizing.appspot.com.

The essential factors to consider are:

- **Input data**
 - Specify the amount of data concerned. The more data you send to Splunk Enterprise, the more time Splunk needs to index it into results that you can search, report and generate alerts on.
- **Data Retention**
 - Specify how long you want to keep the data. You can only keep your imported data for a
- **Architecture**
 - Specify the number of nodes required. The more data to ingest, the greater the number of nodes required. Adding more nodes will improve indexing throughput and search performance.
- **Storage Required**
 - Specify how much space you need for hot/warm, cold, and archived data storage.

- **Storage Configuration**

- Specify the location of the storage configuration. If possible, spread each type of data across separate volumes to improve performance: hot/warm data on the fastest disk, cold data on a slower disk, and archived data on the slowest.

We hope this Splunk cheat sheet makes Splunk a more enjoyable experience for you. To download a PDF version of this Splunk cheat sheet, [click here](#).

Frequently Asked Questions

⊖ What is the Splunk tool used for?

The purpose of Splunk is to search, analyze, and visualize (large volumes of) machine-generated data.

⊕ What are the components of Splunk?

⊕ What are the basic commands in Splunk?

⊕ How many commands are there in Splunk?

⊕ What are Splunk queries?

⊕ How do I write a Splunk query?

⊕ How do I find a specific word in Splunk?

⊕ How do you pull logs from Splunk?



00:00

00:00

1 ⚡

×

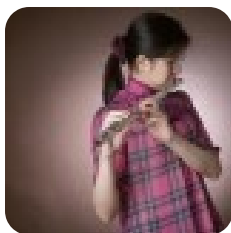
Grow your Cyber Security Skills



FIND OUT MORE

CATEGORIES

CHEAT SHEETS



Cassandra Lee

I make connections across disciplines: cyber security, writing/journalism, art/design, music, mathematics.



00:00

00:00

14



and I'm honored to join students. You can find me on [LinkedIn](#) and [Twitter](#).

Related Articles

NMAP CHEAT SHEET



Nmap Cheat Sheet 2023: All the Commands, Flags & Switches

[Read More »](#)

LINUX CHEAT SHEET



Linux Command Line Cheat Sheet: All the Commands You Need

[Read More »](#)

WIRESHARK CHEAT SHEET



Wireshark Cheat Sheet: All the Commands, Filters & Syntax

[Read More »](#)

COMMON PORTS CHEAT SHEET



Common Ports Cheat Sheet: The Ultimate Ports & Protocols List

[Read More »](#)



00:00

00:00

1 ⚡



INFO

[Affiliates](#)

[Legal Notices](#)

[Privacy Policy](#)

[Site Map](#)

SECURITY ASSESSME NT

[Penetration](#)

[Testing](#)

CONSULTI NG

[Audit &](#)

[Compliance](#)

[Incident](#)

Vulnerability

Response

Scanning

Security

Build Reviews

Architecture

Source Code

Risk Assessment

Review

Security Training

Social

Engineering



COPYRIGHT © 2023 STATIONX LTD. ALL RIGHTS RESERVED.



00:00

00:00

1 ⚡

