



## **Splunk User Manual**

**Version: 3.1**

**Generated: 11/22/2009 04:15 am  
Copyright Splunk, Inc. All Rights Reserved**

# Table of Contents

<b><u>About Splunk</u></b> .....	<b>1</b>
<u>About Splunk</u> .....	1
<b><u>Tutorial</u></b> .....	<b>2</b>
<u>Introduction to Splunk</u> .....	2
<u>Navigating search results</u> .....	6
<u>Event types and punct:</u> .....	10
<u>Alerting</u> .....	13
<u>Reporting</u> .....	14
<u>Using search commands</u> .....	17
<u>Command line interface (CLI)</u> .....	20
<u>Sharing</u> .....	23
<b><u>Reference</u></b> .....	<b>26</b>
<u>Search syntax overview</u> .....	26
<u>Search</u> .....	28
<u>Search modifiers</u> .....	33
<u>Core search fields</u> .....	42
<u>Search fields</u> .....	43
<u>Search commands</u> .....	47

# About Splunk

## About Splunk

### About Splunk

Splunk is an IT Search engine. It is software that indexes any format of IT data from any source in real time, including logs, configurations, scripts, code, messages, traps, alerts, activity reports, stack traces and metrics from all of your applications, servers and devices. Splunk lets you search, navigate, alert and report on all your IT data in real time using an AJAX web interface.

This manual will teach you to use Splunk. See the installation manual and administration manual for guidance on installing, configuring and managing Splunk.

You can also share knowledge and Splunk solutions with other members of the Splunk community via SplunkBase.

# Tutorial

## Introduction to Splunk

### Introduction to Splunk

See a few simple searches in action.

### Requirements

- A supported browser (See system requirements and release notes).
- A copy of Splunk:
  - ◆ On an individual machine, or elsewhere in the organization. (Download and install.)
  - ◆ On Splunk hosted demo servers.

### Log in

Splunk does not require a login when using a Free license. An Enterprise license requires a login:

- For an individual server, the default Splunk username is **admin** and password is **changeme**.
- For the Splunk hosted demo server, the username and password are **guest** and **guest**.

### Index data

Splunk comes with pre-indexed sample data. For help indexing your own data, see the data inputs section of the Admin Manual.

### Simple searches

To start, enter your search in the search box at the top of the page.

**Important:** Throughout this tutorial sample data will be used (in an index called "sampledata"). This index can be searched instead of the main index by including "index::sampledata" in the search criteria.

Start by searching for all the data in the sampledata index. Type the following into the search box.

```
index::sampledata
```

Image:Search 101-Picture 1.png

The results include a timeline that shows exactly when matching results occurred. If there are no results, change the time range from last 24 hours to last 7 days.

Image:Search 101-Picture 2.png

Now, try this simple search:

```
index::sampledata http 500
```

This shows HTTP requests resulting in an internal server error. Notice that there's a typeahead list to help guide you.

Image:Search 101-Picture 6.png

Searches are typically case-insensitive. Exceptions are noted through this tutorial

#### **Click on results**

Click on results to add or remove search terms. For example, click on "500" in one of the search results. Splunk highlights and updates the search to remove "500" as a term -- so now your search results include all http events. This works both ways. Click on "500" again to add it back to the search string.

#### **Scroll through results**

Search for:

```
index::sampledata http
```

Scroll through the results list. Splunk displays more and more results. The red line in the timeline indicates where you are in time.

### Narrow results

Narrow your search results by refining your search. Here are a few tips.

- Alt-click on "200" in any search result (option-click for Mac, shift-click for some popular \*nix windows managers).

Image:Search 101-Picture 7.png

Your search has now been updated with "NOT 200". Splunk supports other Boolean operators too.

Image:Search 101-Picture 9.png

### Follow a relationship

- Ctrl-click on an IP address (cmd-click on a Mac).
- Check "wrap results" to turn on line-wrapping for the long single line events that result from searches.

Your search has been replaced with the IP address. This is an effective way to follow relationships between events.

### Change the time range

If you are using the sample data with the out of the box settings, the timeline shows a cluster of events in a single hour over the last 24 hours.

Click on the bar in the timeline showing the cluster of data, then click **Zoom in**. Any bar in the timeline can also be clicked to zoom in.

Image:Search 101-t1.jpg

Zoom in until there is a narrow enough time range to see only a few bars of data.

Image:Search 101-t2.jpg

Shift-click or drag your mouse across all of the bars and zoom in further.

Image:Search 101-t3.jpg

The timeline should now span several minutes, with one bar equal to one second.

Image:Search 101-t4.jpg

You can also change the time range by using the drop-down located near the upper left of the graph area.

Choose **custom** to specify a start and end time.

**Note:** Your Splunk instance will perform faster if you narrow the time range of your search. Searching over **all time** may result in slow search performance.

#### Boolean searches

Enter the search:

```
> index::sampledata http AND 500
```

Your results will be the same as the previous example search. Splunk implicitly inserts an AND between terms, similar to Google and other search engines. Splunk supports the booleans: AND, OR, and NOT (must be capitalized).

**Note:** There should be three results. If your search produces no results, the time range may still be set from the previous example search. Zoom out, clear the time range or reset it using the time range drop-down menu.

```
index::sampledata http NOT 500
```

All http requests that do not contain error code 500 (internal server error).

```
index::sampledata http NOT (500 OR 503)
```

All http requests that do not contain error code 500 or 503 (service unavailable).

Splunk is able to group Boolean expressions using parentheses.

#### Save a search

To save any search, click on the down arrow to the left of the search box and select **Save search...** from the menu.

Image:Search 101-Picture 10.png

Once you save a search, reuse it by typing `savedsearch::nameyougaveyoursearch` into the search box, or select it from the **Saved searches** menu next to the search box. Search names are case-sensitive.

Image:Search 101-Picture 11.png

## Navigating search results

#### Navigating search results

Navigating search results by following links and using interactive filters is a major component of the Splunk. Filtering is an efficient method to organize the results of a search. The following sections will illustrate some of the navigation features of the SplunkWeb interface.

Run a search for all of the `sampledata` index events.

```
index::sampledata
```

#### Filter on host, source, and sourcetype (search fields)

There are a number of menus below the time graph in the SplunkWeb. These are interactive field filters. By default host, source, and sourcetype are shown.

#### Host

Host shows the originating host of the event. This field enables the targeting of one specific host in the filter. "host::" is stored and indexed alongside each raw event and can be used as a search term. Opening the host menu item shows the top 10 hosts that are in the search results.

Mousing over a host will cause the time graph to show a darker shade illustrating the volume of events for each time period for only that host.

Image:More searching-Picture 2.png

Image:More searching-Picture 1.png

Select any host and the search results will be filtered to show only results for the selected host.

Open the host menu again and select another host.

Image:More searching-host.jpg

Open the host menu once more and select "Add filter to search". This will update the search to include `host::nameofhost`. The restriction of `host::nameofhost` will be applied to the set of search results.

Image:More searching-host2.jpg

#### **Source**

Source is the file, network port, script, or other location where the event was accessed. The source filter menu and host menu function identically. "source::" can be searched just like "host::" can.

Image:More searching-source.jpg

#### **Sourcetype**

A sourcetype categorizes all sources that have similar formats. For example, all apache access logs in W3C common format are given the sourcetype name "access\_common". The sample data contains four distinct sourcetypes - syslog, access\_common, db2 and websphere\_activity. "sourcetype::" can be searched just like any of the other types.

Image:More searching-sourcetype.jpg

### Showing more fields

Additional fields can be searched on besides host, source, and sourcetype.

Execute a search for the http access logs.

```
index::sampledata sourcetype::access_common
```

Select "Fields" to open a list of fields to be able to search on.

Image:More searching-Picture 3.png

### Search fields vs extracted fields

A field is a name/value pair. A field is distinguished from the free-form indexed segments seen in an event. Fields can be categorized by how and when they are processed. Two major categories are: search fields and extracted fields.

*Search fields* are captured in real time as events are processed by Splunk. Information on where the event came from, what type of event, source type, etc, are built into the Splunk input processor. Additional fields can be added for indexing.

*Extracted fields* are created at search time. Splunk picks out obvious name/value pairs in search results. This dynamic extracted field list can be used in filters and reports. Splunk can be trained to recognize additional fields and assign normalized names to the fields.

### Filter on extracted fields

Add a filter on an extracted field. Add this filter to the search.

Image:More searching-Picture 6.png

Notice that instead of adding the field name and value to the main part of the search, Splunk adds a pipe ("|") symbol then a new filter operator. Extracted fields cannot be searched like ordinary event terms because they are not indexed -- they are extracted at search time.

```
sourcetype::access_common | filter method="GET"
```

### **Related events searching**

Splunk can search for related events based on statistical analysis of term frequencies.

Selecting "related" next to an event will cause Splunk to extract search terms from the current search, and perform another search yielding the results similar to the event by using statistical analysis of the term frequencies.

### **Taking snapshots**

Snapshots allow for search results to be saved as a "snapshot". Collections of snapshots can be added to a single snapshot collection. Steps can be retraced by reverting to an earlier snapshot.

Image:More searching-snapshot.jpg

### **SplunkWeb (interface) customization**

Default behaviors of the SplunkWeb (Splunk interface) can be changed through the "Preferences" menu at the top right-hand corner of the interface. Splunk licensed with a free license will save the settings for everyone, and Splunk Enterprise will save changes per individual login accounts.

#### **General preferences**

Changes general interface settings.

#### **Theme**

Allows the selection of a black background theme.

#### **Click behavior**

In earlier examples in the tutorial things were being added to the search by clicking, and being replaced by holding down the ctrl/cmd key while clicking. This is new to Splunk 3.0. In previous versions, selecting a filter was done by alt-ctrl-click. The click behavior preference allows the selection of either method.

## Search preferences

These preferences change parameters for your searches and the display of results.

### Default time range

This will be the default time range for all searches initiated from the home page.

### Maximum results per search

This is the maximum number of results that will be returned from any search unless a different `maxresults::` setting is explicitly added within the search terms. *High max results may cause timeouts and may run into browser limitations.*

### Segment selection

Segment selection governs how mousing over events highlights segments within the results. Choosing "Full" will enable the mouse select from left to right on an IP address and select "192", "192.10", "192.10.20" or "192.10.20.30" to search for anything from the first quad to the full IP address. "Inner" results will come back a tiny bit faster and enable the ability to click to search for "192", "10", "20" or "30" but not "192.10.20.30". "Outer" results will be faster but limits click to search on the whole IP "192.10.20.30".

"Pyramid" is only useful for debugging. If selected, details on how Splunk segments events can be seen.

## Event types and punct::

### Event types and punct::

#### What are event types?

Event types allow you to classify events that have common characteristics. All sshd logins can be an event type. All sendmail syslog messages can be another. Editing, tagging and renaming event types is a big way that a Splunk server gets smarter over time by incorporating the knowledge of its users.

#### What is punct::?

Because the format of an event is often a powerful part of defining an event type, Splunk indexes the punctuation characters of events as an indexed field called "punct." This field, while it looks cryptic at first, is a powerful way of finding similar events quickly.

### Find similar events with punct::

Start by doing a search for all events in your sampledata index:

```
index::sampledata
```

Now, activate the punct:: field by checking it in the fields... menu and selecting Apply.

Image:Event types and punct-Picture 1.png

Filter on each of the first few most common "punct" values.

Image:Event types and punct-Picture 2.png

You'll notice that the events sharing a punct value are strongly similar. This is a fast way of inventorying all of the results of a search with thousands of matches.

Add a filter to your search for the most common punct:: value in your sample data, which should be `..._--_[::]_\"_/?=/_\\.\"_``. Then find access\_common events with URLs similar to `"/trade/app?action=portfolio"`:

```
index::sampledata "punct::..._--_[::]_\"_/?=/_\\.\"_`" sourcetype::access_common
```

### Saving event types

You'll notice that the events you are looking at now are all clearly web requests to the same application but include a mix of different actions - portfolio, home, logout. (This data sample came from an online stock trading application.)

Add "logout" to your search to find just the logout events.

```
index::sampledata "punct::..._--_[::]_\"_/?=/_\\.\"_`" logout
```

Now save your search, but this time, instead of choosing "Save search" from the menu, choose "Save as event type."

Image:Event types and punct-Picture 3.png

Call your event type "trade\_app\_logouts".

(Note: if your event type name contains spaces, upon saving the spaces will be replaced with underscores and the tags will not be saved. It is recommended that you do not include spaces in the eventtype names to avoid this behavior.)

Image:Event types and punct-Picture 4.png

**View and search for event types**

Now add "event types" to your filters via the fields menu.

Image:Event types and punct-Picture 5.png

Image:Event types and punct-Picture 6.png

You'll see the event type name "trade\_app\_logouts" appear underneath each event. You can now click on it to search for "eventtype::trade\_app\_logouts." You can also click on the arrow to the left of the event type tag to look up the event on SplunkBase, search for related events or show the source of the event.

Image:Event types and punct-Picture 7.png

**Automated event type discovery**

Splunk automatically discovers event types based on seeing a large number of events sharing common characteristics. You can edit, delete, rename and tag event types that Splunk discovers for you. You can also make your own event types by saving any search as an event type. Splunk allows you to change the settings that determine which keywords are considered in event type discovery in its eventdiscoverer.conf configuration file.

## Tagging

### Event type tagging

You can tag event types that have very different search terms and punctuation patterns with common words, then find all events of types that have any tag. This is a great way to create higher level classifications like "logouts" that cross different logout event types from different applications.

### Host tagging

Tagging hosts is useful for knowledge capture and sharing, and for crafting more precise searches. Hosts can be tagged with one or more words describing their function or type, enabling users to easily search for all activity on a group of similar servers. See the section in our Admin manual on Host tagging.

# Alerting

## Alerting

Any search that you save can be scheduled and turned into an alert.

### Save a search

Let's save our last search from the previous tutorial, which was a search for:

```
> index::sampledata eventtype::trade_app_logouts
```

### Schedule it

Choose menu command Save Search. In the save search dialog, select "Schedule & Alerts."

Image:Alerting-Picture 1.png

Select run this search on a schedule and define the schedule using either the dropdown, or by entering a more precise schedule using cron notation in "Advanced scheduling."

### Set alerting conditions

You can define alerting conditions based on thresholds and deltas in the number of events, sources and hosts in your results.

### Set the alerting method

You can get alerts via RSS and email. You can also trigger a shell script, such as a script to generate an SNMP trap or call an API to send the event to another system. If you need additional email options (like setting the From: address) see the Alerts page in the Developer manual.

### Permalink your saved search

You can share any search with other users by creating a Permalink. To create a Permalink for any search:

- Click the search bar drop-down menu.
- Click **permalink** to create a Permalink URL in your browser's URL text bar.
- Share the Permalink by copy and pasting it to other users.

**Note:** Splunk doesn't Uuencode its Permalink URLs. Some browsers may experience problems resolving Permalinks if they aren't Uuencoded.

### Manage your saved searches and alerts

We've set up a number of saved searches and alerts in this tutorial. If you want to delete them or change them later, click the drop-down arrow on the left-hand side of the search bar, select "saved searches", and then select "manage saved searches". This will take you to the manage saved searches screen where you can edit your saved searches.

You can display saved searches on the dashboard either by selecting the dashboard from the Save Search dialog box when you create it, or selecting the dashboard from the drop-down menu on the home page and clicking Edit. Select the saved searches you'd like to see in the dialog box and click Apply.

## Reporting

### Reporting

Splunk lets you summarize the results of any search as a report, which can include both a chart and a table of results.

Let's start with a search for some firewall deny events.

```
index::sampledata deny
```

#### Report on a field

Open up the **fields...** menu and select all available extracted fields. You'll see that Splunk has done a good job of figuring out all the fields in the well-structured Netscreen firewall log format.

Image:Reporting-Picture 8.png

Once all of your fields are showing, use the **filter** dropdown on the field **src** and click the **Report on this field** link.

Image:Reporting-Picture 2.png

You are taken to the Report tab and Splunk shows you the count and percent of events matching your search along with a bar chart graphing the results. Note how Splunk has changed your search to:

```
index::sampledata deny | top src
```

We'll cover pipes and operators later in this tutorial.

#### Build a new report

You can also build a new report within the report tab. To do this, search for all of the access\_common data in the sample index:

```
index::sampledata sourcetype::access_common
```

Select the field **byte** from the **fields** menu to the left.

In the parameters section enter "Show sum of byte vs time split by action." Then click **Apply**.

Image:Reporting-Picture 3.png

**Pick a different chart**

Change chart styles by selecting a type from the **display as** drop-down menu above the current chart.

The types of charts that are available:

- column
- line
- area
- scatter
- stacked column
- stacked area
- pie
- donut
- bubble
- heatmap

To see a gallery of samples of some of these charts see the report gallery on our website.

You'll notice that the portfolio action has the largest spikes.

**Add it to your dashboard**

You can save a search from report mode just as you would any other search. When you save the search, add it to your default dashboard by checking the box at the bottom of the save dialog.

Image:Reporting-Picture 7.png

You'll see the report on the dashboard after clicking the logo to return to the home page. Dashboard searches are refreshed every tenth of the time interval (for example, a 4 hour search every 24 minutes) or every hour, whichever is shorter.

**Note:** You won't see your report on your dashboard if you haven't loaded any data to your main index yet. As soon as you have data in your main index, the "getting started" links are replaced with a default dashboard including modules that are predefined in the product, plus additional searches and

reports you've added.

## Using search commands

### Using search commands

Now you're ready to start typing your own variants on the cool searches you've been seeing Splunk paste into your search box. Try these variants in any order. Just paste these examples into your search box to illustrate the power of Splunk search commands.

#### **timechart**

timechart returns statistics bucketed by time and is good for driving line charts. Try these examples.

Count of deny events graphed by time.

```
index::sampledata deny | timechart count(_raw)
```

Sum of bytes for GET requests by 5 second bucket.

```
index::sampledata sourcetype::access_common GET | timechart span=5s  
sum(bytes)
```

Average bytes by method by 10 second bucket.

```
index::sampledata sourcetype::access_common | timechart span=10s  
avg(bytes) by method
```

**stats**

stats provides summary calculations by any field. Try it with a pie chart.

Total bytes sent by destination.

```
index::sampledata sourcetype::syslog | stats sum(sent) by dst
```

**top**

Let's get the top denied source IPs. This will work best with a bar chart.

```
index::sampledata netscreen deny | top src  
(By default top brings back 10 results.)
```

**rare**

You can also get the rarest 100 source IPs (by using rare).

```
index::sampledata netscreen deny | rare 100 src  
where
```

Let's go back to our top source IPs and filter for ones with more than 5 denials by using the where command.

```
index::sampledata netscreen deny | top 100 src | where count > 5  
fields
```

Let's display only the src field now (using fields).

```
index::sampledata netscreen deny | top 100 src | filter count > 5 | fields  
src  
sort
```

We can sort the results using the sort) command.

```
index::sampledata netscreen deny | top 100 src | where count > 5 | fields  
src | sort src
```

(Note this works with individual events in cooked and raw mode too.)

### **Subsearches**

Now we're going to put it all together by doing another search to find which of the actions with more than 2 500 http status codes also had 200 successes (i.e. intermittently failing actions on our application.)

```
index::sampledata 200 [search index::sampledata 500 | top action
```

```
| where count > 2 | fields + action]
```

**diff**

Do a search for errors in db2 and diff the first two results. When you use the diff command with no arguments, the first two results are compared by default.

```
index::sampledata error sourcetype::db2_diag | diff
```

Compare the host field of the 3rd and 4th results.

```
index::sampledata error sourcetype::db2_diag | diff 3 4
```

Now, find the amount of time between two events by comparing the values of the date\_time field.

```
index::sampledata error sourcetype::db2_diag | diff 3 4
```

```
attribute=date_time
```

**set**

Return all urls that have 404 errors but no 303 errors (using set).

```
set diff [search 404 | select url] [search 303 | fields url]
```

**regex**

The regex command is useful in removing results from your search results. Use a regular expression in regex to remove results that do not match the regular expression. Regex is useful in finding regular expressions in search results.

Note: if you want to use the "or" ("|") command in a regex argument, the whole regex expression must be surrounded by quotes (ie. regex "<expression>").

Try the example below:

```
sendmail | regex _raw=(?!\d)10.\d{1,3}\.\d{1,3}\.\d{1,3}(?!\d)
```

Get sendmail events that contain ip addresses in the non-routable class A (10.0.0/8).

**Note:** Splunk's regex command supports inclusion of PCREs (Perl Compatible Regular Expressions).

## Command line interface (CLI)

### Command line interface (CLI)

Splunk includes a command line interface (CLI) that runs from a shell on the server host. It's a great way to integrate Splunk into admin scripts. To use Splunk from the command line, add the `./bin` subdirectory of your Splunk installation to your shell path.

For example:

```
export PATH=/opt/splunk/bin:$PATH
```

**Note:** CLI searches do not include a default time range.

### Examples

Below are typical commands that you could execute from the CLI.

**Note:** CLI commands must be prefaced by the `"splunk"`, unless you have logged into the Splunk CLI.

For example:

```
./splunk search "foo"

search "session root daysago::1"
add tail /var/applog -sourcetype myApp
remove tail /var/log
list tail
spool /my/random/logs.tgz -sourcetype linux_messages_syslog
add batch /var/archive -segmentnum 3
add udp 514
edit udp 514 -sourcetype asterisk_event_syslog
add user -role power -username gwb -full-name "George W Bush" -password changeme
add forward 10.1.1.123:8089
enable receive
add search-server splunk03:8089
list savedsplunk
help commands
```

## Built-in help

The CLI supports the same search syntax as the search bar in SplunkWeb, and commands work the same way (except some reporting commands). The best way to learn to use the CLI is to use its built-in help system. Access it by typing:

```
./splunk help
```

## Basic commands

Below is actual help output from command line interface.

```
# splunk help
Welcome to Splunk's command line interface. Try typing these commands for more help.

help simple, cheatsheet          list common commands, and command line syntax
help commands                  full list of command line commands
help [command]                 type a specific command for its own help page
help [object]                  type a specific object for its own help page
help [topic]                   type any topic to get help on or related to it
help datastore                 manage Splunk's local filesystem use
help distributed                manage distributed configurations such as: data c
and distributed search
help forwarding                manage data forwarding deployments
help input, inputs             manage data inputs
help control, controls         tools to start, stop, manage Splunk processes
help settings                  manage settings for your Splunk server
help tools                     tools to help your Splunk server
help training                  train Splunk to recognize dates, source types, or

help search                    help with Splunk searches
Universal Parameters:
These parameters are usable by any command. Type "splunk help [auth|uri]" for details on
Syntax:

[command] [object] [-parameters]... [-uri][-auth]
help auth                      authentication for commands, can be applied to any command
help uri                       used to send a specified command to a specified server

Try typing "help [object|topic]" to get help on a specific object, or topic you are curious
# splunk help simple
This page shows will get you started with some basic commands, examples of usage,
and a list of help commands for reference on Splunk's search.
Splunk command line syntax:
./splunk [command] [object] [-parameter value]...
These are the basic Splunk commands you need to know:
search                        search a Splunk index
login,logout                  authenticate a session to a Splunk server
start,stop,restart,status    manage Splunk processes
spool                         load a file or directory into an index
add,edit,remove,list         manage data inputs, user accounts, saved searches
set,show                     manage Splunk settings
enable,disable                turn features on and off
```

help	show main help page
install,upgrade	install or upgrade a bundle
refresh	update a deployment server with client server information
reload	reload deployment servers

Examples of typical commands:

```
./splunk search "session root daysago::1"
./splunk add tail /var/applog -sourcetype myApp
./splunk remove tail /var/log
./splunk list tail
./splunk spool /my/random/logs.tgz -sourcetype linux_messages_syslog
./splunk add batch /var/archive -segmentnum 3
./splunk add udp 514
./splunk edit udp 514 ?sourcetype asterisk_event_syslog
./splunk add forward 10.1.1.123:8089
./splunk enable receive
./splunk add search-server splunk03:8089
./splunk help commands
```

Splunk search cheatsheets:

help search	search syntax reference
help search-modifiers	complete list of search modifiers and usage examples
help search-fields	complete list of search fields indexed by Splunk
help search-commands	complete list of search commands

Type "help [object|topic]" to get help on a specific object, or topic.

# splunk help commands

All Splunk commands take the form:

```
./splunk [command] [object] [-parameter value]...
```

Some commands don't require an object or parameters.

Some commands have a default parameter that can be specified by its value alone.

Supported commands and objects:

[command]	[objects]
add,edit	[bundle blacklist deploy-client deploy-class exec fifo forward-server saved-search search-server source sourcetype tail tcp udp u]
anonymize	source
clean	[eventdata globaldata userdata all]
disable	[bundle discoverable dist-search deploy-client deploy-server listen local saved-search search-server source sourcetype tail tcp udp u]
enable	[bundle discoverable dist-search deploy-client deploy-server listen local saved-search search-server source sourcetype tail tcp udp u]
display	[discoverable dist-search deploy-client deploy-server listen local saved-search search-server source sourcetype tail tcp udp u]
export,import	[globaldata userdata eventdata]
find	logs
help	NONE
install,upgrade	bundle
list	[bundle blacklist deploy-client deploy-class exec fifo forward-server saved-search search-server source sourcetype tail tcp udp u]
login,logout	NONE
recover	NONE
reload	deploy-server
remove	[bundle blacklist deploy-client deploy-class exec fifo forward-server saved-search search-server source sourcetype tail tcp udp u]
resurrect,unresurrect	[archive_directory index from_time end_time]
search	NONE
set,show	[datastore-dir deploy-multicast deploy-poll default-hostname default-hostname minfreemb servername server-type splunkd-port web-port]
spool	[globaldata userdata eventdata]
start,stop,restart	[splunkd splunkweb monitor]
status	[splunkd splunkweb monitor]

Type "help [object|topic]" to get help on a specific object, or topic.

# Sharing

## Sharing

Splunk provides useful ways to share knowledge and information. You can create new eventtypes, save them, and put them into Splunk bundles. You can create saved searches and schedule alerts. And you can tap into SpunkBase and search for help, share your experiences, or share your bundles with the Splunk community.

### Creating, tagging, and sharing eventtypes

#### Creating an eventtype

1. In the SplunkWeb user interface: create a search to save for the event type that you want to make.
1. Click the down arrow next to the search bar.
1. From the drop-down menu, choose "Save As Event Type".
1. In the name text box, enter a descriptive name for the event type.
1. Apply a tag to your eventtype (see below).
1. Click Save.

To use your saved eventtype, start a search with:

```
eventtype::
```

#### Tagging an eventtype

Tagging is useful when sharing an eventtype. You can assign tags to the new eventtype in the Tags text box before you save your created eventtype.

(You can make changes to the search at any time. Just make sure to run your changes through the search and re-save each time.)

#### Sharing an eventtype

To share saved eventtypes, you'll have to make a bundle. The Admin Manual will have a more advanced explanation of bundles, and how to make bundles. For now we'll go through a simple explanation on how to create a bundle.

## Saved searches

You can save searches like you save eventtypes. Saved searches allow you to create alerts for certain events, or amounts of a certain event based on a threshold value. Alerts tied to saved searches allow you to trigger events such as a scripts, sending an email, or even trigger an RSS feed.

## SplunkBase

Full information on SplunkBase can be found in the Admin Manual. For our purposes as users, the SplunkBase is a helpful community to obtain answers from Splunk professionals, or other Splunk users. SplunkBase is also where you can share your bundles, or obtain useful bundles from other members of the community. Any content available in SplunkBase is findable through searches, as well as through the site's menus.

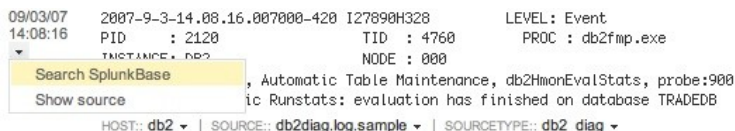
## Looking up events

You can look up any event on SplunkBase. This is a helpful tool for gaining more insight into various events you might not be so familiar with.

From within the Splunk interface, click on the drop-down arrow underneath the timestamp:



You will see an option to *Search SplunkBase*:



Click this link and you will be redirected to the SplunkBase page associated with that event.

## Getting Q & A

A large part of SplunkBase is devoted to Questions & Answers. You can focus the Q&A around your needs by using the Categories list on the left to narrow down to the technology you're interested in.

Then, click a Question to see the list of Answers associated with it.

#### **Using the How-to guides**

Another large section of SplunkBase is devoted to HOWTOs. HOWTOs are documents that explain how to understand or accomplish something. Just as with Q&As, you can focus onto the technology you're looking for by using the Categories links on the side. Click through a HOWTO's name to see its contents.

#### **Add-ons in SplunkBase**

SplunkBase is teeming with bundles you can add to your Splunk installation. From the Add-ons page, you can narrow down the listing either through Categories, or through Types (the types of content within the add-on). Click through a bundle's name to read more about it, see ratings and comments, rate it yourself, view what's inside it by clicking View Contents, or download it by clicking the Download button. Once you have a bundle downloaded, to add it to your Splunk instance, place it into your Splunk server's `$SPLUNKHOME/etc/bundles` directory and extract the tarball or zip file.

# Reference

## Search syntax overview

### Search syntax overview

A Splunk **search** consists one or more commands and their arguments. Any search must have at least one **data-generation command**. The data generated can then be used by other commands in a **search pipeline**.

A **data-generation command** is a command that generates data. The most common **data-generation command** is `search` which retrieves data persisted in a Splunk index, but there are other **data-generation commands** as well. The `remote`, `savedsearch`, and `run` commands all generate data that can be used in Splunk **searches**.

`search` is made up of a search statement followed by **search arguments**. **Search arguments** alter the results of the **search** by specifying what data to operate on, what additional instructions to follow at runtime, and what indexed terms to **search** for specifically. The **search argument** can contain a **subsearch**.

The **search pipeline** is composed of commands and arguments that process data generated by the `search` command and the other data-generation commands.

**Search arguments** alter the results of the **search** by specifying what data to operate on, what additional instructions to follow at runtime, and what indexed terms to **search** for specifically. The **search argument** can contain a **subsearch**. **Search arguments** can be literal keywords, wildcards, Boolean expressions, `search field="value" pairs`, `modifier="value" pairs`, and **subsearches**.

The **search command** arguments are defined in later sections.

### Syntax definition

```
search ::= data_generation_command [ search_pipeline ]
```

```
data_generation_command ::= search_command | remote_command | savedsearch_command | run_command
```

```
search_pipeline ::= [ command ] [ search_pipeline ]
```

**search\_command** ::= search [ **search\_argument** ]

**search\_argument** ::= [keywords] [indexed\_field="value"] [modifier="value"] [search\_command]

**subsearch** ::= search\_command [" search " ]

**remote\_command** ::= remote [server", "...", "server] | [server" "... "server] [search\_pipeline]

**savedsearch\_command** ::= savedsearch [name of saved search]

**run\_command** ::= run [run argument]

#### Syntax for subsearches

A **subsearch** is a **search** with a search command as an argument.

**search\_command** ::= search **search\_argument**

**search\_argument** ::= keyword "|" search\_field=value "|" modifier="value" "|" **subsearch**

**subsearch** ::= **search\_command** [" [ search ] " ]

The **search command** is made up of a search statement followed by **search arguments**.

#### Tuning search performance

Splunk's searches are optimized for text-based searching of raw event data. Search speed is dependent on how your Splunk install is configured. You can improve the speed of your searches by editing the configuration files, and by downloading various add-ons through SplunkBase. Read more

about tuning search performance here.

## Search

### Search

Searching is easy - type any term you'd expect to find in your data into the search box and click or press Enter. For example, to search for all events containing a given IP address, type it into the box. Try it with usernames, error codes, transaction IDs, or whatever else you are looking for.

You can use Boolean ORs and NOTs and combine them with parentheses. Use wildcards or quote marks to find phrases. Splunk also lets you search for fields like host, source and sourcetype. Use "fieldname::value" to search for fields.

```
host::web1 source::/var/log/httpd/access.log sourcetype::access_common  
10.1.2.4
```

Get a full list of search fields.

You can pipe search results to a variety of powerful commands to perform statistics and structured analysis on the results. These commands can use the search fields and more extracted fields that Splunk discovers in your results. Check **Fields** under your timeline to see a list of available fields.

top returns the most frequent values of any field in your search results along with a count and percentage.

where filters your results using a SQL WHERE clause expression.

```
"Password accepted" | where user="mary"
```

Browse a complete reference of search commands.

You can pipe as many commands together as you like and even combine one search with another for more advanced correlation. The below search will find all hosts that have more than 10 "password failed" events and also have "password accepted" events. Square brackets with the addition of the command "search" after the opening bracket let you embed a second search inside your first.

```
"Password accepted" [search "Password failed" | top host | fields +host |  
format]
```

## Keywords

Keywords are not case-sensitive. Just type in any keyword that you want to search for.

Depending on how your Splunk's segmentation is configured, certain special characters may not be allowed in your keyword searches.

Examples:

```
1. 1.2.4
```

Searches for "10.1.2.4".

```
err*
```

Searches for any text with "err" present.

```
"my error"
```

Searches for exactly "my error".

## Wildcards \*

Wildcards may be placed at the start, end or middle of keywords, or at the end of modifier terms.

The following are examples of valid wildcard usage:

- foo\*
- \*foo
- f\*oo
- \*foo\*
- \*f\*o\*o\*
- /var/log/\*

## Searching for ""

In Splunk version 3.0 and above, you cannot search literally for \*. To search for \* you must first search for all "", and then filter the search using a regex.

- ◆ | regex \_raw = \\*

### "Quotation marks"

You must use quotation marks to search for any string that contains quotation marks, whitespace, the pipe character, open or closed parenthesis or brackets.

### Correct usage examples:

Example	Matches
"foo"	foo
"username=Joe Smith"	username=Joe Smith
"authentication denied"	authentication denied
"[bar"	[bar
"foo * bar"	foo * bar
" 404 "	404
"(404)"	(404)

### Incorrect usage examples:

#### Example Matches

- " "404" " Returns nothing.
- " " 404 " " Returns nothing.

#### Punctuation marks

Most punctuation marks such as . , ! % \$ / \ [ ] { } < > @ = + & and # are treated as breaking

#### Booleans

#### AND,OR,NOT

Splunk supports the logic commands AND, OR, and NOT. They must be completely uppercase or they will be treated as regular keywords. XOR is not supported. "AND" is implicit in the search string.

NOTE: Currently phrase searching cannot be used in conjunction with OR and NOT operators. This functionality will be available in a future Splunk release.

Example:

error OR (success NOT login)

### **Parentheses**

Parentheses must have spaces on the outer (convex) side of them.

The following are correct:

```
(foo NOT (bar OR baz) )  
( foo NOT (bar OR baz) )
```

The following are incorrect:

```
(foo NOT(bar OR baz))  
(foo NOT(bar OR baz ))
```

Parentheses must be used when mixing OR and NOT in the same search.

### **Precedence**

Boolean commands are evaluated in this order:

1. ( )
1. OR
1. AND, NOT

### **Fields**

Splunk fields values are assigned to each event by Splunk. They take the format name::value. Each event can have only one value for each fields name. Below are some commonly-used fields:

### **eventtype::**

Event types are defined as saved searches. Create a new event type by choosing "Save as event type..." from the menu. Then search for that event type by entering eventtype::< name of event type >, for example: eventtype::trade\_app\_logouts

## **eventtypetag::**

Event types can be tagged with arbitrary text values such as `tag::failure`. Event type tags are not in beta 1.

## **host::**

The hostname or IP address of the host that generated an event, such as `host::juno.Splunk.com`.

## **hosttag::**

Host values can be tagged to create groupings. For example, each of the host values from production servers could be tagged with the value `hosttag::production`.

## **punct::**

The punctuation pattern of an event, such as `punct::..._ - _ [::]_\"_?=_/\\"_`

## **source::**

The file, network port, or other data stream from which an event was indexed. For example: `source::/var/log/messages`.

## **sourcetype::**

The kind of data identified by Splunk in the event's source. For example: `sourcetype::linux_messages_syslog`.

Splunk automatically creates new fields at search time when it sees name/values pairs in search results. To see these, go to the Fields menu and select More >>. You'll see a list of fields extracted at search time.

See the search fields list for the usage and syntax of fields.

## Modifiers

Modifiers affect a search rather than being search terms themselves. Like meta data, they also take the format name:value. Most do not have default values. Some can only be used once in a search, as noted below, while most can appear several times in the same search with different values. Modifiers can be used before, after or between keywords and Boolean commands. If a search has conflicting modifiers, such as "daysago:1 monthsago:6 daysago:3", the first one from left to right will take precedence. See the Search modifier list section for search modifier details and syntax.

## Subsearches

A subsearch is a search within a search string (delimited by '[' ']') brackets) whose results are passed to the search string its contained in. A search containing a subsearch will execute the subsearch first, and pass the subsearch results to the rest of the search string. By default, a subsearch returns the `_query` field value the top search result. If the first result doesn't contain a `_query` field value, Splunk implicitly calls the `format` command to produce a `_query` field value for each search result in your search.

**Note:** You can nest subsearches within subsearches.

## Example subsearch forms:

```
(search terms) [subsearch string] | ...
```

```
(search terms) [(search terms) [subsearch string] | ...] | ...
```

## Example:

```
sourcetype::access_combined | where [search sourcetype::access_combined | top 4 clientip | fields clientip | format]
```

- Restricts the results from a combined access log to the top four IPs. Implicitly calls the `format` command to produce the `_query` field to output the result of the subsearch.

# Search modifiers

## Search modifiers

Search modifiers are used in the search command, and allow you to modify the results of a search based on time constraints, and other factors. Modifiers are explicitly used within the context of the search command.

There are two types of search modifiers. Search modifiers allow you to specify criteria to narrow your search, and time modifiers that adjust start/stop times and time ranges of your search.

Time modifiers = daysago, enddaysago, endhoursago, endminutesago, endmonthsago, endtime, endtimeeu, hoursago, minutesago, monthsago, searchtimespandays, searchtimespanhours, searchtimespanminutes, searchtimespanmonths, startdaysago, starthoursago, startminutesago, startmonthsago, starttime, starttimeeu, timeformat

Search modifiers = eventtypetag, hosttag, index, maxresults, readlevel, readlimit, related

**Search modifier syntax**

In versions 3.0.x modifiers take the format of:

- modifiername::value

In versions 3.1.x and above, modifiers take the formats:

- modifiername::value
- modifiername="value"
- modifiername=value

Most modifiers do not have default values. Modifiers may appear anywhere in a Splunk command before, after, or in between keywords and logical expressions. If a search has conflicting modifiers, the first one from left to right will take precedence.

#### Search modifier precedence

- Only the first declaration of daysago, hoursago, or minutesago will be evaluated.
  - If there is more than one index modifier in a search command argument, only the first declaration will be evaluated.
  - If there is more than one of the same modifier declared in a search, only the first one will be evaluated.
- 

#### Conventions used in this reference

##### Syntax conventions

**command** *argument* ... [*argument*] ...

- Commands are in bold.
- Any bolded (and not italicized) character in the command syntax is a required term for the expression.
- Required arguments are italicized (and can be bold).
- Optional arguments are in [brackets].
- " ... " means that many arguments can be inserted.
- Arguments are defined in a table.

*argument*=syntax and value(default value) Description, and usage.

- Default values are shown in parentheses ( ).
- Arguments that have a table of options associated with them are italicized and in bold (***argument***).
- " | " is used as a logical OR.
- T | F = True OR False.

#### Other conventions

- Command examples that are applicable to SplunkWeb are shown in a mock-up of a search bar.

foo | top

- Command examples that are applicable to the Splunk command line (CLI) are shown in indented fixed-width font.

```
./splunk search "foo | top"
```

---

#### **daysago**

Search events within the last N days.

#### Syntax

**daysago**=*integer*

#### **enddaysago**

Set an end time (in days) that is = now - number specified.

Syntax

**enddaysago**=*integer*

**endhoursago**

Set an end time (in hours) that is = now - number specified.

Syntax

**endhoursago**=*integer*

**endminutesago**

Set an end time (in minutes) that is = now - number specified.

Syntax

**endminutesago**=*integer*

**endmonthsago**

Set an end time (in months) that is = now - number specified.

Syntax

**endmonthsago**=*integer*

**endtime**

All events must be before the specified time. Use `timeformat` to set the time format to use. For example: if `timeformat=%m/%d/%Y:%H:%M:%S`, then `endtime=09/07/1978:09:00:00`, and all results are before that time.

Syntax

**endtime**=*string*

**hosttag**

Search for events that have hosts that have a matching host tag string.

Syntax

**hosttag**=*string*

**hoursago**

Search events within the last N hours.

Syntax

**hoursago**=*integer*

**index**

Specifies an index to search (main, default, history, splunklogger, or another admin defined index). If there is more than one index modifier in a search command argument, only the first declaration will be evaluated.

Syntax

**index**= "*name of index*" | *name of index*

**maxresults**

Limit the number of results that your search returns by specifying a maximum number of results. The default number of events for any search to return is 10,000.

#### Syntax

**maxresults**=*integer*(10000)

#### minutesago

Search events within the last N minutes.

#### Syntax

**minutesago**=*integer*

#### monthsago

Search events within the last N months.

#### Syntax

**monthsago**=*integer*

#### readlevel

Specifies how much detail is read from events returned from the search processor. This modifier is only useful in command line searches.

#### Syntax

**readlevel**=*level*

#### Arguments

*level*= 0 | 1 | 2 Different read levels to specify.

- 0 Specifies that only the top indexed fields (host,source, sourcetype) get read.
- 1 Specifies to read raw data and 2nd order fields in addition to top indexed fields (read level 0).
- 2 Specifies a full read of event types in addition to all of the above data.

## readlimit

Specify the starting point of events within your results to read and return. By default this is set to 0 (to read all events).

### Syntax

**readlimit**=*integer* | "*integer range*"

- **Example:** `readlimit="20-29"` - Reads events 20-29.

## related

Specifies events that are related to the event of id `event_id`. The value assigned to a related search is a hash value that only makes sense to the server. Related results are sorted by relevance rather than by time.

### Syntax

**related**=*hash value*

- **Example:** `related="0:12345"`

## savedsearch

Search for events that would be found by the specified saved search.

### Syntax

**savedsearch**=*name\_of\_saved\_search*

## searchtimespanminutes

Search within a specified range of minutes (expressed as an integer).

Syntax

**searchtimespanminutes=*integer***

**searchtimespanhours**

Search within a specified range of hours (expressed as an integer).

Syntax

**searchtimespanhours=*integer***

**searchtimespandays**

Search within a specified range of days (expressed as an integer).

Syntax

**searchtimespandays=*integer***

**searchtimespanmonths**

Search within a specified range of months (expressed as an integer).

Syntax

**searchtimespanmonths=*integer***

**startminutesago**

Search the specified number of minutes ago from the present time (expressed as an integer).

Syntax

**minutesago=*integer***

**starthoursago**

Search the specified number of hours ago from the present time (expressed as an integer).

**Syntax**

**hoursago**=*integer*

**startdaysago**

Search the specified number of days ago from the present time (expressed as an integer).

**Syntax**

**daysago**=*integer*

**startmonthsago**

Search the specified number of months ago from the present time (expressed as an integer).

**Syntax**

**monthsago**=*integer*

**timeformat**

Change the format for the starttime and endtime modifiers. All Splunk searches have the default time format of: %m/%d/%Y:%H:%M:%S.

**Syntax**

**timeformat**=*string*

## Arguments

*string* = %m/%d/%Y:%H:%M:%S (default = %m/%d/%Y:%H:%M:%S).

# Core search fields

## Core search fields

Core fields are stored with every event and can be used in the search command. These fields are automatically extracted by Splunk.

### host

Specifies a host to match. The result will return the host that originated the event, as determined by Splunk when it indexed the event being searched.

### Example:

```
host::host.splunk.com
```

### source

Specifies a field value to match either the file, FIFO, network port, database table, or other source from which the event was originally indexed.

### Example:

```
source::/var/log/messages
```

### sourcetype

Specifies a uniquely identified type of data in the source when it was indexed. Source types can be renamed.

### Example:

```
sourcetype::apache
```

# Search fields

## Search fields

Search fields are additional fields that are usable when using the search command in addition to the core fields. Search fields can be applied when using the search command to narrow your searches. The complete search field list is longer than what is displayed in the fields list in SplunkWeb.

### `_raw`

Contains the entire event. `_raw` is the object of the search command, so it cannot be used in a search. However, it can be used by other commands in the search pipeline. In the following example, search results are piped to `regex` command which operates on the `_raw` field.

### Example:

```
[search string] | regex _raw=*10.\d\d\d.\d\d\d.\d\d\d*
```

Get sendmail events that contain an IP address starting with "10".

### `_serial`

A special field containing the serial number of events in a search result. `_serial` cannot be used in a search, but it can be used with a command that works on a search result.

### Example

```
[search string] | where _serial > 100 AND _serial < 200
```

### `_time`

Special field that represents an event's timestamp in UTC seconds. `_time` is used to create the timeline. It cannot be used in a search, however it can be referenced in commands that process searches.

- `_time=` the minimum time for all events.
- `_time+duration` = the maximum time. Duration is the difference between the min and max times.

An example of duration would be in a router log where an interface goes and then comes back up. Minimum time would become the first event alerting that the interface is down, maximum time would be the last event saying that the interface is now up and duration would be how long the interface was down.

### **Example:**

```
[search string]| sort _time, ms
```

#### **date\_hour**

Specifies events from the specified hour from 0 through 23.

#### **date\_minute**

Specifies events from the specified minute from 0 through 59.

#### **date\_month**

Specifies events from the specified month. This value is not displayed in the results, except for the timestamp.

#### **date\_mday**

Specifies events from the specified day of the month. This value is not displayed in the results, except for the timestamp. The value range is 1-31.

### **Example:**

```
date_mday::21
```

#### **date\_second**

Specifies events from the specified second from 0 through 59.

**date\_wday**

Specifies events from the specified day of the week. This value is not explicitly displayed in the results. The value range is: sunday-saturday.

**Example:**

```
date_wday::thursday
```

**date\_year**

Specifies events from the specified year.

**Example:**

```
date_year::2007
```

**date\_zone**

Specifies events from the timezone specified in minutes ahead of UTC. The value range is minutes offset from UTC. The range is: -720 to 720.

**Example:**

```
date_zone::480
```

or

```
date_zone::local
```

**eventtype**

Specifies an event type to match. Event types can either be created by the user, or can be created from the automatic event type discoverer.

**Example:**

eventtype::sendmail\_login\_failure

**eventypetag**

Returns that have specified tags matching the argument.

**Example:**

eventypetag::java

**endtime**

Specifies an end time value that all events must be earlier or equal to.

**Example:**

endtime::12/31/07:04.45.13

**endtimeu**

Specifies an end time in seconds since the Unix Epoch began, 12:00:01 January 1, 1970. See [http://en.wikipedia.org/wiki/Unix\\_epoch](http://en.wikipedia.org/wiki/Unix_epoch).

**Example:**

endtimeu::6278346234283424

**linecount**

Specifies the number of lines each matching event must have. It cannot take expressions (ie. linecount::>40). To search for events within a specified range, OR together multiple linecount:: statements.

**Example:**

linecount::40 OR linecount::41 OR linecount::42

## punct

Specifies a log entry punctuation pattern to match. Patterns of punctuation often correspond to a set of related log files. The string can be made up of the following characters:

`;-#$$%&+./:=?@\|'*\n\r"(){}<>[]^!"`.

### Example:

```
punct::..._--_[//:::_-]_\\"_//?=_/.\"_
```

## starttime

Specifies a start time value that all events must be later or equal to.

### Example:

```
starttime::01/01/2001:01:01:01
```

## starttimeu

Specifies a start time (in seconds) since the Unix epoch began (12:00:01 am January 1, 1970). See [http://en.wikipedia.org/wiki/Unix\\_epoch](http://en.wikipedia.org/wiki/Unix_epoch) .

### Example:

```
starttimeu::9234567891
```

## timestamp::none

Specifies events that did not have any detectable timestamp (i.e., another time rule was used).

## user

Specifies the name of a Splunk user. Used when looking up search history via `index::history`.

## Search commands

## Search commands

Use search commands to generate search results from an index or process search results that get generated. Combine search commands in a search to produce specific sets of search results. Or produce complex reports based on search results (using the "|" to "pipe"/separate commands).

Select search commands from the list below to learn how to use them.

See the search syntax page for a description of the search command pipeline in modified BNF (Backus - Naur Form).

<b>Data-generating</b>	file, remote, run, savedsearch, search
<b>Saving</b>	run, sendemail, outputcsv, outputraw, outputtext, outputxml
<b>Filtering &amp; Re-ordering</b>	page, regex, run, set, sort, uniq, where
<b>Transforming &amp; Reporting</b>	associate, chart, contingency, correlate, diff, format, rare, run, select, stats, timechart, top, xmlunescape
<b>Evaluating</b>	abstract, addtotals, anomalousvalue, bucket, convert, eval, fields, fillnull, kmeans, outlier, rename, replace, run
<b>Extracting</b>	extract(kv), multikv, run, xmlkv
<b>Administering</b>	run, admin

Use **data-generating** commands to get data out of a Splunk index.

**Saving** commands allow you to save data in various formats. Use saving commands to format data for a particular type of output.

**Filtering & Re-ordering** commands don't change data within results. These commands allow you to filter a result set, and re-order how results appear.

**Transforming & Reporting** commands allow you to summarize large result sets.

**Evaluating** commands evaluate each result, and change the fields or values of fields within each result.

**Extracting** commands add fields to results based on raw event data.

**Administering** commands allow you to perform administrative functions.

---

#### Conventions used in the search reference

##### Syntax conventions

**command** *argument* ... [*argument*] ...

- Commands are in bold.
- Any bolded (and not italicized) character in the command syntax is a required term for the expression.
- Required arguments are italicized (and can be bold).
- Optional arguments are in [brackets].
- " ... " means that many arguments can be inserted.
- Arguments are defined in a table.

*argument*= syntax and value(default value) Description, and usage.

- Default values are shown in parentheses ( ).
- Arguments that have a table of options associated with them are italicized and in bold (***argument***).
- " | " is used as a logical OR.
- T | F = True OR False.

##### Other conventions

- Command examples that are applicable to SplunkWeb are shown in a mock-up of a search bar.

```
foo | top
```

- Command examples that are applicable to the Splunk command line (CLI) are shown in indented fixed-width font.

```
./splunk search "foo | top"
```

---

#### The run command

The run command makes calls to external perl or python programs that can modify or generate search results. It takes search results as inputs, and outputs the results of the script(s) called.

To disable the running of a script, delete the script out of the `splunk_home/etc/searchscripts` directory.

#### Syntax

**run** (**perl** OR **python**) *script-name* [*script-argument*] ... [*script-argumentN*] [*maxinputs-arg*]

#### Arguments

<i>script-name</i> =	script name	The name of the script to execute (minus the path and file extension).
<i>script-argument</i> =	script arguments	An argument passed to the script.
<i>maxinputs-arg</i> =	<b>maxinputs</b> =integer(100)	Specify a number of results to pass to the script. If no <code>maxinputs</code> is specified, run will pass up to 10,000 events to scripts.

**Examples**`404 | run python myscript myarg1 myarg2 | sendemail to= email@site.com`

- Searches for events containing 404, and runs the python script= myscript. Then it sends the results in an email to email@site.com.

#### The admin command

This data-generating command returns the values of a specified ".conf file.

#### Syntax

**admin** *configuration file*

#### Arguments

<i>configuration file</i> =	bundle name	Name of a bundle that corresponds to a Splunk .config file (e.g. eventtypes, inputs, props).
-----------------------------	-------------	--

#### Examples

SplunkWeb:

```
admin eventtypes
```

- Returns the values of the eventtypes.conf file.

CLI:

```
./splunk search "admin auth"
```

- Returns authentication settings in auth.conf.

```
./splunk search "admin props"
```

- Returns processing properties - time zones, breaking characters, etc contained in props.conf.