# Table of Contents

**Welcome to the Troubleshooting Manual.**

- What's in the Troubleshooting Manual? ................................................................. 1

**First steps.**

- Introduction to troubleshooting Splunk Enterprise ................................................. 2
- Determine which version of Splunk Enterprise you're running .......................... 3
- Use btool to troubleshoot configurations ............................................................... 4
- Splunk on Splunk app .............................................................................................. 6

**Splunk Enterprise log files.**

- What Splunk software logs about itself ................................................................. 8
- Enable debug logging ............................................................................................ 11
- About metrics.log ................................................................................................... 14
- Troubleshoot inputs with metrics.log ................................................................. 20
- About access logs .................................................................................................. 22

**Platform instrumentation.**

- About Splunk Enterprise platform instrumentation ............................................. 24
- What does platform instrumentation log? .............................................................. 25
- Configure platform instrumentation ...................................................................... 28
- Sample platform instrumentation searches ......................................................... 32

**Contact Splunk Support.**

- Contact Support .................................................................................................... 34
- How to file a great Support case ............................................................................ 35
- Generate a diag ....................................................................................................... 36
- Anonymize data samples to send to Support ......................................................... 42
- Collect pstacks ....................................................................................................... 46
- Command line tools for use with Support ............................................................ 47

**Splunk Web and search problems.**

- I can't find my data! ............................................................................................... 54
- Too many search jobs............................................................................................ 56
- Dashboard in app is not showing the expected results ........................................ 57
- Intermittent authentication timeouts on search peers .......................................... 58
- Performance degraded in a search head pooling environment .......................... 60
- I'm having problems with the Splunk PDF Server app ....................................... 61
- SuSE Linux search error ...................................................................................... 62

**Data acquisition problems.**

- Identify and triage indexing performance problems ............................................ 63
- Event indexing delay ............................................................................................ 66
- Garbled events ..................................................................................................... 69
- Binary file error ................................................................................................... 70
- What do I do with buckets? ................................................................................ 70
Table of Contents

Windows-specific problems..............................................................................................................................................72
  Troubleshoot Windows event log collection.............................................................................................................72
  Common issues with Splunk and WMI....................................................................................................................72
  Advanced help troubleshooting Splunk software for Windows................................................................................80

System administration problems......................................................................................................................................84
  Troubleshoot high memory usage...........................................................................................................................84
  I get errors about ulimit in splunkd.log..................................................................................................................85
  Splunk Enterprise does not start due to unusable filesystem..................................................................................87
  HTTP thread limit issues..........................................................................................................................................88
Welcome to the Troubleshooting Manual

What's in the Troubleshooting Manual?

Use the Troubleshooting Manual to troubleshoot your instance of Splunk Enterprise.

The following list shows a description of each chapter you see in the left navigation bar.

First steps

Get oriented in Introduction to troubleshooting Splunk Enterprise. Find some tips about where to start with your troubleshooting.

Splunk Enterprise log files

Splunk Enterprise logs all sorts of data about itself. Find out what, where, and how in What Splunk software logs about itself.

Platform instrumentation

Splunk Enterprise generates data about system resource utilization in a special index. Read about this additional feature in About Splunk Enterprise platform instrumentation.

Contact Splunk Support

If you're stuck, contact us! Details and tips in Contact Support.

Some common scenarios

These sections include some of the most common scenarios we see in Splunk Support, with suggestions about what to do.

- "Common front-end scenarios" starts with I can't find my data!
- "Common back-end scenarios" starts with What do I do with buckets?
First steps

Introduction to troubleshooting Splunk Enterprise

This topic is intended as a first step in either diagnosing your Splunk Enterprise problem yourself or asking for help.

Narrow down the problem

For example, if the error occurs in a dashboard or alert, check the underlying search first to see whether the error appears there. When troubleshooting searches, it's almost always best to remove the dashboard layer as soon as possible.

For another example, does the problem exist in one app but not the other? With one user but not admins?

Basically, is there any case for which this does work?

Did the error start occurring after the product was functioning normally?

Yes! So what has changed? Remember to think of both Splunk and non-Splunk factors. Was there a server outage? Network problems? Has any configuration or topology changed?

No, it never functioned normally. Check the operating environment and installation. Start with the system requirements in the Installation Manual.

Resources to help you

Configurations

Splunk has configuration files in several locations, with rules about which files take precedence over each other. Use btool to check which settings your Splunk instance is using. Read about btool in this manual.

The *.conf files are case-sensitive. Check settings and values against the spec and example configuration files in the Admin manual.

There are also a lot of settings in the .conf files that aren't exposed in Splunk Web. It's best to leave these alone unless you know what changing these settings might do.

Splunk log files

Splunk has various internal log files that can help you diagnose problems. Read about the log files in this manual.

Understand how your data gets into Splunk

The Distributed Deployment Manual has a high-level overview of the Splunk data pipeline, breaking it into input, parsing, indexing, and search segments.

For more detail on each segment, see this Community Wiki article about how indexing works.
I've figured out exactly where the problem is

Hey, well done!

Check the (continuously growing) chapter in this manual on some of the most common symptoms and solutions.

If you need additional help or opinions, ask the Splunk community! The Community Wiki, Splunk Answers, and the #splunk IRC channel on efnet are available to everyone and provide a great resource.

Test potential fixes or workarounds

Once you've found a way to fix the problem, test it! Test any noninvasive changes first. Then, test any changes that would create minor interruptions. Make sure no new issues arise from your tested solution.

Always test invasive or major changes in a sandbox environment before moving them to your production system! Your sandbox should be an independent system that mirrors the affected environment.

Stuck?

If you get stuck at any point, contact Splunk Support. Don't forget to send a diag! Read about making a diag in this manual.

Determine which version of Splunk Enterprise you're running

In Splunk Web

Click the About link at the bottom left of most pages in Splunk Web to view a JavaScript overlay with the version and build numbers.

At the command line

Use one minus or two minuses; Splunk gets it either way:

> ./splunk --version
Splunk 6.0 (build 181491)

or

> ./splunk -version
Splunk 6.0 (build 181491)

From the files

You can get the version information from the file $SPLUNK_HOME/etc/splunk.version
In Splunk Search

Splunk Enterprise indexes the splunk.version file into the _internal index and sends it along to the indexer by forwarders.

Here's a search that shows you how many installs you have of each Splunk Enterprise version:

```
index=_internal sourcetype=splunk_version | dedup host | top VERSION
```

Use btool to troubleshoot configurations

The Splunk Enterprise configuration file system supports many overlapping configuration files in many different locations. How these configuration files interact with and take precedence over one another is described in Configuration file precedence in the Admin Manual. This flexibility can make it hard to figure out exactly which configuration value Splunk Enterprise is using.

To help you out, Splunk provides **btool**. This is a command line tool that can help you troubleshoot configuration file issues or see what values are being used by your Splunk Enterprise installation.

Btool displays merged on-disk configurations. That is, btool shows you the merged settings in the .conf files. It does not necessarily show you what Splunk software is currently using. So for example if you edit a .conf file and do not restart (and the edit requires a restart), btool reports the newly edited settings rather than the settings that are currently being used. To view current in-memory configurations, query the REST endpoint `/services/properties/`.

**Note:** btool is not officially supported by Splunk. That said, it is what our Support team uses when trying to troubleshoot your issues.

Investigate configuration values of your entire Splunk software instance

You can run **btool** to see all the configuration values in use by your Splunk software instance.

From `$SPLUNK_HOME/bin` type:

```
./splunk cmd btool <conf_file_prefix> list
```

where `<conf_file_prefix>` is the name of the configuration file you're interested in (minus the .conf extension). The `list` literal specifies that you want to list the options.

For example, to see what settings transforms.conf is using, type:

```
./splunk cmd btool transforms list
```

You probably want to send the results of btool into a text file that you can peruse then delete, like this:

```
./splunk cmd btool transforms list > /tmp/transformsconfigs.txt
```
or if not to a file, at least pipe to grep like this:

```
./splunk cmd btool server list --debug | grep '\['
```

which determines which server.conf stanzas are being recognized.

Redirecting the output to a file is handy for all use cases of btool, but for simplicity we'll only explicitly mention it this once.

### Investigate configuration values in one app

You can also run `btool` for a specific app in your Splunk instance. It lists all the configuration values in use by that app for a given configuration file.

To run `btool`, go to `$SPLUNK_HOME/bin` and type:

```
./splunk cmd btool --app=<app_name> <conf_file_prefix> list
```

where `<app_name>` is the name of the app you want to see the configurations for.

For example, if you want to know what configuration options are being used in `props.conf` by the Search app, type:

```
./splunk cmd btool --app=search props list
```

This returns a list of the `props.conf` settings currently being used for the Search app.

The app name is not required. In fact, it is often a good idea **not** to specify the app when using `btool`. In the case of `btool`, insight into all of your configurations can be helpful.

### Learn where configuration values come from

Another thing you can do with `btool` is find out from which specific app Splunk is pulling its configuration parameters for a given configuration file. To do this, add the `--debug` flag to `btool` like in this example for `props.conf`:

```
./splunk cmd btool props list --debug
```

Read about `btool` syntax in Command line tools for use with Support.

### Check for typos in stanza and setting names

There is a conf checker that runs on "splunk start". You can manually invoke it with the following command:

```
./splunk cmd btool check
```

Any typos are printed to stdout. The `cmd` is optional.

For example, say your local `indexes.conf` has an incorrectly capitalized setting like the following:

```bash
$ cat $SPLUNK_HOME/etc/system/local/indexes.conf
[hello]
#This should be capitalized "homePath".
homepath = $SPLUNK_DB/hello/db
```

The command and its output look like the following:
$ splunk btool check
Possible typo in stanza [hello] in /opt/splunk/etc/system/local/indexes.conf, line 3: homepath = $SPLUNK_DB/hello/db

Additional resources

Watch a video on using btool to troubleshoot configuration issues by a Splunk Support engineer.

Video link: "https://www.youtube.com/watch?v=kuUf4qgL2wl?rel=0"

Have questions? Visit Splunk Answers and see what questions and answers the Splunk community has using btool.

## Splunk on Splunk app

Splunk on Splunk (SoS) is a legacy app that uses Splunk Enterprise diagnostic tools to analyze and troubleshoot your configuration. SoS reached its end of life with version 6.3.0 of Splunk Enterprise. The app is no longer available for download and is not supported in any way.

### How Splunk on Splunk differs from the Monitoring Console

The SoS app reached its end of life with version 6.3.0 of Splunk Enterprise. Its functionality is replaced and extended by the Monitoring Console, which is included with Splunk Enterprise versions 6.2.0 and later.

We recommend that you migrate from SoS to the Monitoring Console for Splunk Enterprise monitoring and introspection.

<table>
<thead>
<tr>
<th>Supported</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acquired Via</td>
<td>Splunkbase</td>
<td>Ships with Splunk Enterprise</td>
</tr>
<tr>
<td>Install Location</td>
<td>Search Head</td>
<td>Non-production search head</td>
</tr>
<tr>
<td>Supports Single Instance</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Data Sources</td>
<td>Splunk Logs, Scripted Inputs (counts against license)</td>
<td>Splunk Logs, Introspection (does not count against license), REST</td>
</tr>
<tr>
<td>User Defined Grouping</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Topology View</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Topology - Server Roles</td>
<td>Search Heads, Indexers, Forwards</td>
<td>Search Heads, Indexers, Custom Groups</td>
</tr>
<tr>
<td>Topology - Node Detail</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Topology - Overlay</td>
<td>Status, CPU, Memory</td>
<td>Status, CPU, Memory, Search Count, Indexing Rate</td>
</tr>
<tr>
<td>Topology - Node Relationship</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Configuration File Viewer</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Security Health Check</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Warnings &amp; Errors/Crashlog View</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Resource Usage Views</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

6
<table>
<thead>
<tr>
<th>Resource Usage - CPU/Memory by Splunk Instance</th>
<th>SoS</th>
<th>Monitoring Console</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Usage - CPU/Memory Deployment Views</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Resource Usage - File Descriptor Usage</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>KV Store</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Forwarder Monitoring</td>
<td>No</td>
<td>Yes (6.3.0+)</td>
</tr>
<tr>
<td>HTTP Event Collector</td>
<td>No</td>
<td>Yes (6.4.0+)</td>
</tr>
</tbody>
</table>
Splunk Enterprise log files

What Splunk software logs about itself

Splunk software keeps track of its activity by logging to various files in $SPLUNK_HOME/var/log/splunk.

The Splunk platform internal log files are rolled based on size. You can change the default log rotation size by editing $SPLUNK_HOME/etc/log.cfg.

Search these files in Splunk Web by typing:

index=_internal

Search for errors and warnings by typing:

index=_internal (log_level=error OR log_level=warn*)

Internal logs

Here is a list, with descriptions, of the internal logs in $SPLUNK_HOME/var/log/splunk. The internal logs from Splunk software are useful for troubleshooting or metric analysis.

Note that some log files are not created until your Splunk platform instance uses them.

<table>
<thead>
<tr>
<th>Log file name</th>
<th>Useful for?</th>
</tr>
</thead>
<tbody>
<tr>
<td>audit.log</td>
<td>Information about user activity, most interestingly about a user logging in (or failing to log in), modifying a setting, updating a lookup, or running a search. For example, if you're looking for information about a saved search, audit.log matches the name of a saved search (savedsearch_name) with its search ID (search_id), user, and time. With the search_id, you can look up that particular search elsewhere, like in the search dispatch directory. Read about audit events in the Securing Splunk Manual. Audit.log is the only file indexed to _audit.</td>
</tr>
<tr>
<td>btool.log</td>
<td>Log of btool activity. Read about btool in this manual.</td>
</tr>
<tr>
<td>conf.log</td>
<td>Contains messages about configuration replication related to search head clustering.</td>
</tr>
<tr>
<td>django_access.log</td>
<td>Django HTTP request log (equivalent to web_access.log) for the Django Bindings component of the Splunk Web Framework.</td>
</tr>
<tr>
<td>django_error.log</td>
<td>Raw Django error output from Splunk Web Framework (not really meant to be human readable). Used with link on error screens to see the full error in Splunk Web.</td>
</tr>
<tr>
<td>django_service.log</td>
<td>General Django related messages from Splunk Web Framework (equivalent to web_service.log)</td>
</tr>
<tr>
<td>export_metrics.log</td>
<td>Log of metrics related to exporting data with Hadoop Connect.</td>
</tr>
<tr>
<td>first_install.log</td>
<td>Shows version number.</td>
</tr>
<tr>
<td>intentions.log</td>
<td>Beginning with Splunk 5, no longer used. Read about intentions in Developing Views and Apps for Splunk Web.</td>
</tr>
<tr>
<td>license_audit.log</td>
<td>No longer used.</td>
</tr>
<tr>
<td>license_usage.log</td>
<td>Indexed volume in bytes per pool, index, source, source type, and host. Available only on a Splunk license master.</td>
</tr>
<tr>
<td>Log file name</td>
<td>Useful for?</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>metrics.log</td>
<td>Contains periodic snapshots of Splunk performance and system data, including information about CPU usage by internal processors and queue usage in Splunk's data processing. The metrics.log file is a sampling of the top ten items in each category in 30 second intervals, based on the size of _raw. It can be used for limited analysis of volume trends for data inputs. For more information about metrics.log, see About metrics.log and Work with metrics.log in this manual.</td>
</tr>
<tr>
<td>migration.log</td>
<td>A log of events during install and migration. Specifies which files were altered during upgrade.</td>
</tr>
<tr>
<td>mongodb.log</td>
<td>Contains runtime messages from the Splunk Enterprise app key value store.</td>
</tr>
<tr>
<td>python.log</td>
<td>Python events within Splunk. Useful for debugging REST endpoints, communication with splunkd, PDF Report Server App, Splunk Web display issues, sendmail (email alerts), and scripted inputs. With web_service.log, one of the few Splunk logs that uses &quot;WARNING&quot; instead of &quot;WARN&quot; for second most verbose logging level.</td>
</tr>
<tr>
<td>remote_searches.log</td>
<td>Messages from StreamedSearch channel. This code is executed on the search peers when a search head makes a search request. So this file contains useful information on indexers regarding searches they're participating in.</td>
</tr>
<tr>
<td>scheduler.log</td>
<td>All actions (successful or unsuccessful) performed by the splunkd search and alert scheduler. Typically, this shows scheduled search activity.</td>
</tr>
<tr>
<td>searches.log</td>
<td>Beginning with Splunk 5, no longer used. Instead, use the following search syntax:</td>
</tr>
<tr>
<td>searchhistory.log</td>
<td>No longer used.</td>
</tr>
<tr>
<td>splunkd.log</td>
<td>The primary log written to by the Splunk server. May be requested by Splunk Support for troubleshooting purposes. Any stderr messages generated by scripted inputs, scripted search commands, and so on, are logged here.</td>
</tr>
<tr>
<td>splunkd_access.log</td>
<td>Any action done from splunkd through the UI is logged here, including splunkweb, the CLI, all POST GET actions, deleted saved searches, and other programs accessing the REST endpoints. Also logs the time taken to respond to the requests. Search job artifacts logged here include size of data returned with search. sourcetype=&quot;splunkd_access&quot;</td>
</tr>
<tr>
<td>splunkd_stderr.log</td>
<td>The Unix standard error device for the server. Typically this contains (for *nix) times of healthy start and stop events, as well as various errors like exceptions, assertions, and errors generated by libraries and the operating system.</td>
</tr>
<tr>
<td>splunkd_stdout.log</td>
<td>The Unix standard output device for the server.</td>
</tr>
<tr>
<td>splunkd_ui_access.log</td>
<td>Starting in 6.2, contains a significant portion of the types of events that used to be logged in web_access.log.</td>
</tr>
<tr>
<td>splunkd-utility.log</td>
<td>This log is written to by the prereq-checking utility splunkd clone-prep-clear-config, splunkd validatedb, splunkd check-license, splunkd check-transforms-keys, and splunkd rest (for offline CLI). Each util logs Splunk version, some basic config, and current OS limits like max number of threads, and then messages specific to the util. Consult this log file when splunkd didn't start.</td>
</tr>
<tr>
<td>web_access.log</td>
<td>Requests made of Splunk Web, in an Apache access_log format. Much of the types of events logged here are logged in splunkd-ui_access.log starting in 6.2.</td>
</tr>
<tr>
<td>web_service.log</td>
<td>Primary log written by splunkweb. Records actions made by splunkweb. This and python.log are the only logs that, in second most verbose logging level, write messages with &quot;WARNING&quot; instead of Splunk log files' usual &quot;WARN.&quot;</td>
</tr>
</tbody>
</table>

**Introspection logs**

Splunk Enterprise platform instrumentation refers to data that your Splunk Enterprise deployment logs in the _introspection index. It gathers data about your Splunk instance and operating system and writes it to log files that you can search later to aid in troubleshooting a variety of problems. You can also view the data at REST endpoints.

Read more in “About Splunk Enterprise platform instrumentation” in this manual.
**Splunk platform search logs**

The Splunk platform also creates search logs. These are not indexed to _internal.

Each search has its own directory for all information specific to the search, including a search log. The directory for each search is named with (among other parameters) the search_id. This search_id is matched to a human-readable search name in audit.log. The search directory for a recent search is in `$SPLUNK_HOME/var/run/splunk/dispatch/`. In addition to the search.log file, a search directory contains other information about the search job, including the results. A search directory for a completed search is referred to as a **search artifact**.

If you have any long-running real-time searches, you might want to adjust the maximum size of your search logs. These logs are rotated when they reach a default maximum size of 10 MB. Splunk software keeps up to five of them for each search, so the total log size for a search can conceivably grow as large as 30 MB.

Most searches are unlikely to generate logs anywhere near 10 MB in size. But size can become an issue if you have ongoing real-time searches.

To adjust the log size, edit `$SPLUNK_HOME/etc/log-searchprocess.cfg`.

See Dispatch directory and search artifacts in the **Search Manual**.

**Debug mode**

The Splunk platform has a debugging parameter. Read about enabling debug logging in this manual.

Except where noted above, Splunk platform internal logging levels are **DEBUG INFO WARN ERROR FATAL** (from most to least verbose).

**Note**: Running Splunk software with debugging turned on outputs a large amount of information. Make sure you do not leave debugging on for any significant length of time.

**Use Splunk Web to manage logs**

To view and manage logs, you can use Splunk Web:

1. Navigate to **Settings > Server settings > Server logging**. This generates a list of log channels and their status.

2. To change the logging level for a particular log channel, click on that channel. This brings up a page specific to that channel.

3. On the log channel’s page, you can change its logging level.

When you change the logging level, note the following:

- The change is immediate and dynamic.
- The change is not persistent; it goes away when the Splunk platform is restarted.

**Settings > Server settings > Server logging** is meant only for dynamic and temporary changes to Splunk log files. For permanent changes, use `$SPLUNK_HOME/etc/log.cfg` instead.
Included data models

Splunk Enterprise comes with several sample data models. These data models are constructed from the internal logs from Splunk Enterprise. By interacting with them, you can learn about Splunk software log files and about data models in one fell swoop.

To access the internal log data models, in the Search & Reporting app in Splunk Web, click **Datasets**.

Enable debug logging

The Splunk platform internal logging levels are **DEBUG INFO WARN ERROR FATAL** (from most to least verbose). This topic gives a few popular options for how you might want to put Splunk into debug mode.

Be warned, Splunk software debug mode is **extremely verbose**. All the extra chatter might obscure something that might have helped you diagnose your problem. Running Splunk software in debug mode for an extended time makes your internal log files unwieldy. Running debug mode is not recommended on production systems.

Enable debug logging on all of splunkd.log

Splunk software has a debugging parameter (**--debug**) that you can use when starting Splunk software from the CLI in *nix. This command outputs logs to \$SPLUNK_HOME/var/log/splunk/splunkd.log. To enable debug logging from the command line:

- Navigate to \$SPLUNK_HOME/bin.
- Stop Splunk, if it is running.
- Save your existing splunkd.log file by moving it to a new filename, like splunkd.log.old.
- Restart Splunk in debug mode with splunk start --debug.
- When you notice the problem, stop Splunk.
- Move the new splunkd.log file elsewhere and restore your old one.
- Stop or restart Splunk normally (without the --debug flag) to disable debug logging.

Specific areas can be enabled to collect debugging details over a longer period with minimal performance impact. See the category settings in the file \$SPLUNK_HOME/etc/log.cfg to set specific log levels without enabling a large number of categories as with --debug. Restart Splunk after changing this file.

**Important:** Changes to \$SPLUNK_HOME/etc/log.cfg are overwritten if you upgrade your version of Splunk software.

**Note:** Not all messages marked WARN or ERROR indicate actual problems with Splunk software; some indicate that a feature is not being used.

Note also that this option is not available on Windows. To enable debugging on Splunk software running on Windows, enable debugging on a specific processor in Splunk Web or using log.cfg.

Enable debug logging for a specific processor within splunkd.log

**In Splunk Web**

You can enable these DEBUG settings via Splunk Web if you have admin privileges. Navigate to **Settings > Server settings > Server logging**. Search for the processor names using the text box. Click on the processor name to change the logging level to DEBUG. You do not need to restart Splunk. In fact, these changes will not persist if you restart the
Splunk instance.

In log.cfg

If you want the processors to be in DEBUG on startup, or if you want to turn on debugging for a few processors or for a lightweight forwarder (with no Splunk Web), create or edit a $SPLUNK_HOME/etc/log-local.cfg file to override changes in $SPLUNK_HOME/etc/log.cfg.

In $SPLUNK_HOME/etc/log.cfg, find the category.* entry that relates to the processor you are interested in, and copy the line to log-local.cfg with INFO or WARN modified to DEBUG. There will not always be an existing entry for the processor you are interested in, and it may take some digging through the logs or documentation to find the correct one.

For example, to see how often Splunk software is updating or retrieving progress-tracking records fora particular file, put ‘FileInputTracker’ in DEBUG. Update the existing entry to read category.FileInputTracker=DEBUG

Or for investigating problems monitoring files, use the FileInputTracker and selectProcessor categories.

Restart the Splunk platform. Now every time Splunk software checks the inputs file, it will be recorded in $SPLUNK_HOME/var/log/splunk/splunkd.log. Remember to change these settings back when you are finished investigating.

If a default level is not specified for a category, the logging level defaults to your rootCategory setting.

Note: Leave category.loader at INFO. This is what gives us our build and system info.

To change the maximum size of a log file before it rolls, change the maxFileSize value (in bytes) for the desired file:

appender.A1=RollingFileAppender
appender.A1.fileName=${SPLUNK_HOME}/var/log/splunk/splunkd.log
appender.A1.maxFileSize=250000000
appender.A1.maxBackupIndex=5
appender.A1.layout=PatternLayout
appender.A1.layout.ConversionPattern=%d{%m-%d-%Y %H:%M:%S.%l} %-5p %c - %m%n

About precedence

If you have duplicate lines in log.cfg, the last line takes precedence. For example,

category.databasePartitionPolicy=INFO
category.databasePartitionPolicy=DEBUG

will give you DEBUG, but in the other order it will not.

The other log-* .cfg files behave similarly when you add categories. To set only some things in a search.log into debug mode, in log-searchprocess.cfg add a new category line after the rootCategory:

rootCategory=INFO, searchprocessAppender
category.<foo>=DEBUG
appender.searchprocessAppender=RollingFileAppender
This leaves everything else as it was, which means only the debug messages you want are generated. Putting rootCategory into DEBUG mode makes the dispatch directories huge, so it is not a good choice for long-running debug.

**log-local.cfg**

You can put log.cfg settings into a local file, log-local.cfg file, residing in the same directory as log.cfg. The settings in log-local.cfg take precedence. And unlike log.cfg, the log-local.cfg file doesn’t get overwritten on upgrade.

**With endpoints**

You can access a debugging endpoint that shows status information about monitored files:

https://your-splunk-server:8089/services/admin/inputstatus/TailingProcessor:FileStatus

**Enable debug messages from the CLI**

```
./splunk _internal call /services/server/logger/TailingProcessor -post:level DEBUG
```

*Note:* This search returns the message "HTTP Status: 200". This is not an error and is normal.

You can also enable debugging with this command:

```
./splunk set log-level TailingProcessor -level DEBUG
```

**Enable debug logging for search processes**

Search processes obey the etc/log-searchprocess.cfg rules. Similar to splunkd, they can be overridden in etc/log-searchprocess-local.cfg.

All loggers can be set to DEBUG by adding a line such as

```
rootCategory=DEBUG,searchprocessAppender
```

Specific loggers can be set to debug as well, for example:

```
category.UnifiedSearch=DEBUG
category.IndexScopedSearch=DEBUG
```

This change takes effect immediately for all searches started after the change.

**Debug Splunk Web**

Change the logging level for the splunkweb process by editing the file:

```
$SPLUNK_HOME/etc/log.cfg
```
or if you have created your own $SPLUNK_HOME/etc/log-local.cfg

Locate the [python] stanza and change the contents to:
[python]
splunk = DEBUG
# other lines should be removed

The logging component names are hierarchical so setting the top level splunk component will affect all loggers unless a more specific setting is provided, like splunk.search = INFO.

Restart the splunkweb process with the command .splunk restart splunkweb. The additional messages are output in the file $SPLUNK_HOME/var/log/splunk/web_service.log.

About metrics.log

This topic is an overview of metrics.log.

- To learn about other log files, read “What Splunk logs about itself.”
- For an example using metrics.log, read “Troubleshoot inputs with metrics.log.”

What information is in metrics.log?

Metrics.log has a variety of introspection information for reviewing product behavior.

First, metrics.log is a periodic report, taken every 30 seconds or so, of recent Splunk software activity.

By default, metrics.log reports the top 10 results for each type. You can change that number of series from the default by editing the value of maxseries in the [metrics] stanza in limits.conf.

Structure of the lines

Here is a sample line from metrics.log:

01-27-2010 15:43:54.913 INFO Metrics - group=pipeline, name=parsing, processor=utf8, cpu_seconds=0.000000, executes=66, cumulative_hits=301958

First, boiler plate: the timestamp, the "severity," which is always INFO for metrics events, and then the kind of event, "Metrics."

The next field is the group. This indicates what kind of metrics data it is. There are a few groups in the file, including:

- pipeline
- queue
- thruput
- tcpout_connections
- udpin_connections
- mpool
- per_host_regex_cpu
- per_index_regex_cpu
- per_source_regex_cpu
- per_sourcetype_regex_cpu
**Pipeline messages**

Pipeline messages are reports on the Splunk pipelines, which are the strung-together pieces of "machinery" that process and manipulate events flowing into and out of the Splunk system. You can see how many times data reached a given machine in the Splunk system (executes), and you can see how much cpu time each machine used (cpu_seconds).

Plotting totals of cpu seconds by processor can show you where the cpu time is going in indexing activity. Looking at numbers for executes can give you an idea of data flow. For example if you see:

```
group=pipeline, name=merging, processor=aggregator, ..., executes=998
```
```
group=pipeline, name=merging, processor=readerin, ..., executes=698
```
```
group=pipeline, name=merging, processor=regexreplacement, ..., executes=698
```
```
group=pipeline, name=merging, processor=sendout, ..., executes=698
```

then it's pretty clear that a large portion of your items aren't making it past the aggregator. This might indicate that many of your events are multi-line and are being combined in the aggregator before being passed along.

Read more about Splunk's data pipeline in "How data moves through Splunk" in the Distributed Deployment Manual.

**Queue messages**

Queue messages look like

```
... group=queue, name=parsingqueue, max_size=1000, filled_count=0, empty_count=8, current_size=0, largest_size=2, smallest_size=0
```

Most of these values are not interesting. But current_size, especially considered in aggregate, across events, can tell you which portions of Splunk indexing are the bottlenecks. If current_size remains near zero, then probably the indexing system is not being taxed in any way. If the queues remain near 1000, then more data is being fed into the system (at the time) than it can process in total.

Sometimes you will see messages such as  
```
... group=queue, name=parsingqueue, blocked!!=true, max_size=1000, filled_count=0, empty_count=8, current_size=0, largest_size=2, smallest_size=0
```

This message contains the blocked string, indicating that it was full, and someone tried to add more, and couldn't. A queue becomes unblocked as soon as the code pulling items out of it pulls an item. Many blocked queue messages in a sequence indicate that data is not flowing at all for some reason. A few scattered blocked messages indicate that flow control is operating, and is normal for a busy indexer.

If you want to look at the queue data in aggregate, graphing the average of current_size is probably a good starting point.

There are queues in place for data going into the parsing pipeline, and for data between parsing and indexing. Each networking output also has its own queue, which can be useful to determine whether the data is able to be sent promptly, or alternatively whether there's some network or receiving system limitation.

Generally, filled_count and empty_count cannot be productively used for inferences.

**Thruput messages**

Thruput messages (similar to the English word "throughput") come in a few varieties.
Thruput is measured in the indexing pipeline. If your data is not reaching this pipeline for some reason, it will not appear in this data. Thruput numbers relate to the size of "raw" of the items flowing through the system, which is typically the chunks of the original text from the log sources. This differs from the tcpout measurements, which measure the total byte count written to sockets, including protocol overhead as well as descriptive information like host, source, source type.

First there is a catchall line, which looks like this:

```
... group=thruput, name=index_thruput, instantaneous_kbps=0.287598, instantaneous_eps=1.000000, average_kbps=0.270838, total_k_processed=74197, load_average=1.345703M
```

This is the best line to look at when tuning performance or evaluating indexing load. It tries to capture the total indexing data load.

**Note:** In thruput lingo, kbps does not mean kilobits per second, it means kilobytes per second. The industry standard term would be to write this something like KBps.

The `average_kbps` in the `group=thruput` catchall indicates the average since Splunk Enterprise started. `instantaneous_kbps` indicates the average kbps for the reporting interval (equivalent to kbps for the breakouts below.)

The most useful figure to look at in aggregate is probably `instantaneous_kbps` over time.

Following the catchall, there can be variety of breakouts of the indexing thruput, including lines like:

```
... group=per_host_thruput, series="jombook.splunk.com", kbps=0.261530, eps=1.774194, kb=8.107422, ev=2606, avg_age=420232.710668, max_age=420241

... group=per_index_thruput, series="_internal", kbps=0.261530, eps=1.774194, kb=8.107422, ev=2606, avg_age=420232.710668, max_age=420241

... group=per_source_thruput, series="/applications/splunk4/var/log/splunk/metrics.log", kbps=0.261530, eps=1.774194, kb=8.107422, ev=2606, avg_age=420232.710668, max_age=420241

... group=per_sourcetype_thruput, series="splunkd", kbps=0.261530, eps=1.774194, kb=8.107422, ev=2606, avg_age=420232.710668, max_age=420241
```

In thruput messages the data load is broken out by host, index, source, and source type. This can be useful for answering two questions:

- Which data categories are busy?
- When were my data categories busy?

The series value identifies the host or index, etc. The kb value indicates the number of kilobytes processed since the last sample. Graphing kb in aggregate can be informative. The summary indexing status dashboard uses this data, for example.

- `ev` is a simple total count of events during the sampling period.
- `kbps`, as before, indicates kilobytes per second averaged over the sampling period.

The `avg_age` and `max_age` refer to the difference between the time that the event was seen by the thruput processor in the indexing queue, and the time when the event occurred (or more accurately, the time that Splunk decided the event occurred).

- `max_age` is the largest difference between the current time and the perceived time of the events coming through the thruput processor.
• **avg_age** is the average difference between the current time and the perceived time of the events coming through the thruput processor.

**Note:** The per_x_thruput categories are not complete. Remember that by default metrics.log shows the 10 busiest of each type, for each sampling window. If you have 2000 active forwarders, you cannot expect to see the majority of them in this data. You can adjust the sampling quantity, but this will increase the chattiness of metrics.log and the resulting indexing load and _internal index size. The sampling quantity is adjustable in limits.conf, [metrics] maxseries = num.

Ignore the **eps** value, as it has accuracy issues.

**Tcpout Connections messages**

Tcpout connections messages provide a variety of information about the systems that the tcpout component is currently connected to, via an open socket. One line is produced per connected system.

These lines look like this:

```
... group=tcpout_connections, name=undiag_indexers:10.159.4.67:9997:0, sourcePort=11089, destIp=10.159.4.67, destPort=9997, _tcp_Bps=28339066.17, _tcp_KBps=27674.87, _tcp_avg_thruput=27674.87, _tcp_Kprocessed=802571, _tcp_eps=33161.10, kb=802571.21
```

• **name** is a combination of conf stanza and entries that fully define the target system to the software.
• **sourcePort** is the dynamically assigned port by the operating system for this socket.
• **destIP** and **destPort** define the destination of the socket. **destPort** is typically statically configured, while **destIP** may be configured or returned by name resolution from the network.

All of the size-related fields are based on the number of bytes that Splunk has successfully written to the socket, or SSL-provided socket proxy. When SSL is not enabled for forwarding (the default), these numbers represent the number of bytes written to the socket, so effectively the number of bytes conveyed by the tcp transport (irrespective of overhead issues such as keepalive, tcp headers, ip headers, ethernet frames and so on). When SSL is enabled for forwarding, this number represents the number of bytes Splunk handed off to the OpenSSL layer.

• **_tcp_Bps** is the bytes transmitted during the metrics interval divided by the duration of the interval (in seconds)
• **_tcp_KBps** is the same value divided by 1024
• **_tcp_avg_thruput** is an average rate of bytes sent since the last time the tcp output processor was reinitialized/reconfigured. Typically this means an average since Splunk started.
• **_tcp_KProcessed** is the total number of bytes written since the processor was reinitialized/reconfigured, divided by 1024.
• **_tcp_eps** is the number of items transmitted during the interval divided by the direction of the interval (in seconds). Note that items will frequently not be events for universal/light forwarders (instead, data chunks)
• **kb** is the bytes transmitted during the metrics interval divided by 1024.

**udpin messages**

udpin_connections lines are essentially metering on udp input.

```
group=udpin_connections, 2514, sourcePort=2514, _udp_bps=0.00, _udp_kb=0.00, _udp_avg_thruput=0.00, _udp_kprocessed=0.00, _udp_eps=0.00
```

Some should be relatively self-explanatory.
• bps: bytes per second
• kbps: kilobytes per second
• eps: events (packets) per second
• _udp_kprocessed is a running total of kilobytes processed since udp input processor start (typically since splunk
start, but if reconfigured may reset).

Don't have info on avg_thruput at this time.

Be aware that it's quite achievable to max out the ability of the operating system, let alone Splunk, to handle UDP packets
at high rates. This data might be useful to determine if any data is coming in at all, and at what times it rises. There is no
guarantee that all packets sent to this port will be received and thus metered.

**mpool messages**

The mpool lines represent memory used by the Splunk indexer code only (not any other pipeline components). This
information is probably not useful to anyone other than Splunk developers.

```
group=mpool, max_used_interval=4557, max_used=53878, avg_rsv=180, capacity=268435456, used=0
```

• **max_used_interval** represents the number of bytes used during the reporting interval (since the last output).
• **max_used** represents the maximum amount of memory, in bytes, in use at any time during the component's lifetime
  (most likely since last starting Splunk).
• **avg_rsv** is the average size of a memory allocation across the run time of the system.
• **capacity** is the limit on memory use for the indexer.
• **used** is the current indexer's current memory use.

In this case we can see that some memory is sometimes in use, although at the time of the sample, none is in use, and
that generally the use is low.

**map, pipelineinputchannel name messages**

These messages are primarily debugging information over the Splunk internal cache of processing state and configuration
data for a given data stream (host, source, or source type).

```
group=map, name=pipelineinputchannel, current_size=29, inactive_channels=4,
new_channels=0, removed_channels=0, reclaimed_channels=0, timedout_channels=0,
abandoned_channels=0
```

• **current_size** is the number of total channels loaded in the system at the end of the sampling period.
• **inactive_channels** is the number of channels that have no entries in any pipeline referring to them (typically for
  recently seen data but not for data currently being processed) at the end of the sampling period.
• **new_channels** is the number of channels created during the sampling period, meaning that new data streams
  arrived, or a data stream that was aged out was created again.
• **removed_channels** is the number of channels destructed during the sampling period, which means that enough
  pressure existed to push these channels out of set (there were too many other new data streams).
• **reclaimed_channels** is the number of channels that were repurposed during the sampling period. This will happen
  for reasons similar to **new_channels**, based on the size of the utilization, etc.
• **timedout_channels** is the number of channels that became unused for a long enough time to be considered stale
  and the information to be culled. Typically a timedout file or data source hasn't been producing data for some
  time.
• **abandoned_channels** is the number of channels that were terminated by Splunk forwarding, where the forwarder
  stopped communicating to an indexer, so the indexer shut them down.
**subtask_seconds messages**

Currently there is only one type of these messages, name=indexer, task=indexer_service. The message describes how the indexer has spent time over each interval window.

```
subtask_seconds
    group=subtask_seconds, name=indexer, task=indexer_service, replicate_semislice=0.000000,
    throttle_optimize=0.00015, flushBlockSig=0.000000, retryMove_1hotBkt=0.000000, size_hotBkt=0.000000,
    roll_hotBkt=0.000000, chillOrFreeze=0.000000, update_checksums=0.000000, fork_recovermetadata=0.000000,
    rebuild_metadata=0.000300, update_bktManifest=0.000000, service_volumes=0.000105, service_maxSizes=0.000000,
    service_externProc=0.000645
```

- The `throttle_optimize` subtask represents time that the indexer spends waiting for `splunk_optimize` processes to reduce the count of .tsidx files to a reasonable level within hot buckets. Because `splunk_optimize` can in some cases run more slowly merging .tsidx files than the indexer runs while generating them, this flow-control state must exist for `splunk_optimize` to catch up. When this throttling occurs, messages are logged to `splunkd.log` in category `DatabasePartitionPolicy` similar to `idx=<idxname> Throttling indexer, too many tsidx files in bucket`.

- The `rebuild_metadata` subtask represents time that the indexer spends generating new copies of .data files in hot buckets. This can mean writing out `Hosts.data`, `Sourcetypes.data`, `Sources.data`, or `Strings.data`. Typically the largest cost is for `Strings.data` in configurations where a large amount of data is directed to this file. Setting the `MetaData` log channel to DEBUG can provide more information on which buckets and files might be involved in a slowdown.

**regular expression (regex) messages**

These messages represent the amount of CPU time spent processing regular expressions on events, in seconds. The group field will identify if these metrics are based on host, index, source, or sourcetype. The CPU time metrics for the regex processor are not enabled by default. See the configuration `regex_cpu_profiling` in limits.conf in the *Admin Manual*.

Sample Event:

```
09-06-2018 20:00:54.667 +0000 INFO Metrics - group=per_sourcetype_regex_cpu, series="audittrail", cpu=428,
    cpupe=0.9749430523917996, bytes=138929, ev=439
```

Valid group names are:

- `per_host_regex_cpu`
- `per_index_regex_cpu`
- `per_source_regex_cpu`
- `per_sourcetype_regex_cpu`

For each interval, the following fields are produced.

- `cpu` represents the CPU time for a given series.
- `cpupe` represents the CPU time per event for a given series.
- `bytes` represents the number of bytes processes for a given series.
- `ev` represents total events for a given series.

The default interval for each measurement is 30 seconds.
Troubleshoot inputs with metrics.log

This topic is an example of a problem you can solve using metrics.log.

- To learn about metrics.log, read "About metrics.log."
- To learn about other log files, read "What Splunk logs about itself."

Example: Troubleshoot data inputs

You might want to identify a data input that has suddenly begun to generate uncharacteristically large numbers of events. If this input is hidden in a large quantity of similar data, it can be difficult to determine which one is actually the problem. You can find it by searching the internal index (add `index=_internal` to your search) or just look in `metrics.log` itself in `$SPLUNK_HOME/var/log/splunk`.

There's a lot more in metrics.log than just volume data, but for now let's focus on investigating data inputs.

For incoming events, the amount of data processed is in the `thruput` group, as in `per_host_thruput`. In this example, you're only indexing data from one host, so `per_host_thruput` actually can tell us something useful: that right now host "grumpy" indexes around 8k in a 30-second period. Since there is only one host, you can add it all up and get a good picture of what you're indexing, but if you had more than 10 hosts you would only get a sample.

Example metrics output:

```
03-13-2008 10:49:57.634 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.245401, eps=1.774194, kb=7.607422
03-13-2008 10:50:28.642 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.237053, eps=1.612903, kb=7.348633
03-13-2008 10:50:59.648 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.217584, eps=1.548387, kb=6.745117
03-13-2008 10:51:30.656 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.245621, eps=1.741935, kb=7.614258
03-13-2008 10:52:01.661 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.296938, eps=2.322581, kb=9.642578
03-13-2008 10:52:32.669 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.296938, eps=2.322581, kb=9.642578
03-13-2008 10:53:03.677 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.261593, eps=1.838710, kb=8.109375
03-13-2008 10:53:34.686 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.263136, eps=2.032258, kb=8.157227
03-13-2008 10:54:05.692 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.261530, eps=1.806452, kb=8.107422
03-13-2008 10:54:36.699 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.313855, eps=2.354839, kb=9.729492
```

For example, you might know that `access_common` is a popular source type for events on this Web server, so it would give you a good idea of what was happening:

```
03-13-2008 10:51:30.656 INFO Metrics - group=per_sourcetype_thruput, series="access_common", kbps=0.022587, eps=0.193548, kb=0.700195
03-13-2008 10:52:01.661 INFO Metrics - group=per_sourcetype_thruput, series="access_common", kbps=0.053585, eps=0.451613, kb=1.661133
03-13-2008 10:52:32.670 INFO Metrics - group=per_sourcetype_thruput, series="access_common", kbps=0.031786, eps=0.419355, kb=0.985352
03-13-2008 10:53:34.686 INFO Metrics - group=per_sourcetype_thruput, series="access_common", kbps=0.030998, eps=0.387097, kb=0.960938
03-13-2008 10:54:36.700 INFO Metrics - group=per_sourcetype_thruput, series="access_common", kbps=0.070092,
```
But you probably have more than 10 source types, so at any particular time some other one could spike and access_common wouldn't be reported. per_index_thruput and per_source_thruput work similarly.

With this in mind, let's examine the standard saved search "KB indexed per hour last 24 hours". 

```
index=_internal metrics group=per_index_thruput NOT debug NOT sourcetype=splunk_web_access | timechart fixedrange=t span=1h sum(kb) | rename sum(kb) as totalKB
```

This means: look in the internal index for metrics data of group per_index_thruput, ignore some internal stuff and make a report showing the sum of the kb values. For cleverness, we'll also rename the output to something meaningful, "totalKB". The result looks like this:

```
sum of kb vs. time for results in the past day
  _time totalKB
  1 03/12/2008 11:00:00 922.466802
  2 03/12/2008 12:00:00 1144.674811
  3 03/12/2008 13:00:00 1074.541995
  4 03/12/2008 14:00:00 2695.178730
  5 03/12/2008 15:00:00 1032.747082
  6 03/12/2008 16:00:00 898.662123
```

Those totalKB values just come from the sum of kb over a one hour interval. If you like, you can change the search and get just the ones from grumpy:

```
index=_internal metrics grumpy group=per_host_thruput | timechart fixedrange=t span=1h sum(kb) | rename sum(kb) as totalKB
```

```
sum of kb vs. time for results in the past day
  _time totalKB
  1 03/12/2008 11:00:00 746.46681
  2 03/12/2008 12:00:00 988.568358
  3 03/12/2008 13:00:00 936.092772
  4 03/12/2008 14:00:00 2529.226566
  5 03/12/2008 15:00:00 914.945313
  6 03/12/2008 16:00:00 825.353518
```

We see that grumpy was unusually active in the 2 pm time bin. With this knowledge, we can start to hunt down the culprit by, for example, source type or host.

**Answers**

Have questions? Visit Splunk Answers and see what questions and answers the Splunk community has about working with metrics.log.
About access logs

Splunkd and splunkweb both produce access logs in a format similar to common Apache webserver access log formats.

Splunkd produces `splunkd_access.log`, and splunkweb records logs in `web_access.log`. Both log files are close approximations of the Apache combined log format.

Apache formats are described briefly in the Apache HTTP Server documentation. For example, see Apache 2.4 log file documentation.

**splunkd_access.log**

This file records HTTP requests served by splunkd on its management port. Here is a typical line in `splunkd_access.log`:

```
```

These fields are:

```
<address> - <user> [<time>] "<request>" <status> <response_size> - - - <duration>
```

- **address**: The IP address from which the HTTP client socket appears to originate. Typically these requests originate from splunkweb and come over the localhost/loopback address.
- The second field is a placeholder for the unused `identd` field.
- **user**: The splunk user, if any, making the request. System accesses on behalf of no particular user appear as "-".
- **timestamp**: This is the time that splunkd finished reading in the request. However, the log event is written out when the http server finishes writing the response, so these timestamps can be out of order.
- **request**: The HTTP request made by the client consisting of an action, a URL, and a protocol version.
- **status**: The HTTP status returned as part of the response.
- **response_size**: The size of the body of the response in bytes
- Three additional placeholders.
- **duration**: The time it took from the completion of reading the request to completely writing out the response. This value is logged explicitly in milliseconds.

Between the definitions for timestamp and duration, you can infer the response completion time by adding duration to the timestamp.

**web_access.log**

A web access line is similar:

```
"http://mcp.sv.splunk.com:62100/en-US/manager/search/saved/searches" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:32.0) Gecko/20100101 Firefox/32.0" - 5446ca810b7fb1d855110 11ms
```

Here the format is:

```
<address> - <user> [<time>] "<request>" <status> <response_size> "<referer>" "<user agent>" - <session_id> <duration>
```

where address, user, time, request, status, response_size, and duration are the same as in `splunkd_access.log`. The new components here are:

- **referer**: referer [sic] is the URL that the client told us provided the link to the URL that was accessed.
• **user agent**: The string the http client used to identify itself.
• **session_id**: This represents the splunkweb session. Can be used to follow a stream of requests from a particular client. These sessions are transient starting in Splunk Enterprise 6.2.0.

**splunkd_ui_access.log**

Starting in Splunk Enterprise 6.2.0, splunkd handles requests from the browser that splunkweb handled pre-6.2.0. This file records HTTP requests served by splunkd on the Splunk Web port. The format is identical to `web_access.log`. 
Platform instrumentation

About Splunk Enterprise platform instrumentation

Splunk Enterprise platform instrumentation refers to data that Splunk Enterprise logs and uses to populate the _introspection index. It generates data about your Splunk instance and environment and writes that data to log files to aid in reporting on system resource utilization and troubleshooting problems with your Splunk Enterprise deployment. You can also view the latest instrumentation data at REST endpoints.

Platform instrumentation is included in Splunk Enterprise as an add-on, sometimes referred to as the introspection_generator_addon.

Supported platforms

- Windows
  - x86-32: Server 2008, Server 2008 R2
- Linux
  - x86-64: 2.6 or later kernel
  - x86-32: 2.6 or later kernel
- Solaris
  - x86-64: 10, 11
  - SPARC: 10, 11

What data does Splunk Enterprise record in these introspection log files?

The introspection files contain data about:

- Operating system resource usage for Splunk Enterprise processes, broken down by process.
- Operating system resource usage for the entire host (i.e., all system and user processes).
- Disk object data.
- KV store performance data.

See "What data gets logged" for more information.

Where is this data written?

Events are written to two log files in $SPLUNK_HOME/var/log/introspection. Non-forwarders tail these log files and place results into the local _introspection index. Forwarders, which have no local indexes, forward these events to indexers.

The two log files are disk_objects.log and resource_usage.log. See "What gets logged" for a breakdown of what data goes into which file.

To find platform instrumentation events, qualify your searches:

- Find introspection data:

  index=_introspection
To find introspection data from a forwarder or another instance in your deployment, qualify your search with the remote host name.

**How does this feature affect my Splunk deployment?**

If you are upgrading from a Splunk Enterprise version pre-6.1, expect the new log files to use a bit of disk space (an estimated 300 MB). The _introspection index's disk usage, on the other hand, varies from deployment to deployment.

Each log file has a maximum size of 25 Mb. You can change this limit in log.cfg. You can have up to six instances of each file, according to your log rotation policy. That is, resource_usage.log, resource_usage.log.1, ... resource_usage.log.5, and the same for disk_objects.log. Thus, the introspection log files by default can take up to 300 MB of disk space.

This feature is implemented as an auxiliary low-profile long-running process. This process is where resource usage (RU) introspection data is collected. Collecting disk object (DO) introspection data requires no extra I/O, as it leverages information that other parts of splunkd have already collected and cached.

See the upgrade docs in the Installation Manual for upgrade information.

See "Configure platform instrumentation" for instructions on tuning this feature.

**What does platform instrumentation log?**

This topic describes the contents of log files that are tailed to populate the _introspection index. For the log files that populate _internal, see What Splunk logs about itself.

These log files comply with the Common Information Model (CIM). See the CIM add-on documentation for more information.

"Extra field" indicates a field that is not logged by default. Read more about configuring polling intervals and enabling this feature on a universal forwarder in Configure platform instrumentation.

**resource_usage.log**

**Per-process resource usage data**

Platform instrumentation exposes OS resource usage info for just Splunk software processes, broken down by process. Splunk processes include splunkd, splunkweb, Splunk search processes, splunkd-launched (fsck, splunk-optimize), and modular or scripted inputs launched on behalf of splunkd.

These fields are available:

- in the log file $SPLUNK_HOME/var/log/introspection/resource_usage.log
- in an indexer's _introspection index
- at the endpoint server/status/resource-usage/splunk-processes.

**Data available for all Splunk software processes**

You can view information about operating system resource utilization, broken down by Splunk process. Four fields here are "extra" fields, not logged by default. Read about populating extra fields in Configure platform instrumentation.
See the list of output fields at system/server/status/resource-usage/splunk-processes in the REST API Reference Manual.

**Additional data available only for search processes**

Splunk software can log all the above data for search processes (except `args`). In addition, it logs some additional information about search processes, in a subsection called `search_props`.

See the list of output fields at system/server/status/resource-usage/splunk-processes in the REST API Reference Manual. The search process fields are embedded within the larger process table, at the `search_props` entry.

**Hostwide resource usage data**

You can view host-level, dynamic CPU utilization and paging information.

These fields are available:

- in the log file `resource_usage.log`
- in an indexer's `_introspection` index
- at the endpoint `server/status/resource-usage/hostwide`

See the list of output fields at system/server/status/resource-usage/hostwide in the REST API Reference Manual.

**I/O statistics**

Disk input-output usage statistics. The Splunk Enterprise iostats endpoint displays the most recent data. Historical data is logged to `resource_usage.log`.

Note that the statistics available here are usage statistics, not benchmarks.

See the list of output fields at server/status/resource-usage/iostats in the REST API Reference Manual.

**Search infrastructure data**

Unlike most data available under `server/introspection`, the search infrastructure data is logged in `metrics.log` and `audit.log`, which is indexed to `_internal` and `_audit`, respectively, and available in the file system at `$SPLUNK_HOME/var/log/splunk`. Read about `metrics.log` components in “About `metrics.log`.”

**server/introspection/search/dispatch**

Provides vital statistics for distributed search framework, including details on search peer performance.

**disk_objects.log**

This disk object data is available in the log file `$SPLUNK_HOME/var/log/introspection/disk_objects.log`

Additionally, the latest snapshot of these field values is available at endpoints as itemized below.

**server/info**

Splunk Enterprise server configuration information (static server characteristics; dynamic characteristics go under `server/status`).
See the list of output fields at system/server/info in the REST API Reference Manual.

**data/index-volumes**

Lists the Splunk Enterprise volume(s).

See the list of output fields at data/index-volumes in the REST API Reference Manual.

**data/index-volumes/{Name}**

Characterizes persisted objects at the volume level.

See the list of output fields at index/data/index-volumes/{Name} in the REST API Reference Manual.

**data/indexes-extended**

Provides information about Splunk Enterprise index buckets.

See the list of output fields at index/data/indexes-extended in the REST API Reference Manual.

**data/indexes-extended/{Name}**

Provides bucket-level information for the specified index.

See the list of output fields at data/indexes-extended{Name} in the REST API Reference Manual.

**server/status/dispatch-artifacts**

Accesses search job information.

See the list of output fields at server/status/dispatch-artifacts in the REST API Reference Manual.

**server/status/fishbucket**

Accesses information about the private BTree database. Gives an idea of fishbucket growth. The fishbucket is a directory, $SPLUNK_DB/fishbucket/splunk_private_db/, that keeps a record about each file input. Most fundamentally, this record keeps track of how far into the file we've read, so that if splunkd is stopped and then restarted, it'll know where in each file input to resume reading.

See the list of output fields at server/status/fishbucket in the REST API Reference Manual

**server/status/limits/search-concurrency**

Search concurrency limits for a standalone Splunk Enterprise instance.

See the list of output fields at system/server/status/limits/search-concurrency in the REST API Reference Manual.

**server/status/partitions-space**

Helps track disk usage. These results show only partitions with Splunk disk objects (indexes, volumes, logs, fishbucket, search process artifacts) on them. There is a partitions event for each file system, and each event gives the respective file system type.
A file system (or "volume" in Windows) is a logical concept, identified on UNIX by a number called "device ID." A file system has the property of type (format). For example: ZFS, EXT3.

A partition is a physical concept, simply a chunk of hard drive (or solid state drive). All we know about a partition is its size. A file system can reside on multiple partitions. Splunk Enterprise does not report at the partition level.

See the list of output fields at server/status/partitions-space in the REST API Reference Manual.

**Configure platform instrumentation**

This topic is about log files that are tailed to populate the \_introspection index. Read about this feature in "About Splunk Enterprise platform instrumentation."

This topic helps you configure the default logging interval and enable or disable logging.

**What is logged, and how frequently**

Platform instrumentation is enabled by default on all Splunk Enterprise instances except for universal forwarders.

This table summarizes the default settings:

<table>
<thead>
<tr>
<th>Instance type</th>
<th>Resource usage</th>
<th>Disk objects: indexes, bucket superdirectories, volumes, search dispatch artifacts</th>
<th>Disk objects: fishbucket, partitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universal forwarder</td>
<td>every 600 sec (disabled by default)</td>
<td>N/A (UFs do not have indexes)</td>
<td>every 600 sec</td>
</tr>
<tr>
<td>non-UFs</td>
<td>every 10 sec</td>
<td>every 600 sec</td>
<td>every 600 sec</td>
</tr>
</tbody>
</table>

See "What gets logged" for details about what data is logged.

**Enable logging on a universal forwarder**

The introspection generator add-on is disabled by default on a universal forwarder. To enable: in the forwarder's $SPLUNK_HOME/etc/apps/introspection_generator_addon/local/app.conf, set

```
[install]
state = enabled
```

**Enable the introspection generator add-on using deployment server**

To facilitate the management of collecting introspection logs from Splunk Universal Forwarders, we will use the Splunk Deployment Server to enable the introspection generator add-on.

**Prerequisites**

The instructions require the use of a deployment server running Splunk Enterprise 6.2 or later. Additionally, you must have command line access to the deployment server host, as the changes cannot be completed using the Forwarder Management interface provided with the deployment server.
The introspection generator add-on is only available on Splunk Enterprise version 6.1 or later. All forwarder instances must be configured as deployment clients to a centralized deployment server.

**Configure the introspection generator add-on on the deployment server**

1. SSH into the deployment server.
2. Find the Splunk Enterprise installation path on the local machine. The default installation path is: /opt/splunk
3. Create a new folder: $SPLUNK_HOME/etc/deployment-apps/introspection_generator_addon
4. Create a new folder: $SPLUNK_HOME/etc/deployment-apps/introspection_generator_addon/local
5. Create an app.conf file under $SPLUNK_HOME/etc/deployment-apps/introspection_generator_addon/local
6. Edit the app.conf file and enable the add-on by adding:

   [install]
   state = enabled

7. Save the changes. Review the changes to the app.conf file and the path as a validation step.

**Review the excludeFromUpdate command**

The excludeFromUpdate prevents the deployment server from overwriting the contents of defined folders in an app. For more examples, see the "serverclass.conf" in the Admin Manual.

For this task, we will use excludeFromUpdate to enable the introspection generator add-on, while preventing the deployment server from making any changes to the add-on by blocking it from overwriting the contents in the app/introspection_generator_addon/default and app/introspection_generator_addon/bin folders.

**Update the serverclass.conf file, adding the app to a serverclass for deployment**

1. Find the primary copy of the serverclass.conf file. The location and contents will vary between deployments, but some common locations are: $SPLUNK_HOME/etc/system/local/, and $SPLUNK_HOME/etc/apps/*/local. To use btool to find all serverclass.conf files referenced on the deployment server, run: ./splunk btool --debug serverclass list and review the output.

2. Create a new app definition for deploying the changes to the introspection generator add-on. This task is dependent upon the local environment and how the Splunk administrator has chosen to assign and manage apps deployed to forwarders. Many deployments use one serverclass definition to deploy and manage the most common apps for forwarders. For the purposes of this procedure, all universal forwarders are included under one encompassing serverclass named PrimaryForwarders.

3. Define the field excludeFromUpdate command at the app level.

   [serverClass:PrimaryForwarders:app:introspection_generator_addon]
   excludeFromUpdate = $app_root$/default, $app_root$/bin
   restartSplunkd = True

4. Save the changes. Review the changes to the serverclass.conf file and the path as a validation step.

**Reload the deployment server**

1. Utilize your enterprise change control system to file the requirements and changes for this procedure.

2. Run ./splunk reload deploy-server to reload the deployment server and present the changes to all forwarder hosts at their next check-in interval. The command can be scripted to run on the deployment server after working hours.
**Validate changes have been successfully deployed**

Use the search head to validate the introspection logs are being forwarded. Example: `index_-_introspection host=<forwarder_host> | stats count by source, component`

**Populate "Extra" fields**

Four fields (in per-process resource usage data) are not populated by default but can be turned on. See "What gets logged" for information.

In server.conf you can tell Splunk Enterprise to acquire the "Extra" fields by setting `acquireExtra_i_data` to true. For example:

```plaintext
[introspection:generator:disk_objects]
disabled = false
acquireExtra_i_data = true
collectionPeriodInSecs = 600
```

**Increase the polling period**

**Why might you want to increase the polling period?**

Search processes are polled every 10 seconds (600 seconds on a universal forwarder) by a low-profile process. For healthy Splunk Enterprise deployments, we do not expect this to cause any performance problems. But on a deployment that is already prone to performance problems such as a slow pooled search head environment, there might be some performance implications.

**Configure by collection type**

In server.conf you can increase the polling period by collection type (that is, resource usage data or disk object data).

The default settings (for anything other than a universal forwarder) are:

```plaintext
[introspection:generator:disk_objects]
disabled = false
acquireExtra_i_data = false
collectionPeriodInSecs = 600
```

```plaintext
[introspection:generator:resource_usage]
disabled = false
acquireExtra_i_data = false
collectionPeriodInSecs = 10
```

On a universal forwarder, the default resource usage collection period is 600 seconds.

**Disable logging**

It is possible to disable introspection logging, although in most cases, it's preferable to merely increase the polling interval.
**Turn off all introspection logging**

You can turn off all introspection collection (and subsequent logging) by disabling the Introspection Generator Add-On.

In the `$SPLUNK_HOME/etc/apps/introspection_generator_addon/local/app.conf` file, set

```
[install]
state = disabled
```

**Turn off introspection logging at the component level**

In `server.conf` you can disable, enable, and configure collection by collection type. That is, resource usage data or disk object data.

The default settings are:

```
[introspection:generator:disk_objects]
disabled = false
acquireExtra_i_data = false
collectionPeriodInSecs = 600

[introspection:generator:resource_usage]
disabled = false
acquireExtra_i_data = false
collectionPeriodInSecs = 10
```

**Run resource usage logging from the command line**

If you've disabled this logging on your instance, you can still invoke the CLI command. To invoke, at the command line:

```
$ splunkd instrument-resource-usage [--debug] [--once] [--extra]
```

where the flags mean:

--debug: Set logging level to DEBUG (this can also be done via `log-cmdline.cfg`)

--once: Emit one set of introspection data, and then quit

--extra: This has the same effect as setting `acquireExtra_i_data` to true in the `server.conf` `[introspection:generator:resource_usage]` stanza. See "What gets logged" for which fields are not logged by default and require this flag.

**Change the location of the _introspection index**

In `indexes.conf` you can specify the _introspection index. The default location is in `$SPLUNK_DB`:

```
[_introspection]
homePath   = $SPLUNK_DB/_introspection/db
coldPath   = $SPLUNK_DB/_introspection/colddb
thawedPath = $SPLUNK_DB/_introspection/thaweddb
maxDataSize = 1024
frozenTimePeriodInSecs = 1209600
```
**Sample platform instrumentation searches**

This topic introduces a few examples of analysis you can perform using Splunk Enterprise platform instrumentation. Read About Splunk Enterprise platform instrumentation for an introduction to the feature.

**Aggregate median physical memory usage per search type**

Use this search to find the median total physical memory used in MB, per search type (ad hoc, scheduled, report acceleration, data model acceleration, or summary indexing) for one host over the last hour:

```
index=_introspection host=<hostname> data.search_props.sid=* earliest=-1h | bin _time span=10s|stats
latest(data.mem_used) as mem_used by data.search_props.sid, data.search_props.type, _time | stats
sum(mem_used) as mem_used by data.search_props.sid, data.search_props.type, _time | timechart
median(mem_used) by data.search_props.type
```

As a stacked column chart, this search produces a visualization that looks like this:

![Stacked Column Chart](image)

**Current disk usage per partition in use by Splunk Enterprise**

Use this search to find the latest value of Splunk Enterprise disk usage per partition and instance:

```
| rest /services/server/status/partitions-space | eval usage = capacity - free | eval pct_usage =
round(usage / capacity * 100, 2) | stats first(fs_type) as fs_type first(usage) as usage first(capacity) as capacity first(pct_usage) as pct_usage by mount_point, splunk_server
```

![Current Disk Usage](image)

**Median CPU usage for the main splunkd process for one host**

Use this search to find the median CPU usage of the main splunkd process for one host over the last hour:

```
index=_introspection component=PerProcess host=<hostname> data.process=splunkd (data.args="-p * start" OR data.args="service") earliest=-1h | timechart median(data.pct_cpu) as cpu_usage(%)]
```

Fill in "<hostname>" with the "host" metadata field associated with your instance, as recorded in inputs.conf's "host" property. As an area chart, this search produces something like this:
Median search concurrency by search mode for all instances

Use this search to find the median number of searches running at any given time, split by mode (historical, historical batch, real-time, or real-time indexed):

```
index=_introspection data.search_props.sid=* earliest=-1h | bin _time span=10s|stats dc(data.search_props.sid) as search_count by data.search_props.mode, _time | timechart median(search_count) by data.search_props.mode
```

Peak splunkweb file descriptor usage over time for one instance

```
index=_introspection component=PerProcess host="<hostname>" (data.process="python*" data.args="*/mrsparkle/root.py*") OR data.process=splunkweb | timechart max(data.fd_used) as fd_used
```

Fill in "<hostname>" with the "host" metadata field associated with your instance, as recorded in inputs.conf's "host" property.
Contact Splunk Support

Contact Support

For contact information, see the main Support contact page.

For detailed information about working with Splunk Support, see Working with Support and the Support Portal.

Here is some information on tools and techniques Splunk Support uses to diagnose problems. Many of these you can try yourself.

**Note:** Before you send any files or information to Splunk Support, verify that you are comfortable with sending it to us. We try to ensure that no sensitive information is included in any output from the commands below and in "Anonymize data samples to send to Support" in this manual, but we cannot guarantee compliance with your particular security policy.

**Diagnostic files**

The diag command collects basic info about your Splunk server, including Splunk's configuration details (such as the contents of `$SPLUNK_HOME/etc` and general details about your index, like the host and source names). It does not include any event data or private information.

Be sure to run diag as a user with appropriate access to read Splunk files. On *nix, typically the user you run the splunk service under, such as 'splunk', while on Windows typically the domain user you run splunk as, or some kind of local administrator if you run as "LocalSystem".

See Generate a diag in this manual for instructions on the diag command.

**Upload to your case**

You can upload a diag or other file to your open support case using the diag command. See Generate a diag in this manual.

Alternately, you can upload files to your case by going to these web pages:

- Community Members: https://www.splunk.com/index.php/send_to_splunk

Make sure the user who uploads the file has read permissions to the diag*.tar.gz file.

**Core Files**

To collect a core file if Support asks you for one, use `ulimit` to remove any maximum file size setting before starting Splunk.

```
# ulimit -c unlimited
# splunk restart
```
This setting only affects the processes you start from the shell where you ran the ulimit command. To find out where core files land in your particular UNIX flavor and version, consult the system documentation. The below text includes some general rules that may or may not apply.

On UNIX, if you start Splunk with the --nodaemon option (`splunk start --nodaemon`), it may write the core file to the current directory. Without the flag the expected location is / (the root of the filesystem tree). However, various platforms have various rules about where core files go with or without this setting. Consult your system documentation. If you do start splunk with --nodaemon, you will need to, in another shell, start the web interface manually with `splunk start splunkweb`.

Depending on your system, the core may be named something like core.1234, where '1234' is the process ID of the crashing program.

**LDAP configurations**

If you are having trouble setting up LDAP, Support will typically need the following information:

- The `authentication.conf` file from `$SPLUNK_HOME/etc/system/local/`.
- An ldif for a group you are trying to map roles for.
- An ldif for a user you are trying to authenticate as.

In some instances, a debug `splunkd.log` or `web_service.log` is helpful.

**How to file a great Support case**

When you contact Support, you can save time by starting out with everything we'll need!

Here are some ideas to get you started.

**Describe the issue**

Where does the issue occur? On a forwarder? On an indexer?

What elements are present for the issue? What's the timeline leading to the error? What processes are running when the error appears?

What behavior do you observe, compared to what you expect? Be specific: for example, how late is "late"?

Try to classify the problem:

- Is it a searching issue? These include Splunk Web, management, roles, apps, views and dashboards, search language.
- Is it a back end issue? These problems could include crashing, OS issues, REST API, or SDK.
- Is it a configuration issue? These include extractions, input configurations, forwarding, apps disabling, or authentication.
- Is it a performance problem?
Send diagnosis files

Most Support cases are for functional problems: the software has been configured to do something, but it is behaving in an unexpected way. Splunk Support needs both the context of the problem and insight into the instance that is not performing as expected. That insight comes in the form of a "diag" or diagnostic file, which is essentially a snapshot of the configuration of the Splunk platform instance and the recent logs from that instance.

You can make a diag on any instance type: forwarder, indexer, search head, or deployment server. If you have a forwarder and a receiver that are not working together correctly, send us diags of both. Label the diags so it's clear which instance each is from. If you have many forwarders, send only one representative forwarder diag.

The diag tarball does not contain any of your indexed data, but you can examine its contents before sending it. Read about what you can include or exclude from diags in Generate a diagnostic file in this manual.

Splunk Support might request another diag after recommending a change or update to the instance. This diag can verify that the change has been applied and examine its effect. It is not unusual to have multiple updated diags for a single case. If you send multiple diags, label each one clearly.

Generate a diag

To help diagnose a problem, Splunk Support might request a diagnostic file from you. Diag files give Support insight into how an instance is configured and how it has been operating up to the point that the diag command was issued.

About diag

The diag command collects basic information about your Splunk platform instance, including Splunk's configuration details. It gathers information from the machine such as server specs, OS version, file system, and current open connections. From the Splunk platform instance it collects the contents of $SPLUNK_HOME such as app configurations, internal Splunk log files, and index metadata.

Diag does not collect any of your indexed data and we strongly encourage you to examine the tarball to ensure that no proprietary data is included. In some environments, custom app objects, like lookup tables, could potentially contain sensitive data. Exclude a file or directory from the diag collection by using the --exclude flag. Read on for more details.

Note: Before you send any files or information to Splunk Support, verify that you are comfortable sending it to us. We try to ensure that no sensitive information is included in any output from the commands below and in "Anonymize data samples to send to Support" in this manual, but we cannot guarantee compliance with your particular security policy.

Run diag with default settings

Be sure to run diag as a user with appropriate access to read Splunk files.

On *nix: $SPLUNK_HOME/bin

./splunk diag

On Windows: %SPLUNK_HOME%\bin
splunk diag

If you have difficulty running diag in your environment, you can also run the python script directly from the bin directory using cmd.

On *nix:

```
./splunk cmd python $SPLUNK_HOME/lib/python2.7/site-packages/splunk/clilib/info_gather.py
```

On Windows:

```
splunk cmd python %SPLUNK_HOME%\Python-2.7\Lib\site-packages\splunk\clilib\info_gather.py
```

**Note:** The python version number may differ in future versions of Splunk Enterprise, affecting this path.

This produces diag-<server name>-<date>.tar.gz in your Splunk home directory, which you can upload to your Splunk Support case via the website or built-in upload functionality. If you’re having trouble with forwarding, Support will probably need a diag for both your forwarder and your receiver. Label each diag so it’s clear which is from the forwarder and which is from the receiver.

**Designate content for diag to include or exclude**

Diag can be told to leave some files out of the diag. One way to do this is with path exclusions. At the command line you can use the `--exclude` flag. For example:

```
splunk diag --exclude "*/passwd"
```

This is repeatable:

```
splunk diag --exclude "*/passwd" --exclude "*/dispatch/*"
```

**Note:** File names excluded by the `--exclude` feature are listed in excluded_filelist.txt in the diag bundle to ensure Splunk Support can interpret the diag.

**Components**

A more robust way to exclude content is with components. The following options select which categories of information should be collected.

```
--collect=<list>  Declare a set of components to gather, as a comma-separated list, overriding any prior choices
--enable=<component_name>  Add a component to the work list
--disable=<component_name>  Remove a component from the work list
```

The available components are as follows.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Options</th>
</tr>
</thead>
</table>

37
<table>
<thead>
<tr>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conf_replication_summary</td>
<td>A directory listing of replication summaries produced by search head clustering</td>
</tr>
<tr>
<td>consensus</td>
<td>Copies of the consensus protocol files used for search head cluster member coordination from var/run/splunk/raft</td>
</tr>
<tr>
<td>dispatch</td>
<td>The search dispatch directories. See Dispatch directory and search artifacts in the Search Manual.</td>
</tr>
<tr>
<td>etc</td>
<td>The entire contents of the $SPLUNK_HOME/etc directory, which contains configuration information, including .conf files.</td>
</tr>
<tr>
<td>file_validate</td>
<td>The results of the latest file integrity check. See Check the integrity of your Splunk software files in the Admin Manual.</td>
</tr>
<tr>
<td>index_files</td>
<td>Files from the index that describe their contents. (Hosts.data, Sources.data, Sourcetypes.data, and bucketManifests). User data is not collected. If diag collects index files on larger deployments, it might take a while to run. Read about index files in the Splexicon.</td>
</tr>
<tr>
<td>index_listing</td>
<td>Directory listings of the index contents are gathered, in order to see file names, directory names, sizes, timestamps, and the like. This information is recorded in systeminfo.txt.</td>
</tr>
<tr>
<td>kvstore</td>
<td>Directory listing of the Splunk key value store files.</td>
</tr>
<tr>
<td>log</td>
<td>The contents of $SPLUNK_HOME/var/log/... See What Splunk Enterprise logs about itself.</td>
</tr>
</tbody>
</table>

### etc

- By default, diag excludes lookup files in etc/apps and etc/users starting in Splunk Enterprise version 6.5.0. To include lookups, use the option `--include-lookups`.
- By default, diag excludes files in $SPLUNK_HOME/etc larger than 10 MB. To modify this limit, use `--etcfilesize-limit=<level>`, where level is the file size in kilobytes and 0 disables this filter.

### log

- Set the log age to gather using `--log-age=<days>`. Log files over this many days old are not included, 0 disables this filter. Default: 60.
- By default diag gathers at most three Windows crash .dmp files. To gather every .dmp file, use `--all-dumps=<bool>`.
- Fully gather files in $SPLUNK_HOME/var/log smaller than the size specified by `--logfilesize-limit=size`. For log files larger than this size, gather only this many bytes from the end of the file (capture truncated trailing bytes). [default: 1GB]
- To redact search terms from audit.log and remote_searches.log, use `--filter-searchstrings`. To not modify these log files, use `--no-filter-searchstrings`.
If search head pooling is enabled, the contents of the pool directory will be excluded. By default, diag excludes lookup files starting in Splunk Enterprise version 6.5.0. To include lookups, use the option --include-lookups.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>rest</td>
<td>splunkd httpd REST endpoint gathering. Collects output of various splunkd urls into xml files to capture system state. Off by default.</td>
<td></td>
</tr>
<tr>
<td>searchpeers</td>
<td>Directory listing of the &quot;searchpeers&quot; location, actually the data provided by search<em>heads</em> on indexers/search nodes.</td>
<td></td>
</tr>
<tr>
<td>app:&lt;app_name&gt;</td>
<td>If you have an app installed that extends diag, adding apps-specific troubleshooting data, it will offer a component similar to this. For information on what type of data the app provides, see the app documentation, review the content stored in the produced tar file, or contact the app developers.</td>
<td>An app might offer additional app-specific flags, in the form --app_name:setting. For example, the most commonly requested files collected are log files and configuration files only for initial analysis. To collect only those two components, use: $SPLUNK_HOME/bin/splunk diag --collect=log,etc.</td>
</tr>
</tbody>
</table>

Defaults can also be controlled in server.conf. Refer to server.conf.spec in the Admin Manual for more information. Apps do not currently offer defaulting of their settings in server.conf.

**Search string redaction**

Diag by default removes some types of sensitive information from search strings in diag files. Read about configuring search string redaction in server.conf.spec.

These options cause diag to redact or hide data from the output diag.

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--filter-searchstrings</td>
<td>Attempt to redact search terms from audit.log &amp; remote_searches.log that may be private or personally identifying</td>
</tr>
<tr>
<td>--no-filter-searchstrings</td>
<td>Do not modify audit.log &amp; remote_searches.log</td>
</tr>
</tbody>
</table>

**Run diag on a remote instance**

If you are not able to SSH into every machine in your deployment, you can still gather diags from full Splunk platform installations, but not from universal forwarders. First, make sure you have the get_diag capability. The admin role has this capability by default. You also need login credentials for the remote server.

The syntax is:

```
splunk diag -uri https://<host>:<mgmtPort>
```

The options recognized for remote diag collection from the command line are --basename, --all-dumps, and exclude.

**Upload a file to Splunk Support**

If you have a support case already open, you can attach a diag at the conclusion of generating the output. Alternatively, you can upload a file that already exists, such as a previously generated diag or other debugging data.

To generate and upload a diag, the syntax is the following:

```
splunk diag --upload
```

To upload a file you already have, the syntax is the following:

```
splunk diag --upload-file=a-filename.zip
```

This command interactively prompts for values such as a splunk.com user name and password, choice of open cases for that user, and a description of the upload.

Optionally, you can perform the upload work non-interactively, by providing the required values as flags:

```
[...]
--case-number=case-number
  Case number to attach to, e.g. 200500
--upload-user=UPLOAD_USER
  splunk.com username to use for uploading
--upload-password=UPLOAD_PASSWORD
  splunk.com password to use for uploading
--upload-description=UPLOAD_DESCRIPTION
  description of file upload for Splunk support
--firstchunk=chunk-number
  For resuming upload of a multi-part upload; select the first chunk to send
```

User names on splunk.com do not include @domain.com. The --firstchunk flag matters only if uploading a huge file fails after partial success. In this case, the diag output explicitly tells you the command to use to retry.

For example:

```
splunk diag --upload --case-number=757096 --upload-user=jawlige --upload-password=<passwd>
--upload-description="Monday diag, as requested."
```

**Examples**

**Exclude a lookup table**

These two examples exclude content on the file level. A lookup table can be one of several formats, like .csv, .dat, or text.

Exclude all .csv files, or all .dat files, in $SPLUNK_HOME:

```
splunk diag --exclude "*.csv" or
splunk diag --exclude "*.dat"
```

**Note:** These examples exclude all files of that type, not only lookup tables. If you have .csv or .dat files that will be helpful for Support in troubleshooting your issue, exclude only your lookup tables. That is, write out the files instead of using an asterisk.

**Exclude the dispatch directory**

This example excludes content on the component level. Exclude the dispatch directory to avoid gathering search artifacts (which can be very costly on a pooled search head):

```
$SPLUNK_HOME/bin/splunk diag --disable=dispatch
```

**Exclude multiple directories**

To exclude multiple components, use the --disable flag once for each component.

Exclude the dispatch directory and all files in the shared search head pool:

```
$SPLUNK_HOME/bin/splunk diag --disable=dispatch --disable=pool
```

39
**Note:** This does not gather a full set of the configuration files in use by that instance. Such a diag is useful only for the logs gathered from `$SPLUNK_HOME/var/log/splunk`. See What Splunk Enterprise logs about itself in this manual.

**Gather only logs**

To include only the Splunk Enterprise internal log files:

```
$SPLUNK_HOME/bin/splunk diag --collect=log
```

**Generate a diag, then upload it**

```
$SPLUNK_HOME/bin/splunk diag --upload
```

**Fetch a diag from a remote instance, then upload it**

```
$SPLUNK_HOME/bin/splunk diag --uri https://splunkserver.example.com:8089
$SPLUNK_HOME/bin/splunk diag --upload-file=<diag_from_prior_command>
```

**Save the settings for diag in server.conf**

You can update the default settings for diag in the `[diag]` stanza of server.conf.

```
[diag]
EXCLUDE=<class> = <glob expression>
* Specifies a glob / shell pattern to be excluded from diags generated on this instance.
* Example: */etc/secret_app/local/*.conf
```

Flags that you append to `splunk diag` override server.conf settings.

**Diag contents**

Primarily, a diag contains server logs, from `$SPLUNK_HOME/var/log/splunk` and `$SPLUNK_HOME/var/log/introspection`, and the configuration files, from `$SPLUNK_HOME/etc`.

Specifically, by pathname, there is:

- `_raft/...`
  Files containing the state of the consensus protocol produced by search head clustering from `var/run/splunk/_raft`

- `composite.xml`
  The generated file that splunkd uses at runtime to control its component system (pipelines & processors), from `var/run/splunk/composite.xml`

- `diag.log`
  A copy of all the messages diag produces to the screen when running, including progress indicators, timing, messages about files excluded by heuristic rules (eg if size heuristic, the setting and the size of the file), errors, exceptions, etc.

- `dispatch/...`
  A copy of some of the data from the search dispatch directory. Results files (the output of searches) are not included, nor other similar files (events/*)

- `etc/...`
  A copy of the contents of the configuration files. All files and directories under `$SPLUNK_HOME/etc/auth` are excluded by default.

`excluded_filelist.txt`
A list of files which diag would have included, but did not because of some restriction (exclude rule, size restriction). This is primarily to confirm the behavior of exclusion rules for customers, and to enable Splunk technical support to understand why they can't see data they are looking for.

**introspection/**
- The log files from $SPLUNK_HOME/var/log/introspection

**log/**
- The log files from $SPLUNK_HOME/var/log/splunk

**rest-collection/**
- Output of several splunkd http endpoints that contain information not available in logs. File input/monitor/tailing status information, server-level admin banners, clustering status info if on a cluster.

**scripts/**
- A single utility script may exist here for support reasons. It is identical for every diag.

**systeminfo.txt**
- Generated output of various system commands to determine things like available memory, open splunk sockets, size of disk/filesystems, operating system version, ulimits.
- Also contained in systeminfo.txt are listings of filenames/sizes etc from a few locations.
  ◊ Some of the splunk index directories (or all of the index directories, if full listing is requested.)
  ◊ The searchpeers directory (replicated files from search heads)
  ◊ Search Head Clustering -- The summary files used in synchronization from var/run/splunk/snasphot

**Typically var/**
- The paths to the indexes are a little 'clever', attempting to resemble the paths actually in use (For example, on windows if an index is in e:\someother\largedrive, that index's files will be in e\someother\largedrive inside the diag). By default only the .bucketManifest for each index is collected.

**app_ext/<app_name>/**
- If you have an app installed which extends diag, the content it adds to the produced tar.gz file will be stored here.

### Behavior on failure

If for some reason diag fails, it does the following:

1. Cleans up temporary files it created while running.
2. Leaves a copy of the output in a temporary filename it references.

For example:

rjodman@mcp:~$ splunk/bin/splunk diag
[... lots of normal output...]
Selected diag name of: diag-mcp-2014-09-24
Starting splunk diag...
[etc .... etc]
Getting index listings...
Copying Splunk configuration files...
Exception occurred while generating diag, we are deeply sorry.
Traceback (most recent call last):
  File "/opt/splunk/lib/python2.7/site-packages/splunk/clilib/info_gather.py", line 1959, in main
    create_diag(options, log_buffer)
  File "/opt/splunk/lib/python2.7/site-packages/splunk/clilib/info_gather.py", line 1862, in create_diag
    copy_etc(options)
  File "/opt/splunk/lib/python2.7/site-packages/splunk/clilib/info_gather.py", line 1626, in copy_etc
    raise Exception("OMG!")
Exception: OMG!

Diag failure, writing out logged messages to '/tmp/diag-fail-F2B94h.txt', please send output + this file to
For most real errors, diag tries to guess at the original problem, but it also writes out a file for use in bugfixing diag. Send this file to Support, and at least a workaround can often be provided quickly.

Additional resources

Watch a video on making a diag and using the anonymize command by a Splunk Support engineer:

Video link: "https://www.youtube.com/watch?v=FTK7iTcyrXI?rel=0"

Have questions? Visit Splunk Answers and see what questions and answers the Splunk community has about diags.

Anonymize data samples to send to Support

Splunk Enterprise has a few methods to anonymize data in files you send to Support. This lets Splunk Enterprise users share log data without revealing confidential or personal information from their networks.

Diag by default removes some types of sensitive information from search strings in diag files. Read about configuring search string redaction in server.conf.spec.

The anonymize function combs through sample log files or event files to replace identifying data - like usernames, IP addresses, domain names - with fictional values that maintain the same word length and event type. For example, it might turn the string `user=carol@adalberto.com` into `user=plums@wonderful.com`.

The anonymized file is written to the same directory as the source file, with ANON- prepended to its filename. For example, `/tmp/messages` will be anonymized as `/tmp/ANON-messages`. In Windows, a file `\temp\messages` becomes `\temp\ANON-messages`.

Anonymize is controlled from the Splunk Enterprise CLI. See About the CLI for instructions on accessing the Splunk Enterprise CLI.

Simple method

The easiest way to anonymize a file is with the anonymizer tool's defaults, as shown in the session below.

From the CLI while you are in `$SPLUNK_HOME/bin` or `%SPLUNK_HOME%\bin`, type the following:

<table>
<thead>
<tr>
<th>Unix/Linux</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>./splunk anonymize file -source &lt;/path/to/filename&gt;</code></td>
<td><code>.\splunk anonymize file -source &lt;\path\to\filename&gt;</code></td>
</tr>
</tbody>
</table>

It is good practice to copy the file somewhere safe (like /tmp or \temp) before performing this command.

Unix/Linux example:

```
> cp -p /var/log/messages /tmp
> cd $SPLUNK_HOME/bin
> ./splunk anonymize file -source /tmp/messages
Processing files: ['/tmp/messages']
```
Getting named entities
Processing /tmp/messages
Adding named entities to list of public terms: Set({'secErrStr', 'MD_SB_DISKS', 'TTY', 'target', 'precision', 'lpj', 'ip', 'pci', 'hard', 'last bus', 'override with idebus', 'SecKeychainFindGenericPassword err', 'vector', 'USER', 'irq ', 'com user', 'uid'})
Processing /tmp/messages for terms.
Calculating replacements for 4672 terms.
-----------------------------------------------------------------------------------------------------------------------------------
Wrote dictionary scrubbed terms with replacements to "/tmp/INFO-mapping.txt"
Wrote suggestions for dictionary to "/tmp/INFO-suggestions.txt"
-----------------------------------------------------------------------------------------------------------------------------------
Writing out /tmp/ANON-messages
Done.

Windows example:

C:gxcopy c:\apache\apache.error.log c:temp
C:\apache\apache.error.log
1 File(s) copied
C:gio\program files\Splunk\bin
C:\Program Files\Splunk\bin>\splunk anonymize file -source c:\temp\apache.error.log
Processing files: ['c:\temp\apache.error.log']
Getting named entities
Processing c:\temp\apache.error.log
Adding named entities to list of public terms: set([])
Processing c:\temp\apache.error.log for terms.
Calculating replacements for 44 terms.
-----------------------------------------------------------------------------------------------------------------------------------
Wrote dictionary scrubbed terms with replacements to "c:\temp\INFO-mapping.txt"
Wrote suggestions for dictionary to "c:\temp\INFO-suggestions.txt"
-----------------------------------------------------------------------------------------------------------------------------------
Writing out c:\temp\ANON-apache.error.log
Done.

Advanced method

You can customize the anonymizer by telling it what terms to anonymize, what terms to leave alone, and what terms to use as replacements.

On *nix:


On Windows:


On both Windows and *nix, the optional parameters are defined as follows:

- **filename**
  - Default: None
  - Path and name of the file to anonymize.

- **public_terms**
  - Default: $SPLUNK_HOME/etc/anonymizer/public-terms.txt or $SPLUNK_HOME\etc\anonymizer\public-terms.txt
  - A list of locally used words that will not be anonymized if they are in the file. It serves as an appendix to the dictionary file.
Here is a sample entry:

2003 2004 2005 2006 abort aborted am apr april aug august auth authorize authorized authorizing bea certificate class com complete

- **private_terms**
  - Default: `$SPLUNK_HOME/etc/anonymizer/private-terms.txt` or `%SPLUNK_HOME%\etc\anonymizer\private-terms.txt`
  - A list of words that will be anonymized if found in the file, because they may denote confidential information.
  - Here is a sample entry:

  401-51-6244
  passw0rd

- **name_terms**
  - Default: `$SPLUNK_HOME/etc/anonymizer/names.txt` or `%SPLUNK_HOME%\etc\anonymizer\names.txt`
  - A global list of common English personal names that Splunk software uses to replace anonymized words.
  - Anonymize always replaces a word with a name of the exact same length, to keep each event's data pattern the same.
  - Anonymize uses each name in `name_terms` once to replace a character string of equal length throughout the file. After it runs out of names, it begins using randomized character strings, but still mapping each replaced pattern to one anonymized string.
  - Here is a sample entry:

    charlie
    claire
    desmond
    jack

- **dictionary**
  - Default: `$SPLUNK_HOME/etc/anonymizer/dictionary.txt` or `%SPLUNK_HOME%\etc\anonymizer\dictionary.txt`
  - A global list of common words that will not be anonymized, unless overridden by entries in the `private_terms` file.
  - Here is a sample entry:

    algol
    ansi
    arco
    arpa
    arpanet
    ascii

- **timestamp_config**
  - Default: `$SPLUNK_HOME/etc/anonymizer/anonymizer-time.ini` or `%SPLUNK_HOME%\etc\anonymizer\anonymizer-time.ini`
  - File built into Splunk software that determines how timestamps are parsed.
Output Files

Splunk's anonymizer function will create three new files in the same directory as the source file.

- **ANON-filename**
  - The anonymized version of the source file.
- **INFO-mapping.txt**
  - This file contains a list of which terms were anonymized into which strings.
  - Here is a sample entry:

Replacement Mappings
----------------------
kb900485 --> LO200231
1718 --> 1608
transitions --> tstymnbxno
reboot --> SPLUNK
cdrom --> pqyvi

- **INFO-suggestions.txt**
  - A report of terms found in the file that, based on their appearance and frequency, you may want to add to public_terms.txt or to private-terms.txt or to public-terms.txt for more accurate anonymization of your local data.
  - Here is a sample entry:

Terms to consider making private (currently not scrubbed):
['uid', 'pci', 'lpj', 'hard']
Terms to consider making public (currently scrubbed):
['jun', 'security', 'user', 'ariel', 'name', 'logon', 'for', 'process', 'domain', 'audit']

**Linux tip: Anonymize all log files from a diag at once**

Here are the steps to generate a diagnostic (diag file) and then anonymize the logs of that diag.

1. Generate the diag: For example:

   ```bash
cd $SPLUNK_HOME/bin
./splunk diag --exclude "*/passwd"
```

2. Uncompress the diag. For example:

   ```bash
cd <path_to_uncompressed_diag>/
tar xzf my-diag-hostname.tar.gz
```

3. Run `anonymize` on each file of the diag. If you run this command for all *.log, then make note of the log files that now have a prefix of ANON*.log. For example:

   ```bash
find <absolute_path_to_uncompressed_diag>/ -name "*.log* | xargs -I{} ./splunk anonymize file -source '{}'
```

4. Keep all the files that now have a prefix of ANON*.log while deleting the non-anonymized versions in the diag directory.
5. Compress the diag.

```
tar cfz my-diag-hostname.tar.gz <path_to_uncompressed_diag>
```

6. Upload the diag, adding it to the Support case, with the ADD FILE button in the case.

**Collect pstacks**

Support might ask you to gather thread call stacks with pstack, for example if your deployment experiences:

- unexplained high CPU, along with identified threads using high CPU,
- frozen Splunk that’s not doing anything, when it obviously should, or
- unexplainably slow behavior in splunkd (that is, not limited by disk or CPU).

**On *nix**

**Find or install pstack**

Pstack is available on Red Hat and Centos Linux and Solaris by default. Pstack is installable on several other flavors of Linux.

Test whether pstack is installed:

```
which pstack
/usr/bin/pstack
```

If you get an error message instead of a location, you might still be able to install pstack. On RHEL and its derivatives (CentOS, Oracle Linux, etc), pstack is part of the gdb package.

**Error on Linux from pstack: no symbols**

On Linux flavors that aren't based on RHEL, pstack might be useless for troubleshooting, in that it does not support threads.

If you get output from pstack such as:

```
29175: splunkd -p 8089 start
(No symbols found)
0x7fd3740e96d9: ???? (100, 0, 7fff6bedf00, 100000010, 25bb080, ffffffff00000010) + ffff8001594106da
```

Then you probably have the x86-64-specific pstack binary, which is less capable than the redhat gdb-based one, as it does not understand posix threaded applications. Ensure that the gdb package is installed, and try the gstack command as a substitution for pstack. gstack is available on Ubuntu, for example. If gstack is not available, a very barebones gstack is provided here:

```
pid=$1
echo 'thread apply all bt' | gdb --quiet -nx /proc/$pid/exe $pid
```
Installable on nearly any Unix.

```sh
# ps aux |grep splunkd
root  31038  0.5  0.6 245292 104884 ?   Sl   Sep07  66:45 splunkd -p 17011 restart
root  31039  0.0  0.0  47012  7076 ?     Ss   Sep07   4:47 splunkd -p 17011 restart
# gdb -p 31038  #this will freeze splunk temporarily
... lots of output you don't care about ...
(gdb) <-this is the prompt
(gdb) thread apply all bt
<... interesting output here...>
(gdb) quit # important! otherwise splunk is frozen forever
```

**Run pstack**

To run pstack from the *nix command line,

```sh
# ps aux |grep splunkd
root  31038  0.5  0.6 245292 104884 ?   Sl   Sep07  66:45 splunkd -p 17011 restart
root  31039  0.0  0.0  47012  7076 ?     Ss   Sep07   4:47 splunkd -p 17011 restart
# pstack 31038
<... output here ...>
```

It is usually beneficial to get multiple pstacks separated by 1 second. Here is an example of getting 100 pstacks separated by 1 second and storing them in /tmp:

```sh
% i=0; while [ $i -lt 100 ]; do date > /tmp/pstack$i.out; pstack $splunkd_pid >> /tmp/pstack$i.out; let "i+=1"; sleep 1; done
```

Note that this script requires bash (let is not a portable expression).

**On Windows**

You can gather many pstacks at once, like with *nix:

http://wiki.splunk.com/Community:GatherWindowsStacks

**Command line tools for use with Support**

This topic contains information on CLI tools to help with troubleshooting Splunk Enterprise. Most of these tools are invoked using the Splunk CLI command "cmd".

Do not use these tools without first consulting with Splunk Support.

For general information about using the CLI in Splunk software, see Get help with the CLI in the Admin Manual.
cmd

Runs the specified utility in $SPLUNK_HOME/bin with the required environment variables preset.

To see which environment variables will be set, run "splunk envvars".

Examples:

./splunk cmd btool inputs list
./splunk cmd /bin/ls

Syntax: cmd <command> [parameters...]

Objects: None

Required Parameters: None

Optional Parameters: None

btool

View or validate Splunk software configuration files, taking into account configuration file layering and user/app context.

Syntax:

btool <CONF_FILE> list [options]
btool check [options]

Objects: None

Required Parameters: None

Optional Parameters:

--user=SPLUNK_USER  View the configuration data visible to the given user
--app=SPLUNK_APP   View the configuration data visible from the given app
--dir=DIR          Read configuration data from the given absolute path
                   instead of $SPLUNK_HOME/etc
--debug           Print and log extra debugging information

Examples:

List: ./splunk cmd btool [--app=app_name] conf_file_prefix list [stanza_prefix]

Add: ./splunk cmd btool [--app=app_name] conf_file_prefix add

Delete: ./splunk cmd btool --app=app_name --user=user_name conf_file_prefix delete stanza_name [attribute_name]
Check for typos: ./splunk cmd btool check

For more information, read Use btool to troubleshoot configurations.

btprobe

Queries the fishbucket for checkpoints stored by monitor inputs. Any changes made to the fishbucket using btprobe take effect only after a restart. Shut down your Splunk software before using btprobe. For up-to-date usage, run btprobe --help.

You must specify either -d <dir> or --compute-crc <file>.

There are two ways to invoke this tool.

1. Query a specified BTree for a given key or file.

From the Splunk software installation directory, type:

```bash
./btprobe [-h or --help] -d <btree directory> [-k <hex key OR ALL> | --file <filename>] [--salt <salt>] [--validate] [--reset] [--bytes <bytes>] [-r]
```

The options are as follows:

- `-d` Directory that contains the btree index. (Required.)
- `-k` Hex crc key or ALL to get all the keys.
- `--file` File to compute the crc from.
- `-r` Rebuild the btree .dat files (i.e., var/lib/splunk/fishbucket/splunk_private_db/)
  One of -k and --file must be specified.
- `--validate` Validate the btree to look for errors.
- `--salt` Salt the crc if --file param is specified.
- `--reset` Reset the fishbucket for the given key or file in the btree.
  Resetting the checkpoint for an active monitor input reindexes data, resulting in increased
  license use.
- `--bytes` Number of bytes to read when calculating CRC (default 256).
- `--sourcetype` Sourcetype to load configurations and check Indexed Extraction
  and compute CRC accordingly.

2. Computes a crc from a specified file, using a given salt if any.

From the Splunk software installation directory, type:

```bash
./btprobe [-h or --help] --compute-crc <filename> [--salt <salt>] [--bytes <bytes>]
```

- Example: Reset a specific file in the fishbucket:

  ```bash
  ./splunk cmd btprobe -d /opt/splunkforwarder/var/lib/splunk/fishbucket/splunk_private_db --file
  /var/log/messages --reset
  ```

- Example:

  ```bash
  ./btprobe -d /opt/splunk/var/lib/splunk/fishbucket/splunk_private_db -k 0xe8d17ddba85e714 --validate
  ```
classify

The “splunk train sourcetype” CLI command calls classify. To call it directly use:

```bash
$SPLUNK_HOME/bin/splunk cmd classify <path/to/myfile> <mysourcetypename>
```

check-rawdata-format

Unpacks and verifies the 'rawdata' component one or more buckets. 'rawdata' is the record of truth from which Splunk software can rebuild the other components of a bucket. This tool can be useful if you are worried or believe there may be data integrity problems in a set of buckets or index. Also you can use it to check for journal integrity prior to issuing a rebuild, if you wish to know whether the rebuild can complete successfully before running it.

Complementary but nonoverlapping with the splunk fsck command

```bash
splunk check-rawdata-format -bucketPath <bucket>
splunk check-rawdata-format -index <index>
splunk check-rawdata-format -allindexes
```

fsck

Diagnoses the health of your buckets and can rebuild search data as necessary. Can take a long time to run on several buckets, and you must stop Splunk software before running it. See Nonclustered bucket issues in Managing Indexers and Clusters of Indexers for help repairing buckets.

The output of splunk fsck --help is as follows:

**USAGE**

Supported modes are: scan, repair, clear-bloomfilter, check-integrity, generate-hash-files

```bash
<bucketSelector> := --one-bucket|--all-buckets-one-index|--all-buckets-all-indexes --index-name=<name>
[--bucket-name=<name>] [--bucket-path=<path>] [--include-hots] [--local-id=<id>] [--origin-guid=<guid>]
[--min-ET=<epochSecs>] [--max-LT=<epochSecs>]
```

```bash
fsck repair <bucketSelector> <otherFlags> [--bloomfilter-only] [--backfill-always] [--backfill-never]
fsck scan <bucketSelector> <otherFlags> [--metadata] [--check-bloomfilter-presence-always] [--include-rawdata]
fsck clear-bloomfilter <bucketSelector> <otherFlags>
```
fsck check-integrity <bucketSelector> fsck generate-hash-files <bucketSelector>

fsck check-rawdata-format <bucketSelector>

fsck minify-tsidx --one-bucket --bucket-path=<path> --dont-update-manifest|--home-path=<dir>

Notes: The mode verb 'make-searchable' is synonym for 'repair'. The mode 'check-integrity' will verify data integrity for buckets created with the integrity-check feature enabled. The mode 'generate-hash-files' will create or update bucket-level hashes for buckets which were generated with the integrity-check feature enabled. The mode 'check-rawdata-format' verifies that the journal format is intact for the selected index buckets (the journal is stored in a valid gzip container and has valid journal structure. Flag --log-to--splunkd-log is intended for calls from within splunkd. If neither --backfill-always nor --backfill-never are given, backfill decisions will be made per indexes.conf 'maxBloomBackfillBucketAge' and 'createBloomfilter' parameters. Values of 'homePath' and 'coldPath' will always be read from config; if config is not available, use --one-bucket and --bucket-path but not --index-name. All <bucketSelector> constraints supplied are implicitly ANDed. Flag --metadata is only applicable when migrating from 4.2 release. If giving --include-hots, please recall that hot buckets have no bloomfilters. Not all argument combinations are valid. If --help found in any argument position, prints this message & quits.

./splunk --repair works only with buckets created by Splunk Enterprise version 4.2 or later.

For more information about buckets, read How Splunk stores indexes in Managing Indexers and Clusters of Indexers.

locktest

./splunk cmd locktest

If you run Splunk Enterprise on a file system that is not listed, the software might run a startup utility named 'locktest' to test the viability of the file system. 'Locktest' is a program that tests the start up process. If 'locktest' fails, then the file system is not suitable for running Splunk Enterprise. See System Requirements for details.

locktool

./splunk cmd locktool

Usage:

lock : [-l | --lock ] [dirToLock] <timeOutSecs>

unlock [-u | --unlock ] [dirToUnlock] <timeOutSecs>

Acquires and releases locks in the same manner as splunkd. If you were to write an external script to copy db buckets in and out of indexes you should acquire locks on the db colddb and thaweddb directories as you are modifying them and release the locks when you are done.

parsetest

./splunk cmd parsetest

Usage:

parsetest "<string>" ["<sourcetype>|source::<filename>|host::<hostname>"]
parsetest file <filename> ["<sourcetype>|host::<hostname>"]
Example:
  parsetest "10/11/2009 12:11:13" "syslog"
  parsetest file "foo.log" "syslog"

**pcregextest**

Simple utility tool for testing modular regular expressions.

```
./splunk cmd pcregextest mregex=<regex>
```

Usage: pcregextest mregex="query_regex" (name="subregex_value")* (test_str="string to test regex")?

Example: pcregextest mregex="\[[ip:src_]\] \[[ip:dst_]\] " ip="(?<ip>\d+\[[dotnum]\]{3})" dotnum="\./\d+"
  test_str="1.1.1.1 2.2.2.2"

That is, define modular regex in the 'mregex' parameter. Then define all the subregexes referenced in 'mregex'. Finally you can provide a sample string to test the resulting regex against, in 'test_str'.

**searchtest**

```
./splunk cmd searchtest search
```

**signtool**

Sign

```
./splunk cmd signtool [-s | --sign] [dir to sign]
```

Verify

```
./splunk cmd signtool [-v | --verify] [dir to verify]
```

Using logging configuration at /Applications/splunk/etc/log-cmdline.cfg.

Allows verification and signing splunk index buckets. If you have signing set up in a cold to frozen script. Signtool allows you to verify the signatures of your archives.

**tsidxprobe**

This will take a look at your **time-series index files** (or "tsidx files"; they are appended with .tsidx) and verify that they meet the necessary format requirements. It should also identify any files that are potentially causing a problem.

go to the $SPLUNK_HOME/bin directory. Do “source setSplunkEnv”.

Then use tsidxprobe to look at each of your index files with this little script you can run from your shell (this works with bash):

```
  • for i in `find $SPLUNK_DB -name "*.tsidx"`; do tsidxprobe $i >> tsidxprobeout.txt; done
```

(If you've changed the default datastore path, then this should be in the new location.)
The file tsidxprobeout.txt will contain the results from your index files. You should be able to gzip this and attach it to an email and send it to Splunk Support.

### tsidx_scan.py

For Splunk Enterprise versions 4.2.2 or later, this utility script searches for tsidx files at a specified starting location, runs tsidxprobe for each one, and outputs the results to a file.

From $SPLUNK_HOME/bin, call it like this:

```bash
splunk cmd python tsidx_scan.py [path]
```

Example:

```bash
splunk cmd python tsidx_scan.py /opt/splunk/var/lib/splunk
```

If you omit the optional path, the scan starts at $SPLUNK_DB

The output is written to the file tsidxprobe.YYYY-MM-DD.txt in the current directory.

### walklex

This tool "walks the lexicon" to tell you which terms exist in a given index. For example, with some search commands (like `tstat`), the field is in the index; for other terms it is not. Walklex can be useful for debugging.

Walklex outputs a line with three pieces of information:

- term ID (a unique identifier)
- number of occurrences of the term
- term

Usage:

From $SPLUNK_HOME/bin, type

```bash
./splunk cmd walklex </path/to/tsidx_file.tsidx> "<key>::<value>"
```

It recognizes wildcards:

```bash
./splunk cmd walklex </path/to/tsidx_file.tsidx> ""
```

```bash
./splunk cmd walklex </path/to/tsidx_file.tsidx> "*::*"
```

Empty quotes return all results, and asterisks return all keys or all values (or both, as in the example above).

Example:

```bash
./splunk cmd walklex </path/to/tsidx_file.tsidx> "token"
```
Splunk Web and search problems

I can't find my data!

Are you searching for events and not finding them, or looking at a dashboard and seeing “No result data”? Here are a few common mistakes to check.

Are you running Splunk Free?

Splunk Free does not support multiple user accounts, distributed searching, or alerting.

Saved searches that were previously scheduled by other users are still available, and you can run them manually as required. You can also view, move, or modify them in Splunk Web or in savedsearches.conf.

Review this topic about object ownership and this topic about configuration file precedence in the Admin Manual for information about where Splunk writes knowledge objects such as scheduled searches.

Was the data added to a different index?

Some apps, like the *nix and Windows apps, write input data to a specific index (in the case of *nix and Windows, that is the "os" index). If you're not finding data that you're certain is in Splunk, be sure that you're looking at the right index. See Retrieving events from indexes in the Search Manual for more information. You might want to add the os index to the list of default indexes for the role you're using. For more information about roles, refer to Add and edit roles with Splunk Web in the Securing Splunk Enterprise manual. For information about troubleshooting data input issues, see Troubleshoot the input process in the Getting Data In manual.

Do your permissions allow you to see the data?

Your permissions can vary depending on the index privileges or search filters. See Add and edit roles in Splunk Web in Securing Splunk Enterprise for more information.

What about issues related to time?

Double check the time range that you're searching. Are you sure the events exist in that time window? Try increasing the time window for your search.

You can try a time picker value of All time for some part of your data, like a source type or string. This is one of the few ways to show events that have been erroneously timestamped with a future timestamp.

If you are running a report, check the time zone of the user who created the report.

The indexer might be incorrectly timestamping for some reason. See How timestamp assignment works in the Getting Data In manual.

Are you using forwarders?

Check that your data is in fact being forwarded. Here are some searches to get you started. You can run all these searches, except for the last one, from the Splunk default Search app. The last search you run from the CLI to access the
Are my forwarders connecting to my receiver? Which IP addresses are connecting to Splunk as inputs, and how many times is each IP logged in metrics.log?

```
index=_internal source=*metrics.log* tcpin_connections | stats count by sourceIp
```

• What output queues are set up?

```
index=_internal source=*metrics.log* group=queue tcpout | stats count by name
```

• What hosts (not forwarder/TCP inputs) have logged an event to Splunk in the last 10 minutes? (Including rangemap.)

```
| metadata type=hosts index=netops | eval diff=now()-recentTime | where diff < 600 | convert ctime(*Time) | stats count | rangemap field=count low=800-2000 elevated=100-799 high=50-99 severe=0-49
```

• Where is Splunk trying to forward data to? From the Splunk CLI issue the following command:

```
$SPLUNK_HOME/bin/splunk search 'index=_internal source=*metrics.log* destHost | dedup destHost'
```

• If you need to see if the socket is getting established you can look at the forwarder’s log of this in splunkd.log

"Connected to idx=<ip>:<port>" , and on the receiving side if you set the log category TcpInputConn to INFO or lower you can see messages "Connection in cooked mode from src=<ip>:<port>"

Read up on forwarding in the Forwarding Data Manual.

**Are you using search heads?**

Check that your search heads are searching the indexers that contain the data you’re looking for. Read about distributed search in the Distributed Search Manual.

**Are you still logged in and under your license usage?**

If you have a number of license violations within a range of days, you will be prevented from searching your data. See About license violations in the Admin Manual. The Splunk platform will continue to index your data, and no data will be lost. You can use the admin user to search the _internal index and troubleshoot the problem.

**Are you using a scheduled search?**

Your time range could be excluding the events. Search over all time to verify.

Are you sure the incoming data is indexed when you expect and not lagging? For example, indexing can lag for tens of minutes under certain conditions. If you run a scheduled search every 20 minutes, you might not see the most recent data yet. But if you run the same search 70 minutes later, the data will be there.

To identify a lag between the event’s timestamp and indexed time, manually run the scheduled search with the following added syntax:

```
| eval time=_time | eval itime=_indextime | eval lag=(itime - time)/60 | stats avg(lag), min(lag), max(lag) by index host sourcetype
```

55
See Event indexing delay in this manual.

Missing data can also result from a scheduler problem. See Configure the priority of scheduled reports in the Reporting Manual.

Other common problems with scheduled searches are searches getting rewritten, saved, run incorrectly, or run not as expected. Investigate scheduled searches in audit.log and the search's dispatch directory. See What Splunk logs about itself in this manual and Dispatch directory and search artifacts in the Search Manual.

Check your search query

- Are you using NOT, AND, or OR? Check your logic.
- Are you using views and drilldowns? Splunk Web might be rewriting the search incorrectly via the intentions functionality.
- Double check that you're using the correct index, source, sourcetype, and host.
- Are you correctly using escape characters when needed?
- Are your subsearches ordered correctly?
- Are your subsearches being passed the correct fields?

Are you extracting fields?

- Check your regex. One way to test regexes interactively is in Splunk using the rex command.
- Do you have privileges for extracting and sharing fields? Read about sharing fields in the Knowledge Manager Manual.
- Are your extractions applied for the correct source, sourcetype, and host?

Additional resources

Watch a video on troubleshooting missing forwarder data by a Splunk Support engineer.

Have questions? Visit Splunk Answers and see what questions and answers the Splunk community has.

If you get stuck at any point, contact Splunk Support. Don't forget to send a diag! Read about making a diag in this manual.

Too many search jobs

A real-time (all-time) scheduled search might spawn many search jobs in the search dispatch directory when alert throttling is not enabled. This might negatively affect search performance.

Symptom

Splunk Web displays a warning about too many search jobs in the dispatch directory.

Remedies

Make sure that alert throttling is configured for any real-time all-time scheduled searches. Configure throttling in Settings > Searches, reports, and alerts. See Throttle alerts in the Alerting Manual.
If alert throttling is configured and you still see this warning, make the alert expiration shorter than the default of 24 hours. For example, change "alert expiration time" from 24 hours to 1 hour (or less, if you need your alert triggered very frequently). See Additional alert configuration options in the Alerting Manual.

The Monitoring Console has a helpful view, Distributed search: Instance. The view provides details on search artifacts, including time to reap the dispatch directory.

**Dashboard in app is not showing the expected results**

You are using an app, and one of its views does not show you the results you expect. Begin troubleshooting here.

**Determine the search string that powers the panel that is not showing the expected results**

There are many methods to achieve this.

You can look at the view source by appending "?showsource=1" ("&showsource=1" if other parameters have already been appended) to the view URL in the browser address bar.

**Expand macros and event types**

Macros and event types are convenient knowledge objects, but unless you know exactly what they do, they can obscure the way a given search works. For that reason, it is often easier to expand them manually so that you know exactly what your search is doing.

You can see the contents of your entire search by using a keyboard shortcut, Command+Shift+E (Mac OSX) or Control+Shift+E (Linux or Windows) from the Search bar in the Search page. This opens a preview that displays the expanded search string, including all search macros and saved searches. For more info, see Expand your search in the Search Manual.

**Run the search manually from the time line, in the relevant app context**

Answer the question: Can you reproduce this manually, outside of the view it was reported in?

**Compare results against source events**

The next step is simple: Compare the results generated by the search and its multiple evals against the source events.

**Dig deeper**

In order to drill down to the source of the problem, pick one example. A good one if possible: A search that we know was run by an actual user.

Add the SID as a search term.

As discussed earlier, stats first(user) by search_id picks up the most recent value of the user field for a given search id.
Intermittent authentication timeouts on search peers

Splunk Web users can experience intermittent timeouts from search peers when there are more concurrent searches attempting to run than the search peers can respond to.

A group of search heads can schedule more concurrent searches than some peers are capable of handling with their CPU core count.

Symptoms

On the search head, you might see yellow banners in quick succession warning that a peer or peers are 'Down' due to Authentication Failed and/or Replication Status Failed. Typically this can happen a few times a day, but the banners appear and disappear seemingly randomly.

On the search head, splunkd.log will have messages like:

WARN DistributedPeerManager - Unable to distribute to peer named xxxx at uri http://xxxx:8089 because peer has status = "Authentication Failed".

WARN DistributedPeerManager - Unable to distribute to peer named xxxx:8089 at uri http://xxxx:8089 because peer has status = "Down".

WARN DistributedPeerManager - Unable to distribute to peer named xxxx at uri http://xxxx:8089 because replication was unsuccessful. replicationStatus Failed

These symptoms can appear with or without other Splunk features such as search head clustering and indexer clustering being enabled. The symptoms are more common in environments with two or more search heads.

Diagnosis

To diagnose this issue and proceed with its resolution, set up the monitoring console for your deployment. Next time the issue occurs, you can use the monitoring console to view performance data and validate the diagnosis. For monitoring console set up instructions, see Multi-instance deployment monitoring console setup steps.

1. Find an auth-token timeout to scope the time the issue occurred.

The authentication timeout is 10 seconds, so when the auth-tokens endpoint on the peer takes more than 10 seconds to respond, you'll see an auth or peer status banner on the search head.

To find an auth timeout on the peer named in the search head banner:

```
index=_internal source=*splunkd_access* splunk_server="search_peer_name" auth | timechart max(spent)
```

Or to find an auth timeout on any peer:

```
index=_internal source=*splunkd_access* auth spent > 10000 NOT streams | table splunk_server spent _time
```

2. Examine the load average just before the auth timeout and check for a dramatic increase.

Now that you've established the time frame in step 1, examine metrics.log's load average over the time frame to determine whether the load increased significantly just before the timeouts were triggered. Typically the total time frame is about 2 minutes.
To find the load average:

```
index=_internal source=*metrics.log* host="search_peer_name" group=thruput | timechart bins=200
max(load_average)
```

3. Examine the CPU and memory usage on the search peers.

Use the monitoring console Resource Usage dashboard (Settings > Monitoring Console > Resource Usage: Instance) to review the peak resource usage on the search peer during the time scoped above. Look at the Average CPU Usage panel. If you have too many concurrent searches, you will see that the peer uses more than the available percentage of CPU per core.

For example: A healthy 8 core box will show no more than 100% x 8 cores = 800% average CPU usage. In contrast, a box overtaxed with searches typically shows 1000% or more average CPU usage during the time frame where the timeouts appear.

For more information, see Resource usage dashboards in the Monitoring Splunk Enterprise manual.

**Remedies**

Examine the concurrent search load. There are typically searches that had dubious scheduling choices made are scoped in inefficient ways.

Use the monitoring console Search Activity dashboards (Settings > Monitoring Console > Search > Activity) to learn about the dispatched searches, the app they were triggered from, search concurrency details, and so on. For more information, see Search activity dashboards.

Once you’ve identified your pileup of concurrent searches, start work on this list of things to do. All of them are good practices.

- Most importantly, spread the searches out to reduce concurrency. For example: Use the advanced scheduling cron to balance the searches so that they don't all run on the same minute every hour. Read about scheduling reports and configuring scheduled report priority in the Reporting Manual.
- Limit real-time searches. For example, a scheduled real-time alert that looks back over the last 2 minutes that only triggers an email. Unless it triggers a script, this task should be configured as a scheduled search set to run 10 minutes in the past (to address potential source latency) over a 5 minutes window, combined with a cron offset. This offers the same effect without tying down a CPU core across all peers, all the time. Read more about expected performance and known limitations of real-time searches and reports in the Search Manual.
- Re-scope the search time for actual information needs. For example: Scheduled searches that run every 15 minutes over a 4 hour time frame are a waste of limited resources. Unless you have a very good reason why a search should look back an additional 3 hours and 45 minutes on every search (such as extreme forwarder latency), it's a waste of shared resources. Read more about alerts in the Alerting Manual.
- Additionally, there’s the option to use limits.conf to lower the search concurrency of all the search heads. Note that if you do only this step, you will get a different set of banners (about reaching the max number of concurrent searches) and you will still not be able to run concurrent searches. But if you do some of the other steps, too, you might want to configure the search concurrency like this:

```
[search]
based_max_searches = 2 # Defaults to 6
max_searches_per_cpu = 1 # Defaults to 1
```
max_rt_search_multiplier = 1
# Defaults to 1 in 6.0, in 5.x defaults to 3

[scheduler]
max_searches_perc = 20
# Defaults to 50
auto_summary_perc = 10
# Defaults to 50

- Add more search peers.
- Add more cores to the search peers.
- As a last resort, there's also the option of increasing the distsearch.conf timeouts. This is a workaround, and will slow down search results during peak load times. Increase the timeouts in distsearch.conf on the search head:

[distributedSearch]
statusTimeout = 30
# Defaults to 10
authTokenConnectionTimeout = 30
# Default is 5
authTokenSendTimeout = 60
# Default is 10
authTokenReceiveTimeout = 60
# Default is 10

Performance degraded in a search head pooling environment

Symptoms

In a pool environment, you notice that searches are taking longer than they used to. How do you figure out where your performance degradation is coming from? This topic suggests a few tests you can run.

Time some simple commands

Try some basic commands outside of Splunk Enterprise. If either of these operating system commands takes more than ten or so seconds to complete, it indicates an issue on the shared storage.

- On the search head, in the pooled location, at the *nix command line,

```
time find /path/to/pool/dir | wc -l
```

measures the time to find the things in .../dir and then count them.

- Another simple command to try is:

```
time ls -lR /path/to/pool/dir | wc -l,
```

which measures how long it takes to count items in the pool.

If you do not have shell access, other tests you can run include:

- logging in (which uses a shared token)
- accessing knowledge objects.
Compare searches in and out of search head pooling

Run a simple search (for example, `index=_internal source=*splunkd.log | tail 20`) with and without search head pooling enabled. Compare the timings.

UseSplunk Enterprise log files

In `splunkd.log` searchstats

```
index=_internal source=*splunkd_access.log NOT rtsearch spent>29999
```

any search taking over 30 seconds to return is a slow search.

If

- the only slow things are searches (but not, for example, bundle replication), then your problem might be with your mount point. Run some commands outside of Splunk Enterprise to validate that your mount point is healthy.
- accessing knowledge objects takes a long time, search in `metrics.log` for the load_average:

```
index=_internal source=*metrics.log load_average
```

look in metrics for 2-5 minutes before and after the duration of the slow-running search

If you see this is high, and you have SoS installed, refer to the same period of time and look at the CPU graphs on SoS to make sure you’re not seeing a system load.

If the problem is with the mount point, the box is not going to be challenged.

If the problem is with the search load, the CPU usage will be high for the duration of the slow search.

Is it a search load problem?

Start turning off field extractions. Is it still slow?

Next turn off real-time all-time and wildcards in your searches.

If you have the Splunk on Splunk app, check the search load view. If you have the Distributed Management Console, check the Search Activity views.

Consider search scheduling. Have you scheduled many searches to run at the same time? Use the Distributed Management Console Search Activity view to identify search scheduling issues. If you’ve identified issues, move some of your scheduled searches to different minutes past the hour.

I'm having problems with the Splunk PDF Server app

This topic is about troubleshooting the PDF Server for Linux app. This app enabled a Linux-based Splunk instance to generate emailed reports in PDF format. The PDF Report Server App was deprecated in Splunk Enterprise version 6.0. This feature was removed from Splunk Enterprise in version 6.2.

In version 6.2 you cannot generate PDFs from dashboards or forms that were built using advanced XML.
Splunk Enterprise continues to support integrated PDF generation, as described in Generate PDFs of your reports and dashboards in the Reporting Manual.

**SuSE Linux search error**

Users running Splunk software on a SuSE server might see the following error message when executing a search:

Unable to get a properly formatted response from the server; canceling the current search

Alternatively, the dashboard might not display properly.

To resolve this issue:

1. Open the `/etc/mime.types` file for editing.
2. Delete (or comment out) these two lines:

   ```
   text/x-xsl xsl
   text/x-xslt xslt xsl
   ```

3. Change the line with `text/xml xml` to `text/xml xml xsl`.
4. With these changes in place, restart the Splunk platform and clear your browser cache.

**Note:** If you are using a proxy, you need to flush that as well.
Data acquisition problems

Identify and triage indexing performance problems

Use this topic to begin to troubleshoot indexing performance problems in Splunk software.

Problems

The following are some symptoms of indexing performance issues:

- Messages in Splunk Web indicate data stalls on indexers or on instances sending data to indexers.
- Forwarders are unable to send data to indexers.
- Receiving ports on indexers are being closed. You might have learned of this by looking at the Splunk TCP Input: Instance Monitoring Console dashboard.
- Event-processing queues are saturated. You might have learned of this from the Monitoring Console Health Check or from a Monitoring Console platform alert.
- Indexing rate is unusually low. You might have learned of this from the Monitoring Console Health Check.
- Data is arriving late. See Event indexing delay.

Gather information

You need three pieces of information to begin to diagnose indexing problems: indexing status, indexing rate, and queue fill pattern.

Before you continue, consider reading How indexing works in Managing Indexers and Clusters of Indexers.

Determine indexing status

The indexer processor can be in one of several states: normal, saturated, throttled, or blocked.

View the current state using any of these methods:

- Monitoring Console health check
- Monitoring Console indexing performance views
- "Saturated event-processing queues" platform alert, included with the Monitoring Console
- server/introspection/indexer endpoint

See About the Monitoring Console in Monitoring Splunk Enterprise.

Determine and categorize the indexing rate

Distinguish between an indexing rate that is nonexistent (0 MB/s), low (1 MB/s), or high (at least several MB/s).

Use the Monitoring Console to determine indexing rate. This information is available in several locations within the Monitoring Console:

- Overview > Topology with the Indexing rate overlay.
- Indexing > Indexing Performance: Deployment.
• Indexing > Indexing Performance: Instance.

Determine queue fill pattern

Indexing queue fill profiles can be grouped into three basic shapes. For this diagnosis, differentiate between flat and low, spiky, and saturated.

Indexing is not necessarily "blocked" even if the event-processing queues are saturated.

Use the Monitoring Console to determine the queue fill pattern. See Monitoring Console > Indexing > Indexing Performance: Deployment and also Monitoring Console > Indexing > Indexing Performance: Instance.

Here is an example of a flat and low queue fill pattern.
Here is an example of a spiky but healthy queue fill pattern. Although some of the queues saturate briefly, they recover quickly and completely.

Here is an example of a saturated queue fill pattern.
Causes and solutions

Once you have the three pieces of information from Gather information, use the following table to diagnose your system.

<table>
<thead>
<tr>
<th>Indexing status</th>
<th>Indexing rate</th>
<th>Queue fill pattern</th>
<th>Diagnosis</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>Either low or high</td>
<td>Flat or spiky</td>
<td>Your indexer is running normally.</td>
<td>None required.</td>
</tr>
<tr>
<td>Normal</td>
<td>High</td>
<td>Saturated</td>
<td>Indexers are at or near capacity.</td>
<td>If this behavior persists through all cycles of system usage, you can configure more indexers, parallel pipelines, or high performance storage. See Dimensions of a Splunk Enterprise deployment in the Capacity Planning Manual.</td>
</tr>
</tbody>
</table>
| Normal          | Low           | Saturated          | Possibilities include:  
- Insufficient provisioning.  
- A configuration error (very large events or a bad transform).  
- Search competing with your system I/O or CPU.  
- A Splunk software defect. | Respectively:  
- Store hot buckets on better hardware and/or get more CPU power.  
- To find a bad transform, run the Monitoring Console Health Check, click Event processing issues, and follow the drilldown.  
- For a suspected Splunk software defect, file a Support case. See Known issues in the Release Notes. |
| Blocked or throttled | Nonexistent (zero) | Saturated | Possibilities include:  
- Disk full.  
- Indexer is forwarding to another indexer or a third-party system that is not accepting data.  
- Hardware failure.  
- Out of memory. | Respectively:  
- Manage disk space.  
- Check forwarding configuration.  
- Ensure the indexer can write to disk.  
- Ensure the indexer has enough memory.  
- Contact Support. |

Event indexing delay

Symptoms

Events collected from a forwarder or from a log file are not yet searchable on Splunk. Even though the time stamps of the events are within the search time range, a search does not return the events. Later, a search over the same time range returns the events.

Diagnosis

Quantify the problem by measuring how long your Splunk deployment is taking to make your data searchable.

To measure the delay between the time stamp of the events and the indexing time (the time that the indexer receives and processes the events), use the following method:

1. Look at the delay in seconds per host.

source=mysource | eval delay_sec=_indextime_-_time | timechart min(delay_sec) avg(delay_sec) max(delay_sec) by host
2. Look at the delay in seconds per source for a particular host.

```
source=mysource host=myhost | eval delay_sec=_indextime-_time | timechart min(delay_sec) avg(delay_sec) max(delay_sec) by source
```

3. Look at the delay per host for the Splunk internal logs.

```
index=_internal source=*splunkd.log* | eval delay_sec=_indextime-_time | timechart min(delay_sec) avg(delay_sec) max(delay_sec) by host
```

Run these searches on `realtime - all time` mode for a little while to see the events that are being received right now. In addition to real-time, you can run `historical` searches to compare a day this week to a day from a previous week.

Compare the delay between hosts and forwarders.

- Determine the common denominator between them. For example, all of the delayed events might be from the same log file or the same host or source type.

Compare the delay from your events with the delay from the internal Splunk logs.

- If all the logs are delayed, including the internal logs, then the delay is a forwarding issue.
- If some sources are delayed but not others, this indicates a problem with the input.

As you implement each fix below, you can measure how well it's working by running these searches again.

**Root causes**

There are several possible root causes. Some might not be applicable to your situation.

**Possible throughput limits**

Universal and lightweight forwarders have a **default throughput limit of 256Kbps**. This default can be configured in `limits.conf`. The default value is correct for a forwarder with a low profile, indexing up to ~920 Mb/hour. But in the case of higher indexing volumes, or when the forwarder has to collect the historical logs after the first start, the default might be too low. This could delay the recent events.

To check the forwarder default throughput limit, on the command line in the `splunk` folder type:

```
   cd $SPLUNK_HOME/bin
   ./splunk cmd btool limits list thruput --debug
```

**Example of result on a universal forwarder. The default limit is 256KBps, and is set up in the app:**

```
   /opt/splunkforwarder/etc/apps/SplunkUniversalForwarder/default/limits.conf [thruput]
   /opt/splunkforwarder/etc/apps/SplunkUniversalForwarder/default/limits.conf maxKBps = 256
```

**Example of result on an indexer or heavy forwarder. The default is unlimited:**

```
   /opt/splunk/etc/system/default/limits.conf [thruput]
   /opt/splunk/etc/system/default/limits.conf maxKBps = 0
```
To verify in the forwarder: When the thruput limit is reached, monitoring pauses and the following events are recorded in splunkd.log:

```plaintext
INFO TailingProcessor - Could not send data to output queue (parsingQueue), retrying...
```

To verify how often the forwarder is hitting this limit, check the forwarder's metrics.log. (Look for this on the forwarder because metrics.log is not forwarded by default on universal and light forwarders.)

```bash
cd $SPLUNK_HOME/var/log/splunk
grep "name=thruput" metrics.log
```

Example: The instantaneous_kbps and average_kbps are always under 256KBps.

```
11-19-2013 07:36:01.398 -0600 INFO  Metrics - group=thruput, name=thruput, instantaneous_kbps=251.790673, instantaneous_eps=3.934229, average_kbps=110.691774, total_k_processed=101429722, kb=7808.000000, ev=122
```

**Remedy**

Create a custom limits.conf with a higher limit or no limit. The configuration can be in system/local, or in an app that will have precedence on the existing limit.

Example: Configure in a dedicated app, in /opt/splunkforwarder/etc/apps/Gofaster/local/limits.conf

Double the thruput limit, from 256 to 512 KBps:

```
[thruput]
maxKBps = 512
```

Or for unlimited thruput:

```
[thruput]
maxKBps = 0
```

**Notes:**

- Unlimited speed can cause higher resource usage on the forwarder. Keep a limit if you need to control the monitoring and network usage.
- Restart to apply.
- Verify the result of the configuration with btool.
- Later, verify in metrics.log that the forwarder is not reaching the new limit constantly.

**Possible network limits**

Once the thruput limit is removed, if the events are still slow, use the metrics method to check if the forwarders are hitting a network limit. Compare with other forwarders on different networks or different VPN tunnels.

**Monitoring archived logs file**

Compressed files (like .gz and .zip) are handled by the Archive processor, and are serialized. Therefore if you index a large set of compressed files, they will come through the indexer one after the other. The second file will only come through after the first one has been indexed.
Example: splunk ArchiveProcessor starting to read file in splunkd.log
12-12-2013 00:18:07.388 +0000 INFO ArchiveProcessor - handling file=/var/log/application/rsyslog.log.1.gz

Remedy

An available workaround is to have Splunk to monitor the uncompressed files.

Possible timestamp/timezone issue

Using a real time-alltime search, if an event is visible immediately in real time, but not otherwise (that is, with an historical search), it might be that the time stamp of the event is in the future. Historical searches (even all-time) show events with timestamps only within the window of the search.

Use this search to verify the source type, the time stamp detected (_time), the time of the user on the search head (now), and the time zone applied (date_zone).

```
source=mysource host=myhost | eval delay_sec=_indextime-_time | convert ctime(_indextime) AS indextime | eval now=now() | table _time indextime now date_zone source sourcetype host
```

Notes:

- The _time is converted to the user profile time zone configured on the search head at search time.
- The date_zone is applied at index time on the indexer.

Remedy

Fix the time zone and time stamp extraction. Take a sample of the data and test it with data preview.

Windows event logs delay

If the only events delayed are WinEventLogs, and the forwarder is on a busy domain controller, with a high number of events per second, you might be encountering the Windows collection log performance limit on Splunk and 5.x.

Or if the forwarder was recently started, it might be still collecting the older events first.

Remedy

- Use the Splunk Enterprise 6.0 Windows forwarder, with WinEventLog collection switched to improved modular inputs. See “Welcome to Splunk Enterprise 6.0” in the release notes.
- If possible, collect only recent logs (current_only=1), or monitor the recent events first (start_from=newest). See Monitor Windows event log data in Getting Data In.
- Reduce the volume of events to collect. Use the whitelist/blacklist filters in the WinEventLog stanza to exclude particular EventCodes. See inputs.conf.spec.

Garbled events
Symptom

Events in Splunk look strange or display as foreign characters or files.

Explanation

Many files are human readable that are not in a properly encoded format. Many applications will auto-trim text or special characters including nulls, so it is important to know what is included in the log file, not just what the application displays.

Solution

To correct this, set the charset in props.conf for this input to the appropriate character set (using the CHARSET attribute).

If you don't know the encoding of your source file, and have access to a *nix machine, you can use the “file” command:

```bash
file sample.log
sample.log: UTF-8 Unicode English text
```

In this example, the encoding is UTF-8. Note, though, that Splunk accepts many other encodings. Find a list of supported character sets, and instructions on specifying a charset, in "Configure character set encoding" in the Getting Data In Manual.

Binary file error

Symptom

My file will not index. The splunkd.log on my forwarder has an error saying my file might be binary.

Explanation

Sometimes non-UTF-8 logs are not processed because they are seen as binary in the binary check process.

Solution

Set the charset in props.conf for this input to the appropriate charset. This error shows up in the splunkd.log where the props.conf needs to be specified. So, if you are using a forwarder, the forwarder's splunkd.log is where you will find the error, and that is also where you need to configure the props.conf.

What do I do with buckets?

Buckets are portions of Splunk indexes. This article points you to a few resources for troubleshooting problems with buckets.

Might I be having issues with bucket rotation?

An unsuitable bucket rotation and retention policy can lead to:
old events being deleted before they reach frozen buckets,
- hot and warm buckets filling up, stopping Splunk,
- old events not being archived correctly and thus still searchable when they shouldn't be, and
- poor searching or indexing performance.

Here's a Community Wiki article about bucket rotation and retention with specific recommendations and examples.

**Recover metadata for a corrupt Splunk index directory**

Contact Splunk Support for direction before using this command.

The *recover-metadata* command recovers missing or corrupt metadata associated with any Splunk index directory, sometimes also referred to as a bucket.

If your Splunk instance will not start, a possible cause is that one or more of your index buckets is corrupt in some way. Contact Support; they will help you determine if this is indeed the case and if so, which bucket(s) are affected. Then, run this command:

```bash
$SPLUNK_HOME/bin/splunk cmd recover-metadata <full path to the exact index directory/bucket>
```

Splunk returns a success or failure message.

**Recovering and rebuilding buckets**

*Managing Indexers and Clusters* has a thorough explanation of buckets. See the Troubleshoot indexers and clusters of indexers chapter for help troubleshooting bucket problems, like crash recovery, rebuilding buckets, bucket replication issues, and configuration bundle issues.
Windows-specific problems

Troubleshoot Windows event log collection

This topic discusses solutions to problems encountered when attempting to get Windows event log data into Splunk.

Problems with collection and indexing of Windows event logs generally fall into two categories:

- **Event logs are not collected from the server.** This is usually due to either a local configuration problem or, in the case of remote event log collection, a network, permissions, or authentication issue.
- **Event logs are collected from the server, but information within the event log is either missing or incorrect.** This is usually due to problems associated with a particular event log channel, or because of the methods used to collect data from those channels.

Troubleshooting issues with event logs collected locally

When you have problems getting data into your local Splunk instance, try these tips to fix the problem:

- Make sure that the desired event log channels are selected in Splunk Web or properly configured in inputs.conf.
- Make sure to select fewer than 64 event log channels per event log input.
- Make sure that you are not attempting to index exported event logs that are incompatible with the indexing system (for example, attempting to index event logs exported from a Windows Server 2008 computer on a Windows XP computer will result in missing log data).
- Make sure that, if you are monitoring non-standard event log channels, that you have the appropriate dynamic linked libraries (DLLs) that are associated with that event log channel. This is particularly important when indexing exported log files from a different computer.

Troubleshooting issues with event logs collected remotely

When you experience issues getting event logs from remote Windows servers, try these solutions to fix the problem:

- Make sure that your Splunk user is configured correctly for WMI.
- Make sure that your Splunk user is valid, and does not have an expired password.
- Make sure that the Event Log service is running on both the source and target machines.
- Make sure that your Active Directory (AD) is functioning correctly.
- Make sure that your computers are configured to allow WMI data between them.
- Make sure that your event logs are properly configured for remote access.

For more information

See the Admin Manual for information on getting started for Windows admins.

Common issues with Splunk and WMI

This topic discusses common issues encountered when getting WMI-based data into Splunk. It offers solutions for problems such as the following:
• Splunk can't get data from remote machines.
• Splunk can't get local data through WMI.
• Splunk sometimes crashes when getting remote data.
• Splunk connects to WMI differently depending on product version.

Splunk can't get data from remote machines

When Splunk can index events on the local machine, but can't get data from remote machines using WMI, authentication or network connectivity is often the reason. Splunk requires a user account with valid credentials for the Active Directory (AD) domain or forest in which it's installed in order to collect data remotely. It also requires a clear network path to the machine from which it gets data, unblocked by firewalls on either the source or target machines.

Determine that Splunk has been installed as a domain user

The first thing to do is to make sure that Splunk is installed as a domain user. If this requirement isn't met, Splunk won't be able to get data remotely even if the network is functioning.

1. Open a command prompt.
2. Run the `sc` command to query the Services Command Manager about the `splunkd` and `splunkweb` services.

```
C:\> sc qc splunkd
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: splunkd
  TYPE : 10  WIN32_OWN_PROCESS
  START_TYPE : 2  AUTO_START
  ERROR_CONTROL : 1  NORMAL
  BINARY_PATH_NAME : "C:\Program Files\Splunk\bin\splunkd.exe" service
  LOAD_ORDER_GROUP :
  TAG : 0
  DISPLAY_NAME : Splunkd
  SERVICE_START_NAME : LocalSystem
```

The `SERVICE_START_NAME` field tells you the user that Splunk is configured to run as. If this field shows `LocalSystem`, then Splunk is not configured to run as a domain user. Uninstall Splunk, then reinstall it and make sure to specify "Other user" during the setup process.

Note: You can also determine which user Splunk is configured to run as by using the Services control panel.

Review the splunkd.log file

If Splunk is correctly configured as a domain user, the next step is to investigate why Splunk is having problems connecting to WMI providers.

Open the `%SPLUNK_HOME%\var\log\splunk\splunkd.log` file and search for `wmi`.

When Splunk encounters an error attempting to connect to a WMI provider, it logs errors in `splunkd.log` as follows:

```
03-11-2009 10:08:29.296 ERROR ExecProcessor - error from "python E:\Splunk\bin\scripts\splunk-wmi.py" ERROR WMI - Instantiation of IWbemServices::ExecQueryAsync failed (error code 800706be)
```
The following table shows the most common errors encountered when connecting to WMI providers:

<table>
<thead>
<tr>
<th>Error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>80070005</td>
<td>Access is denied. (due to an incorrect login)</td>
</tr>
<tr>
<td>80041064</td>
<td>User credentials cannot be used for local connections.</td>
</tr>
<tr>
<td>800706BA</td>
<td>The RPC server is unavailable.</td>
</tr>
<tr>
<td>80041003</td>
<td>Access Denied. (due to explicit access restrictions)</td>
</tr>
</tbody>
</table>

If you see lines within the log file that contain `HRESULT error` then Splunk is unable to complete the WMI operation due to a network connectivity or authentication problem. You can use the `WBEMTEST` utility to corroborate what is shown in Splunk's log file.

**Enable debug logging**

You can get even more detailed information about what is causing the errors by enabling debug logging in Splunk's logging engine.

**Note:** After you have confirmed the cause of the error, be sure to turn debug logging off.

To enable debugging for WMI-based inputs, you must set two parameters:

1. **Edit** `log.cfg` in `%SPLUNK_HOME\etc`. Add the following parameter:

   ```
   [splunkd]
   category.ExecProcessor=DEBUG
   ```

2. **Edit** `log-cmdline.cfg`, also in `%SPLUNK_HOME\etc`. Add the following parameter:

   ```
   category.WMI=DEBUG
   ```

   **Note:** You can place this attribute/value pair anywhere in the file, as long as it is on its own line. `log-cmdline.cfg` does not use stanzas.

3. **Restart Splunk:**

   C:\Program Files\Splunk\bin> splunk restart

4. **Once Splunk has restarted,** let it run for a few minutes until you see debug log events coming into Splunk.

   **Note:** You can search Splunk's logfiles within Splunk by supplying `index=\_internal` as part of your search string. Review "What Splunk logs about itself" in the Troubleshooting Manual for additional information.

5. **Once Splunk has collected enough debug log data,** send a diag to Splunk Support:
C:\Program Files\Splunk\bin> splunk diag

After you finish troubleshooting, revert to the default settings:

1. In log.cfg, change the category.ExecProcessor attribute to its default setting:

    [splunkd]
category.ExecProcessor=WARN

    **Note:** You can also remove this entry from the file.

2. In log-cmdline.cfg, change the category.WMI attribute to its default setting:

    category.WMI=ERROR

    **Note:** Any changes made to log.cfg are overwritten when you upgrade Splunk. Create a log-local.cfg in %SPLUNK_HOME%\etc to avoid this problem.

*Use the WBEMTEST utility to reproduce the error outside of Splunk*

If you see HRESULT error entries in the splunkd.log, use the WBEMTEST utility to confirm the error outside of Splunk.

1. Log into the Splunk server as the Splunk user.

2. Click Start > Run?

3. In the Run dialog, type in wbemtest and click OK.

4. In the Windows Management Instrumentation Tester window, click the Connect? button.

    The Connect window appears.

5. In the Namespace field of the Connect window, type in the namespace of the server that is experiencing errors.

    **Note:** You must type in the full path of the namespace. For example, if the server you are attempting to connect to is called ADLDBS01, you must type in \\ADLDBS01\root\cimv2 (including the backslashes).

6. Click Connect.
Note: You should be able to connect to the server without needing to supply credentials. If you are prompted for credentials, then the Splunk user is not correctly configured to access WMI.

7. Once you are connected to the server, set your WMI connection mode by selecting one of the radio buttons in Method Invocation Options the lower right corner of the WBEMTEST window:

- For Splunk 3.4.9 and earlier, choose Asynchronous.
- For versions of Splunk after 3.4.9, choose Semisynchronous.

8. Click “Query?”

The Query window appears.

9. In the Query window, type in a valid Windows Query Language (WQL) statement, such as the one supplied below, then click Apply.

Following is a WQL statement that you can test WMI connections with:

```
SELECT Category, CategoryString, ComputerName, EventCode, EventIdentifier, EventType, Logfile, Message, RecordNumber, SourceName, TimeGenerated, TimeWritten, Type, User FROM Win32_NTLogEvent WHERE Logfile = "Application"
```
The following graphic shows an example of successful results:

**Check Windows Firewall**

If Windows Firewall (or any other firewall software) is running on either the source or target machine, Splunk might be blocked from getting data through WMI providers. Make sure that you explicitly allow WMI through on the firewalls on both machines. You can also disable Windows Firewall, but this is not recommended by Splunk or Microsoft.


**Splunk is unable to get local data through WMI**

When Splunk is unable to get data from the local machine through WMI providers, this might be because WMI is experiencing issues under load. When this happens, try restarting the Windows Management Instrumentation (wmimgmt) service from within the Services control panel, or by using the sc command-line utility.

**Splunk sometimes crashes when collecting data over WMI**

WMI can occasionally cause the splunk-wmi.exe process to crash. Splunk will spawn a new process when this happens (you can tell by the changed process ID).

- While there is no guaranteed fix for this issue, you can reduce the number of crashes by reducing the number of servers you are monitoring through WMI with any given Splunk instance. Limit the number of WMI-based inputs per instance to 80 or fewer.

- If you monitor the same subset of WMI providers on large numbers of machines, you can run into WMI memory constraints on the monitoring server. This can also cause crashes. Limit the number of WMI-based data inputs per server monitored through WMI. It's best to reduce the total number of WMI connections per instance to 120 or fewer on 32-bit Windows servers, and 240 or fewer on 64-bit Windows servers.

- Consider using universal forwarders to get your data. You can either install universal forwarders on a few machines and get data from other machines through WMI, or you can put universal forwarders on all remote machines.
**Splunk connects to WMI differently based on product version**

When Splunk makes requests to WMI, it does so in one of three ways: Synchronous, asynchronous and semisynchronous.

Splunk makes what are known as *semisynchronous* calls to WMI providers. This means that when Splunk makes a call to WMI, it continues running while WMI deals with the request.

Semisynchronous mode offers the best balance of resource usage and security on the computer making the request. It differs from the faster *asynchronous* mode, but is more secure due to the way that the system handles retrieval of the WMI objects. Both of these modes are faster than *synchronous* mode, which forces programs making that kind of WMI request to wait until WMI returns the data.

When WMI is dealing with a large number of requests, you might notice a slower response because memory usage on the system increases until the retrieved WMI objects are no longer needed by Splunk (after indexing).

More information about how WMI calls are made is available at "Calling a Method", http://msdn.microsoft.com/en-us/library/aa384832(VS.85).aspx on MSDN.

**Note:** Versions of Splunk prior to 3.4.10 make asynchronous connections to WMI providers.

**Manually verify that WMI is working**

To test WMI, you can run the `splunk-wmi.exe` command manually with a desired query and/or namespace to see the output that it produces.

**Caution:** When running this command, be sure to temporarily change Splunk’s data store directory (the location that `SPLUNK_DB` points to), so that you do not miss any WMI events. To change Splunk’s database store, refer to “Test access to WMI providers” in the Getting Data In Manual.

Here is an example of a valid `splunk-wmi` statement:

```
C:\Program Files\Splunk\bin> splunk cmd splunk-wmi.exe -wql "select * FROM Win32_PerfFormattedData_PerfDisk_PhysicalDisk"
```

The following output shows a failure to connect to the desired WMI provider:

```
$ ./splunk cmd splunk-wmi.exe -wql "select * FROM Win32_PerfFormattedData_PerfDisk_PhysicalDisk_typo"
ERROR WMI - Error occurred while trying to retrieve results from a WMI query (error="Specified class is not valid." HRESULT=80041010) (.: select * FROM Win32_PerfFormattedData_PerfDisk_PhysicalDisk_typo)
ERROR WMI - Giving up attempt to connect to WMI provider after maximum number of retries at maximum backoff time (.: select * FROM Win32_PerfFormattedData_PerfDisk_PhysicalDisk_typo)
Clean shutdown completed.
```

The following shows a successful connection to a WMI provider:

```
jrodman@jrodman-PC /cygdrive/c/Program Files/Splunk/bin
$ ./splunk cmd splunk-wmi.exe -wql "select * FROM Win32_PerfFormattedData_PerfDisk_PhysicalDisk"
20090904144105.000000
80
```

---

78
For more information

See the Admin Manual for information on getting started for Windows admins.

Advanced help troubleshooting Splunk software for Windows

Review this topic if you're having trouble getting data into your Splunk for Windows instance, or if Splunk is having problems starting or running.

This topic provides solutions to common issues encountered when working with the Windows version of Splunk. It's divided into several subtopics:

• Generic issues
• Issues with WMI
• Issues with forwarders

General issues

This section contains solutions to common issues encountered when running Splunk on Windows.

Splunk fails to start

There are several factors that might prevent Splunk from starting properly. Whether it didn't start automatically, or you are having problems manually starting it, here are some solutions to try:

• Make sure that your system meets the Splunk system requirements. These requirements differ depending on the type of Splunk you're trying to run (full instance versus forwarder).

• Make sure that the Splunk services are enabled. Go into Control Panel and check that the splunkd and splunkweb services have their Startup type set to "Automatic."

• Check file and security permissions. When you install Splunk as a user other than Local System, Splunk does not have full permissions to run on the system by default. Try these solutions to get Splunk back up and running:
  ♦ Make sure the Splunk user is in the local Administrators group on the machine.
  ♦ Make sure that the Splunk user has Full Control permissions for the entire %SPLUNK_HOME% directory, and is also the owner of all files and subdirectories in %SPLUNK_HOME%. You must explicitly define this in the Security properties of the %SPLUNK_HOME% directory.
  ♦ Be sure to read the "Considerations for deciding how to monitor remote Windows data" for additional information about permissions required to run Splunk as a domain user.

• Make sure Splunk isn't crashing. Check your %SPLUNK_HOME%/var/log/splunk directory. If Splunk crashes on startup, you'll see one or more dump files located in this directory. If Splunk is crashing, try the solutions listed above first. If those don't work, then uninstall and reinstall the program. If you still encounter problems, contact
Splunk Support or visit the Answers page for additional guidance.

Note: Splunk Support requires an enterprise license in most cases.

No data is received

Splunk for Windows operates similarly to Splunk for other operating systems. If you’re not getting data and it’s not because of a permissions or network connectivity issue, then there is likely something happening within Splunk, such as an incorrectly configured input.

If you’re having trouble collecting Windows event logs, review "Troubleshooting Windows event logs."

Try these solutions to figure out where your data is going:

• Make sure the clocks on all machines in your network are synchronized. If your Active Directory is set up correctly, this should already be done for you. Make sure the W32Time services on all your machines are running and properly syncing time with the appropriate domain controller.
• Make sure your inputs are correctly configured. In particular, the performance monitoring, Registry monitoring and Active Directory monitoring inputs must be properly configured if you want them to return data. When collecting performance logs remotely over WMI, for example, use Splunk Web instead of configuration files to create those inputs, as typos in configuration files will prevent Splunk from collecting the data you want.
• Make sure the type of input you are attempting to configure actually returns data. Some performance counters and Registry keys do not change throughout the course of a Windows session, for example. If there is no change, there is no data to collect.
• When using the Search app, make sure the time range for your search is correct. If you’re searching for events that are outside the time range shown in the Search app, they won’t appear there. Adjust the Search app’s time range if needed.
• Make sure that the indexing or parsing pipelines on your indexer or forwarder are not blocked. For example, if Splunk can’t send events from a forwarder to an indexer, due to a network issue, it may appear as though Splunk is not indexing the data, when data has not actually arrived. Check %SPLUNK_HOME%/var/log/splunk/metrics.log for information about the status of Splunk’s processing queue. For more information on metrics.log, consult "Work with metrics.log" in this manual.

WMI issues

This section contains information about problems encountered when using WMI providers to gather data from remote machines.

No WMI-based events come into Splunk

When Splunk is unable to index WMI-based events, it is likely because of a permissions or security issue. Be sure to review the permissions checklist located in "Monitor WMI-based data" in the Getting Data In Manual. A summary of that checklist follows:

• Splunk must run as a user that is a member of the local Administrators group on the server doing the indexing.
• The Splunk user must be a member of the domain groups required to access the appropriate WMI resources.
• The Splunk user must be configured with specific local and domain security policy rights.
• WMI security must be correctly configured.
• Windows Firewall must be correctly configured.
• User Access Control must be considered.

81
You can also see additional information about Splunk's WMI operations by turning on debug logging. To turn on debug logging, follow the instructions in "Troubleshooting WMI Logging" in the Getting Data In Manual.

**WMI-based events come in, but sometimes Splunk crashes**

WMI can sometimes cause the Splunk WMI process (`splunk-wmi.exe`) to crash. If that happens, Splunk will start another WMI process immediately, but you might see crash files in your `%SPLUNK_HOME%\var\log\splunk` directory.

If Splunk is crashing, try the following solutions:

- **Reduce the amount of WMI inputs on each Splunk instance.** For best results, limit the number of WMI connections per instance to 120 or fewer on 32-bit Windows systems, or 240 or fewer for 64-bit systems. Note that each server monitored can use more than one WMI connection, depending on the amount of inputs configured for each server.
- **Use a universal forwarder to get data.** Splunk recommends that you use a universal forwarder to send data from remote machines to an indexer. Universal forwarders are more scalable and reliable than WMI in nearly all cases, and require far less security management than WMI does.

**Splunk's WMI process runs slowly**

Splunk makes what are known as *semisynchronous* calls to WMI providers. This means that when Splunk makes a call to WMI, it continues running while WMI deals with the request.

Semisynchronous mode offers the best balance of resource usage and security. It differs from the faster *asynchronous* mode, but is more secure due to the way that the system handles retrieval of the WMI objects. Both of these modes are faster than *synchronous* mode, which forces programs making that kind of WMI request to wait until WMI returns the data.

When WMI is dealing with a large number of requests, you might notice a slower response because memory usage on the system increases until the retrieved WMI objects are no longer needed by Splunk (after indexing).

**More help**

If you are still having issues, read "Troubleshooting common issues with Splunk and WMI".

**Forwarder Issues**

This section provides help for users who use Splunk's forwarding and receiving capabilities, including the new universal forwarder included with Version 4.2 and later.

**Forwarder doesn't send any data**

If you're using a forwarder to send data to a receiver and the receiver isn't getting any data, there are a number of things you can try to fix the problem:

- **Make sure that there is network connectivity between the forwarder and the receiver.**
  - On the machine running the forwarder, open a command prompt and telnet to the IP address and port of the receiver. For example, if your Splunk receiver is configured at IP address 192.168.1.10 port 9997, you would type:
    
    ```
    > telnet 192.168.1.10 9997
    ```
  - Check the Windows Firewall on both the forwarder and the receiver. Windows Firewall must be configured to allow access in both directions for WMI. Either open ports to allow traffic, or disable Windows Firewall.
Note: On versions of Windows later than Windows XP or Windows Server 2003, the **telnet** client might not be installed. While the **telnet** client is not required for forwarding, you will not be able to use it to determine basic IP connectivity if it isn't installed. Follow the instructions located at "Install Telnet Client" (http://technet.microsoft.com/en-us/library/cc771275(WS.10).aspx) on MS TechNet to install the **telnet** client.

- **Make sure the configuration files on your forwarder are properly formatted.**
  - Review your configuration files carefully, and check for spelling and syntax errors.
  - Stanza names must always be bracketed with square brackets ([ ]). Don't use curly braces or parentheses.

- **If the universal forwarder is running as a user other than Local System, confirm that security and access control are correctly configured.**
  - Ensure that the Splunk user is a local Administrator on the machine.
  - Ensure that the Splunk user is valid (for example, it is neither locked out of the domain nor expired). Check that the user's password is also valid (not expired).
  - Ensure that the Splunk user has access to the desired resource(s).
  - Remember that special permissions are required to access some resources, such as the Security event log. Additionally, changing permissions for these resources sometimes requires special knowledge (such as the Security Description Definition Language).
  - Make sure that Active Directory is functioning correctly, and fix it if it is not.

Once you have confirmed any or all of these, restart the universal forwarder to ensure it gets a new authentication token from a domain controller.

**Note:** When assigning access, it's best practice to use the least permissive security paradigm. This entails denying all access to a resource initially, and only then granting access for specific users as necessary.

**For more information**

See the Admin Manual for information on getting started for Windows admins.

Have additional questions or need more help? Be sure to visit Splunk Answers and see what questions and answers the Splunk community has around troubleshooting Splunk on Windows.
System administration problems

Troubleshoot high memory usage

Problem

Your Splunk platform instance goes down because it runs out of memory.

Or, the Monitoring Console alerts you to excessive physical memory usage (either through a platform alert or a health check).

Causes

To diagnose the cause of the excessive memory usage, confirm whether Splunk software is responsible, identify which process type is responsible, and understand how the memory usage changes over time.

First, determine whether Splunk software is responsible for the excessive memory usage:

2. At the top of the historical data section, select a time range of the 30-60 minutes leading up to the time that the issue appeared.
3. Scroll down to the Physical Memory Usage panel (which can be median, average, or max).
4. Verify that the Splunk software physical memory usage nears or exceeds the capacity of the machine. If the machine runs an operating system supported by platform instrumentation, this should be easy to determine at a glance. See About the platform instrumentation framework.

Next, look at the Physical Memory Usage panel to assess the system memory usage issue and note the growth pattern, or the shape of the data. The growth pattern helps distinguish between a leak and high usage as follows:

- A memory leak grows steadily and does not go away until you restart splunkd. A leak is likely a Splunk software defect.
- If the memory issue is not a leak, if it grows then plateaus at a high level (that is, a level near capacity), your Splunk software usage might simply require that much memory.

Finally, identify which process class (search, main splunkd, or other) is involved as follows:

2. Scroll down to the Physical memory usage by process class panel. Most Splunk software out-of-memory situations are search related, but not all.

Solution

If you confirm that Splunk software is not using a large amount of memory, consult your sysadmin about pruning non-Splunk processes.
For cases that are related to Splunk software but not attributed to search processes (especially if the main splunkd process grows in memory usage over time), contact Splunk Support.

If you have attributed the excessive memory usage to searches, in Splunk Web select Settings > Monitoring Console > Search > Activity > Search activity: Instance. Scroll down to the Top 20 Memory-Consuming Searches panel to identify and review the individual offending searches. The following is a list of solutions to the most common search memory usage problems:

- If a few of your searches are using a lot of memory, make sure they are as efficient as possible. Remember to filter early in a search and choose search commands that use memory efficiently. See Quick tips for optimization and Write better searches in the Search Manual.
- Consider limiting the memory usage per search. See Limit search process memory usage in the Search Manual.
- Note that certain Splunk apps have additional system requirements. For example, Enterprise Security requires a search head with significantly more memory than Splunk Enterprise requires by default. See Deployment planning in the Enterprise Security documentation.
- If you have a single search using unreasonable amounts of memory, and you are not sure why, check Known Issues and file a Support ticket. The problem is especially likely to be caused by a defect if the search process displays a growth pattern indicating a leak.
- Remember not to schedule all your reports on the hour. Offset scheduled reports to avoid reaching your concurrent search limit.
- Enable the "Critical system physical memory usage" Platform Alert for this for next time. See Enable and configure platform alerts in the Monitoring Splunk Enterprise Manual.

I get errors about ulimit in splunkd.log

Are you seeing messages like these in splunkd.log while running Splunk software on *nix, possibly accompanied by a Splunk software crash?

```
03-03-2011 21:50:09.027 INFO  ulimit - Limit: virtual address space size: unlimited
03-03-2011 21:50:09.027 ERROR ulimit - Splunk may not work due to low file size limit
03-03-2011 21:50:09.027 INFO  ulimit - Limit: data file size: 2147483646 bytes
03-03-2011 21:50:09.027 INFO  ulimit - Limit: cpu time: unlimited
03-03-2011 21:50:09.027 INFO  loader - Splunkd starting (build 95063).
```

If so, you might need to adjust your server ulimit. Ulimit controls the resources available to a *nix shell and processors the *nix shell has started. A machine running Splunk software needs higher limits than are provided by default.

Check current limits

There are a few ways you can check your current ulimit settings.

- On the command line, you can type `ulimit -a`
- You can restart Splunk Enterprise and look in splunkd.log for events mentioning ulimit:
The monitoring console has a health check for ulimits. See Access and customize health check in Monitoring Splunk Enterprise.

Set new limits

Your Splunk administrator determines the correct level and sets each of these values. To persistently modify the values, edit the limit settings in your operating system. How you do this depends on the version of *nix that you run:

- For earlier versions of Linux that use the init system, edit the /etc/security/limits.conf file.
- For the latest versions of Linux that run the systemd system, edit either /etc/systemd/system.conf, /etc/systemd/user.conf or, if Splunk software has been configured to run as a systemd service, /etc/systemd/system/splunkd.service. This path might vary depending on your distribution of Linux.

The most important values are:

- **The file size** (ulimit -f). The size of an uncompressed bucket file can be very high.
- **The data segment size** (ulimit -d). Increase the value to at least 1 GB = 1073741824 bytes.
- **The number of open files** (ulimit -n), sometimes called the number of file descriptors. This should be at least 8192. Your machine might concurrently need file descriptors for every forwarder socket, deployment client socket, file to be indexed, and user connected. Each bucket can use 10 to 100 files, every search consumes up to four file descriptors, and KV store can use many file descriptors.
- **The max user processes** (ulimit -u). This number must be large enough to accommodate all Splunk threads. The thread count grows with concurrent http connections, parallel pipelines, KV store, and most of all concurrent searches. If you must have a limit (other than unlimited), choose a value in the high thousands or tens of thousands.

Another value that you might need to modify on an older system (but not on most modern systems) is the system-wide file size, fs.file-max, in /etc/sysctl.conf.


Set limits using /etc/security/limits.conf

These instructions are for machines that run the init service.

1. Become the root user or an administrative equivalent with `su`:
   ```
sudo su -
   ```
2. Open `/etc/security/limits.conf` with a text editor.
3. Add at least the following values, or confirm that they exist:
   ```
   * hard nofile 64000
   * hard nproc 8192
   * hard fsize -1
   ```
4. Save the file and exit the text editor.
5. Restart the machine to complete the changes.

Set limits using the /etc/systemd configuration files

These instructions are for machines that run the systemd service. Editing the `/etc/systemd/system.conf` file sets system-wide limits, while editing `/etc/systemd/user.conf` sets limits for services that run under a specific user within
systemd.

Splunk has not released an official systemd unit file for splunkd, but this Splunk answer details a Splunk community effort to create one, and you can use it as the basis for creating a systemd unit file on your machine.

1. Become the root user or an administrative equivalent with su:

   sudo su -

2. Open /etc/systemd/system.conf with a text editor.

3. Add at least the following values to the file:

   [Manager]
   DefaultLimitFSIZE=-1
   DefaultLimitNOFILE=64000
   DefaultLimitNPROC=8192

4. Save the file and exit the text editor.

5. Restart the machine to complete the changes.

Splunk Enterprise does not start due to unusable filesystem

If you receive an error message like the following when you start Splunk Enterprise on a *nix machine, it might be because the software does not know how to write to your machine filesystem.

homePath='/opt/splunk/var/lib/splunk/audit/db' of index=_audit on unusable filesystem. Validating databases (splunkd validatedb) failed with code '1'.

Splunk Enterprise must be able to write to the local filesystem to index your data. Splunk provides support for many different filesystems, as described in System requirements in the Installation Manual. On machines with an unrecognized filesystem, Splunk Enterprise runs a utility called locktest that confirms whether it can work with the filesystem. If locktest fails for any reason, splunkd does not start, to prevent you from indexing data to a filesystem that it cannot write to.

The locktest utility can fail for a number of reasons:

- The filesystem is not known, and Splunk Enterprise cannot perform the proper file locking on it.
- The filesystem has been marked as read-only, or has otherwise been changed by the operating system.
- A library or function that locktest uses to perform the tests is not available or cannot be loaded.

This troubleshooting topic does not apply to Splunk Enterprise instances that run on Windows machines.

Temporarily bypass filesystem checks

If you are a Splunk administrator who understands the risks, you can temporarily bypass filesystem checks to get Splunk Enterprise running again.

Configuring this setting can be dangerous and is not supported in normal operations. Irrevocable data loss can occur. You perform this action solely at your own risk. By configuring the setting, you actively bypass filesystem checks that confirm if Splunk Enterprise can run on your machine filesystem. In a production environment, you must not use this setting as a long-term solution to a filesystem problem. If you use the setting under the guidance of Splunk Support, immediately report any problems that you encounter with indexing or search.

Use the setting in one or more of the following scenarios only:
• You are a skilled Splunk administrator and understand the risks of bypassing filesystem checks.
• You use Splunk software in a development environment.
• You want to recover from a situation where the default filesystem has been changed outside of your control, such as during an operating system upgrade.
• You want to recover from a situation where a Splunk bug has invalidated a previously functional filesystem after an upgrade.
• You want to evaluate the performance of a filesystem for which Splunk has not yet offered support.
• You have been given explicit instruction from Splunk Support to use the setting to solve a problem where Splunk software does not start because of a failed filesystem check.
• You understand and accept all of the risks of using the setting, up to and including losing all your data with no ability to recover it.

1. On the machine that is experiencing the failure, open a shell prompt.
2. Become root or an administrative equivalent with `sudo`:
   
   ```
   sudo su -
   ```
3. Open `$SPLUNK_HOME/etc/splunk-launch.conf` with a text editor.

   `$SPLUNK_HOME` represents where you have installed Splunk Enterprise. For example, if you installed Splunk Enterprise in `/opt/splunk`, then you would edit `/opt/splunk/etc/splunk-launch.conf`.

4. In the file, add the following line anywhere:

   ```
   OPTIMISTIC_ABOUT_FILE_LOCKING=1
   ```
5. Save the file and close the text editor.
6. Restart Splunk Enterprise.
7. Confirm that the `splunkd` service has started.

### HTTP thread limit issues

When you run Splunk Enterprise in a way that uses lots of HTTP connections for Representational State Transfer (REST) operations (for example, a deployment server in a large distributed environment), you might encounter undesirable behavior, including but not limited to logging of errors in `splunkd.log` like the following.

```
03-19-2015 14:36:10.971 -0500 WARN  HttpListener - Can't handle request for /services/broker/connect/8D0E0E2C-8EB5-40D2-9E8A-083F8E9B2516/ISP1065C/241655/windows-x64/8089, max thread limit for REST HTTP server is 6008, threads already in use is 6008
```

This error occurs because, as of Splunk Enterprise 6.0 and later, the software limits the number of REST HTTP connections an instance uses to prevent service failure caused by resource exhaustion.

### How Splunk Enterprise calculates threads and sockets for REST HTTP operations

Splunk Enterprise needs threads and file descriptors to perform REST HTTP operations. Threads let the processes perform tasks, and sockets let the processes communicate with the network. If Splunk Enterprise runs out of either HTTP sockets or threads, it can't complete REST calls to its backend and any such calls fail. Splunk Enterprise thus reserves threads and file descriptors to use for these services.

Splunk Enterprise uses the following formulas to compute the thread limit for HTTP REST:

```
MAX_THREADS = MAX_RAM / (256K * sizeof(void *))
MAX_HTTP_REST_THREADS = MAX_THREADS / 3
```
When it starts, Splunk Enterprise determines the amount of available memory in the host, in bytes. By default, it divides this number by the default stack size and pointer size to get the total number of available threads. It then divides the result by three. This final number is the number of threads available for REST HTTP operations.

- The way Splunk Enterprise calculates the initial MAX_RAM value varies from platform to platform. For example, on a Centos system, MAX_RAM can be retrieved using the following command:

  ```sh
cat /proc/meminfo | grep MemTotal * 1024
  ```

  - 256K (=262144 bytes) is the default stack size
  - `sizeof(void *)` is the size of the pointer, which is hardware and compiler specific and cannot be easily retrieved unless programmatically. Typically, this value is 8 on a 64-bit system as we need 8 bytes (64-bits) to store a memory address. On a 32-bit system, this value is typically 4.
  - The calculated MAX_THREADS value must be less than the value of "ulimit -u" and it must be greater than 20 but less than 150000, which is the hard limit.

For example, on a 64-bit Centos system with 8GB RAM (results rounded down to the nearest integer):

```
MAX_THREADS = (8251723776 / (262144 * 8)) = 3935
MAX_HTTP_REST_THREADS = 3935 / 3 = 1312
```

It then checks the number of available file descriptors for the system, as configured by the `ulimit` command. It divides that number by three. The result is the number of file descriptors available for sockets for REST HTTP operations. For example, if the number of open file descriptors is 36000, then Splunk Enterprise reserves 12000 for sockets for REST HTTP operations.

The number of available file descriptors is different than the number of threads. Both must be present before Splunk Enterprise can make REST calls.

**Override automatic socket and thread configuration**

You can override this automatic configuration by making changes to `server.conf`. Increasing the number of threads can increase the amount of memory that the Splunk Enterprise instance uses.

1. In the `$SPLUNK_HOME/etc/system/local`, create or edit `server.conf`.
2. In the `[httpServer]` stanza, set the `maxThreads` attribute to specify the number of threads for REST HTTP operations that Splunk Enterprise should use.
3. Set the `maxSockets` attribute to specify the number of sockets that should be available for REST HTTP operations.
4. Save the file.
5. Restart Splunk Enterprise. The changes should take effect immediately.

The following example sets the number of HTTP threads to 100000 and the number of sockets to 50000:

```
[httpServer]
maxThreads=100000
maxSockets=50000
```