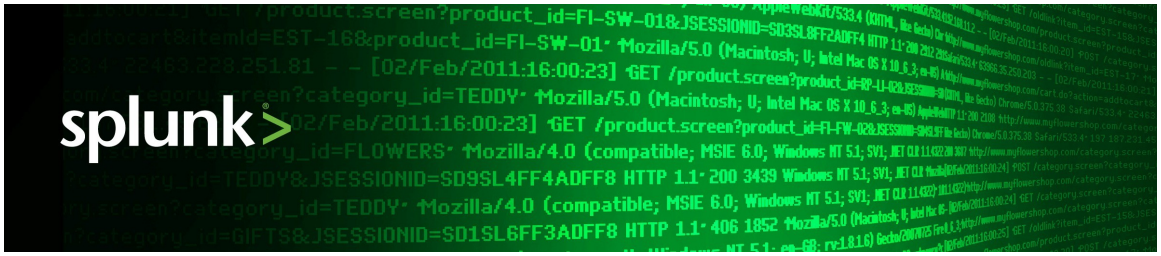


Splunk® Enterprise Admin Manual 6.5.0

Generated: 10/29/2016 2:54 pm

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Welcome to Splunk Enterprise administration

How to use this manual

This manual provides information about the different ways you can administer Splunk. It also introduces you to some initial administration tasks for Windows and *nix.

Note: Unless otherwise stated, tasks and processes in this manual are suitable for both Windows and *nix operating systems.

For a bigger picture overview of the Splunk administration process, including tasks not described in this manual (such as setting up users or data and security configuration), see "Splunk Administration: The big picture," in this manual.

For a list and simple description of the other manuals available to Splunk users, see "Other manuals for the Splunk administrator".

What you can do with the Administration Manual

Task:	Look here:
Start Splunk and do some initial configuration	All the things you need to do to get started on Splunk, from starting Splunk and installing your license, to binding Splunk to an IP. See: "What to do first" for more information.
Use Splunk Web to configure and administer Splunk	An overview of Splunk Web and how you can use it to administer Splunk. See "Use Splunk Web" for more information.
Use configuration files to configure and administer Splunk	A discussion about configuration files: where to find them, how to create and edit them, and some important stuff about file precedences. See "About configuration files" to get started.

Use the Splunk command line interface (CLI) to configure and administer Splunk	An overview of how to use the Command Line Interface to configure Splunk. See "About the CLI" for more information.
Optimize Splunk on Windows	Some Windows-specific things you should know about working with Splunk, including some tips for optimal deployment and information about working with system images. See "Introduction for Windows admins" for more information.
Learn about Splunk licenses	Install your license then go here to learn everything you need to know about Splunk licenses: "Manage Splunk licenses" for more information.
Get familiar with Splunk apps	An introduction and overview of Splunk Apps and how you might integrate them into your Splunk configuration. See "Meet Splunk apps" for more information.
Manage user settings	The Manage users chapter shows you how to manage settings for users. For more information about creating users, see Users and role-based access control in the Securing Splunk Enterprise manual.

Splunk platform administration: the big picture

The Admin Manual provides information about the initial administration tasks as well as information about the different methods you can use to administer your Splunk software. For a more specific overview of what you can do with the Admin Manual, see How to use this manual.

Below are administration tasks you might want to do after initial configuration and where to go to learn more.

Task:	Look here:
Perform backups	Back up configuration information Back up indexed data Set a retirement and archiving policy

Define alerts	The *Alerting Manual*
Manage search jobs	Manage search jobs

For more administration help, see the manuals described below.

Install and upgrade Splunk Enterprise

The Installation Manual describes how to install and upgrade Splunk Enterprise. For information on specific tasks, start here.

Task:	Look here:
Understand installation requirements	Plan your installation
Estimate hardware capacity needs	Estimate hardware requirements
Install Splunk	Install Splunk Enterprise on Windows Install Splunk Enterprise on Unix, Linux, or MacOS
Upgrade Splunk Enterprise	Upgrade from an earlier version

Get data in

Getting Data In is the place to go for information about data inputs: how to consume data from external sources and how to enhance the value of your data.

Task:	Look here:
Learn how to consume external data	How to get data into Splunk
Configure file and directory inputs	Get data from files and directories
Configure network inputs	Get network events
Configure Windows inputs	Get Windows data
Configure miscellaneous inputs	Other ways to get stuff in
Enhance the value of your data	Configure event processing Configure timestamps Configure indexed field extraction Configure host values Configure source types Manage event segmentation Use lookups and workflow actions

See how your data will look after indexing	Preview your data
Improve the process	Improve the data input process

Manage indexes and indexers

Managing Indexers and Clusters tells you how to configure indexes. It also explains how to manage the components that maintain indexes: indexers and clusters of indexers.

Task:	Look here:
Learn about indexing	Indexing overview
Manage indexes	Manage indexes
Manage index storage	Manage index storage
Back up indexes	Back up indexed data
Archive indexes	Set a retirement and archiving policy
Learn about clusters and index replication	About clusters and index replication
Deploy clusters	Deploy clusters
Configure clusters	Configure clusters
Manage clusters	Manage clusters
Learn about cluster architecture	How clusters work

Scale Splunk platform deployments

The Distributed Deployment Manual describes how to distribute Splunk platform functionality across multiple components, such as forwarders, indexers, and search heads. Associated manuals cover distributed components in detail:

- The Forwarding Data Manual describes forwarders.
- The Distributed Search Manual describes search heads.
- The Updating Splunk Components Manual explains how to use the deployment server and forwarder management to manage your deployment.

Task:	Look here:
Learn about distributed Splunk	Scale deployments

platform deployments	
Perform capacity planning for Splunk platform deployments	Estimate hardware requirements
Learn how to forward data	Forward data
Distribute searches across multiple indexers	Search across multiple indexers
Update the deployment	Deploy configuration updates across your environment

Secure Splunk Enterprise

Securing Splunk tells you how to secure your Splunk Enterprise deployment.

Task:	Look here:
Authenticate users and edit roles	User and role-based access control
Secure data with SSL	Secure authentication and encryption
Audit Splunk software	Audit system activity
Use Single Sign-On (SSO) with Splunk software	Configure Single Sign-on
Use Splunk software with LDAP	Set up user authentication with LDAP

Troubleshoot Splunk software

The Troubleshooting Manual provides overall guidance on Splunk platform troubleshooting. In addition, topics in other manuals provide troubleshooting information on specific issues.

Task:	Look here:
Learn about Splunk platform troubleshooting tools	First steps
Learn about Splunk log files	Splunk log files
Work with Splunk support	Contact Splunk support
Resolve common problems	Some common scenarios

References and other information

The Splunk documentation includes several useful references, as well as some other sources of information that might be of use to the Splunk software administrator.

Reference:	Look here:
Configuration file reference	Configuration file reference in the Admin Manual
REST API reference	REST API Reference Manual
CLI help	Available through installed instances of Splunk Enterprise. For details on how to invoke it, read Get help with the CLI in the Admin Manual.
Release information	Release Notes
Information on managing Splunk platform knowledge objects	Knowledge Manager Manual

Other manuals for the Splunk platform administrator

The *Admin Manual* is one of several books with important information and procedures for the Splunk Enterprise administrator. But it's just the beginning of what you can do with Splunk Enterprise.

If you need to configure, run, or maintain Splunk Enterprise as a service for yourself or other users, start with this book. Then go to these other manuals for details on specific areas of Splunk Enterprise administration.

Manual	What it covers	Key topic areas
Getting Data In	Specifying data inputs and improving how Splunk software handles data	**How to get data into Splunk** **Configure event processing** **Preview your data**
Managing Indexers and Clusters	Managing Splunk indexers and clusters of indexers	**About indexing and indexers** **Manage indexes** **Back up and**

		archive your indexes **About clusters and index replication** **Deploy clusters**
Distributed Deployment	Scaling your deployment to fit the needs of your enterprise.	**Distributed Splunk overview**
Forwarding Data	Forwarding data into Splunk.	**Forward data**
Distributed Search	Using search heads to distribute searches across multiple indexers.	**Search across multiple indexers**
Updating Splunk Components	Using the deployment server and forwarder management to update Splunk components such as forwarders and indexers.	**Deploy updates across your environment**
Securing Splunk	Data security and user authentication	**User authentication and roles** **Encryption and authentication with SSL** **Auditing**
Monitoring Splunk Enterprise	Use included dashboards and alerts to monitor and troubleshoot your Splunk Enterprise deployment	**About the monitoring console**
Troubleshooting	Solving problems	**First steps** **Splunk log files** **Some common scenarios**
Installation	Installing and upgrading Splunk	**System requirements** **Step by step installation procedures** **Upgrade from an earlier version**

The topic "Learn to administer Splunk" provides more detailed guidance on where to go to read about specific admin tasks.

Other books of interest to the Splunk administrator

In addition to the manuals that describe the primary administration tasks, you might want to visit other manuals from time to time, depending on the size of your Splunk Enterprise installation and the scope of your responsibilities. These are other manuals in the Splunk Enterprise documentation set:

- **Search Tutorial**. This manual provides an introduction to searching with Splunk.
- **Knowledge Manager**. This manual describes how to manage Splunk knowledge objects, such as event types, tags, lookups, field extractions, workflow actions, saved searches, and views.
- **Alerting**. This manual describes Splunk's alerting and monitoring functionality.
- **Data Visualizations**. This manual describes the range of visualizations that Splunk provides.
- **Search Manual**. This manual tells you how to search and how to use the Splunk search language.
- **Search Reference**. This reference contains a detailed catalog of the Splunk search commands.
- **Developing Views and Apps for Splunk Web**. This manual explains how to develop views and apps using advanced XML. It also contains other developer topics, such as custom scripts and extending Splunk.
- **REST API Reference**. This manual provides information on all publicly accessible REST API endpoints.
- **Release Notes**. Look here for information about new features, known issues, and fixed problems.

The larger world of Splunk documentation

For links to the full set of Splunk Enterprise documentation, including the manuals listed above, visit: **Splunk Enterprise documentation**.

To access all the Splunk documentation, including manuals for apps, go to this page: **Welcome to Splunk documentation**.

Make a PDF

If you'd like a PDF version of this manual, click the red **Download the Admin Manual as PDF** link below the table of contents on the left side of this page. A PDF version of the manual is generated on the fly. You can save it or print it to read later.

Introduction for Windows admins

Welcome!

Splunk is a powerful, effective tool for Windows administrators to resolve problems that occur on their Windows networks. Its out-of-the-box feature set positions it to be the secret weapon in the Windows administrator's toolbox. The ability to add apps that augment its functionality makes it even more extensible. And it has a growing, thriving community of users.

How to use this manual as a Windows user

This manual has topics that will help you experiment with, learn, deploy, and get the most out of Splunk.

Unless otherwise specified, the information in this manual is helpful for both Windows and *nix users. If you are unfamiliar with Windows or *nix operational commands, we strongly recommend you check out Differences between *nix and Windows in Splunk operations.

We've also provided some extra information in the chapter "get the most out of Splunk on Windows". This chapter is intended for Windows users to help you make the most of Splunk and includes the following information.

Deploy Splunk on Windows provides some considerations and preparations specific to Windows users. Use this topic when you plan your deployment.

Optimize Splunk for peak performance describes ways to keep your Splunk on Windows deployment running properly, either during the course of the deployment, or after the deployment is complete.

Put Splunk onto system images helps you make Splunk a part of every Windows system image or installation process. From here you can find tasks for installing Splunk and Splunk forwarders onto your system images.

For more information

Here's some additional Windows topics of interest in other Splunk manuals:

- An overview of all of the installed Splunk for Windows services (from the Installation Manual)
- What Splunk can monitor (from the Getting Data In Manual)

- Considerations for deciding how to monitor remote Windows data (from the Getting Data In Manual). Read this topic for important information on how to get data from multiple machines remotely.
- Consolidate data from multiple hosts (from the Universal Forwarder Manual)

Other useful information:

- Where is my data? (from the Getting Data In Manual)
- Use Splunk's Command Line Interface (CLI) (from the Getting Data In Manual)
- Sources, sourcetypes and fields (from the Getting Data In Manual)
- Fields and field extraction (from the Knowledge Manager Manual)
- Real-time searches (from the User Manual)
- Saved searches (from the User Manual)
- Dashboard creation (from the User Manual)

If you need help

If you are looking for in-depth Splunk knowledge, a number of education programs are available.

When you get stuck, Splunk has a large free support infrastructure that can help:

- Splunk Answers.
- The Splunk Community Wiki.
- The Splunk Internet Relay Chat (IRC) channel (EFNet #splunk). (IRC client required)

If you still don't have an answer to your question, you can get in touch with Splunk's support team. The Support Contact page tells you how to do that.

Note: Levels of support above the community level require an Enterprise license. To get one, you'll need to speak with the Sales team.

About Splunk Free

Splunk Free is the totally free version of Splunk. The Free license lets you index up to 500 MB per day and will never expire.

The 500 MB limit refers to the amount of new data you can add (we call this indexing) per day. But you can keep adding data every day, storing as much as you want. For example, you could add 500 MB of data per day and eventually have 10 TB of data in Splunk Enterprise.

If you need more than 500 MB/day, you'll need to purchase an Enterprise license. See How Splunk licensing works for more information about licensing.

Splunk Free regulates your license usage by tracking license violations. If you go over 500 MB/day more than 3 times in a 30 day period, Splunk Free continues to index your data, but disables search functionality until you are back down to 3 or fewer warnings in the 30 day period.

Is Splunk Free for you?

Splunk Free is designed for personal, ad hoc search and visualization of IT data. You can use Splunk Free for ongoing indexing of small volumes (<500 MB/day) of data. Additionally, you can use it for short-term bulk-loading and analysis of larger data sets--Splunk Free lets you bulk-load much larger data sets up to 3 times within a 30 day period. This can be useful for forensic review of large data sets.

What is included with Splunk Free?

Splunk Free is a single-user product. All Splunk Enterprise features are supported, with the following exceptions:

- Distributed search configurations (including search head clustering) are not available.
- Forwarding in TCP/HTTP formats is not available. This means you can forward data to other Splunk platform instances, but not to non-Splunk software.
- Deployment management capabilities are not available.
- Alerting (monitoring) is not available.
- Indexer clustering is not available.
- Report acceleration summaries are not available.
- While a Splunk Free instance can be used as a forwarder (to a Splunk Enterprise indexer) it cannot be the client of a deployment server.
- There is no authentication or user and role management when using Splunk Free. This means:
    - There is no login. The command line or browser can access and control all aspects of Splunk Free with no user/password prompt.

- All accesses are treated as equivalent to the admin user. There is only one role (admin), and it is not configurable. You cannot add more roles or create user accounts.
- Searches are run against all public indexes, 'index=*'.
- Restrictions on search, such as user quotas, maximum per-search time ranges, and search filters, are not supported.
- The capability system is disabled. All available capabilities are enabled for all users accessing Splunk Free.

Switching to Free from an Enterprise Trial license

When you first download and install Splunk, you are automatically using an Enterprise Trial license. You can continue to use the Enterprise Trial license until it expires, or switch to the Free license right away, depending on your requirements.

What you should know about switching to Free

Splunk Enterprise Trial gives you access to a number of features that are not available in Splunk Free. When you switch, **be aware of the following**:

- User accounts or roles that you have created will no longer work.
- Anyone connecting to the instance will automatically be logged on as admin. You will no longer see a login screen, though you will see the update check occur.
- Any knowledge objects created by any user other than admin (such as event type, transaction, or source type definitions) and not already globally shared will not be available. If you need these knowledge objects to continue to be available after you switch to Splunk Free, you can do one of the following:
  - Use Splunk Web to promote them to be globally available before you switch. See Manage app and add-on objects.
  - Hand edit the configuration files they are in to promote them. See App architecture and object ownership.
- Any alerts you have defined will no longer trigger. You will **no longer receive alerts** from Splunk software. You can still schedule searches to run for dashboards and summary indexing purposes.
- Configurations in outputs.conf to forward to third-party applications in TCP or HTTP formats will stop working.

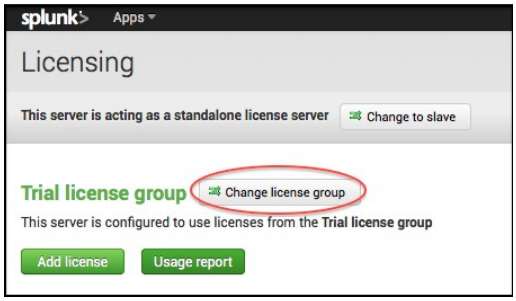
When you attempt to make any of the above configurations in Splunk Web while using an Enterprise Trial license, you will be warned about the above limitations in Splunk Free.

How do I switch to Splunk Free?

If you currently have Splunk Enterprise (trial or not), you can either wait for your Enterprise license to expire, or switch to a Free license at any time. To switch to a Free License:

1. Log in to Splunk Web as a user with admin privileges and navigate to **Settings > Licensing**.

2. Click **Change license group** at the top of the page.



3. Select **Free license** and click **Save**.

4. You are prompted to restart.

Differences between *nix and Windows in Splunk operations

This topic clarifies the functional differences that you'll encounter between *nix and Windows operating systems, under the context in which they matter in Splunk operations. It does not delve into technical comparisons of - or advocacy for - either flavor of OS, but rather explains why you'll see things referenced one way or another on various OS-specific Splunk manual pages.

Paths

A major difference in the way that *nix operating systems handle files and directories is the type of slash used to separate files or directories in the pathname. *nix systems use the forward slash, ("/"). Windows, on the other hand, uses the backslash ("\").

An example of a *nix path:

```
/opt/splunk/bin/splunkd
```

An example of a Windows path:

```
C:\Program Files\Splunk\bin\splunkd.exe
```

Environment variables

Another area where the operating systems differ is in the representation of environment variables. Both systems have a way to temporarily store data in one or more environment variables. On *nix systems, this is shown by using the dollar sign ("$") in front of the environment variable name, like so:

```
# SPLUNK_HOME=/opt/splunk; export $SPLUNK_HOME
```

On Windows, it's a bit different - to specify an environment variable, you need to use the percent sign ("%"). Depending on the type of environment variable you are using, you may need to place one or two percent signs before the environment name, or on either side of the name.

```
> set SPLUNK_HOME="C:\Program Files\Splunk"
> echo %SPLUNK_HOME%
C:\Program Files\Splunk
>
```

To set the %SPLUNK_HOME% variable in the Windows environment, you can do one of two things:

- Edit `splunk-launch.conf` in %SPLUNK_HOME%\etc.
- Set the variable by accessing the "Environment Variables" window. Open an Explorer window, and on the left pane, right-click "My Computer", then select "Properties" from the window that appears. Once the System Properties window appears, select the "Advanced" tab, then click on the "Environment Variables" button that appears along the bottom window of the tab.

Configuration files

Splunk Enterprise works with configuration files that use ASCII/UTF-8 character set encoding. When you edit configuration files on Windows, configure your text editor to write files with this encoding. On some Windows versions, UTF-8 is not

the default character set encoding. See How to edit a configuration file.

Ways you can configure Splunk

Splunk maintains its configuration information in a set of **configuration files**. You can configure Splunk by using any (or all!) of these methods:

- Use Splunk Web.
- Use Splunk's Command Line Interface (CLI) commands.
- Edit Splunk's configuration files directly.
- Use App setup screens that use the Splunk REST API to update configurations.

All of these methods change the contents of the underlying configuration files. You may find different methods handy in different situations.

Use Splunk Web

You can perform most common configuration tasks in Splunk Web. Splunk Web runs by default on port 8000 of the host on which it is installed:

- If you're running Splunk on your local machine, the URL to access Splunk Web is `http://localhost:8000`.
- If you're running Splunk on a remote machine, the URL to access Splunk Web is `http://<hostname>:8000`, where `<hostname>` is the name of the machine Splunk is running on.

Administration menus can be found under **Settings** in the Splunk Web menu bar. Most tasks in the Splunk documentation set are described for Splunk Web. For more information about Splunk Web, see Meet Splunk Web.

Edit configuration files

Most of Splunk's configuration information is stored in .conf files. These files are located under your Splunk installation directory (usually referred to in the documentation as `$SPLUNK_HOME`) under `/etc/system`. In most cases you can copy these files to a local directory and make changes to these files with your preferred text editor.

Before you begin editing configuration files, read "About configuration files".

Use Splunk CLI

Many configuration options are available via the CLI. These options are documented in the CLI chapter in this manual. You can also get CLI help reference with the `help` command while Splunk is running:

```
./splunk help
```

For more information about the CLI, refer to "About the CLI" in this manual. If you are unfamiliar with CLI commands, or are working in a Windows environment, you should also check out Differences between *nix and Windows in Splunk operations.

Setup screens for an app

Developers can create setup screens for an app that allow users to set configurations for that app without editing the configuration files directly. Setup screens make it easier to distribute apps to different environments, or to customize an app for a particular usage.

Setup screens use Splunk's REST API to manage the app's configuration files.

For more information about setup screens, refer to Create a setup page for a Splunk app on the Splunk Developer Portal.

Managing a distributed environment

The Splunk deployment server provides centralized management and configuration for distributed environments. You can use it to deploy sets of configuration files or other content to groups of Splunk instances across the enterprise.

For information about managing deployments, refer to the "Updating Splunk Components" manual.

Get the most out of Splunk Enterprise on Windows

Deploy Splunk on Windows

You can integrate Splunk into your Windows environment in any number of ways. This topic discusses some of those scenarios and offers guidelines on how to best adapt your Splunk for Windows deployment to your enterprise.

While this topic is geared more toward deploying Splunk in a Windows environment, Splunk itself also has distributed deployment capabilities that you should be aware of, even as you integrate it into your Windows enterprise. The Distributed Deployment Manual has lots of information on spreading Splunk services across a number of computers.

When deploying Splunk on Windows on a large scale, you can rely completely on your own deployment utilities (such as System Center Configuration Manager or Tivoli/BigFix) to place both Splunk and its configurations on the machines in your enterprise. Or, you can integrate Splunk into system images and then deploy Splunk configurations and apps using Splunk's deployment server.

Concepts

When you deploy Splunk into your Windows network, it captures data from the machines and stores it centrally. Once the data is there, you can search and create reports and dashboards based on the indexed data. More importantly, for system administrators, Splunk can send alerts to let you know what is happening as the data arrives.

In a typical deployment, you dedicate some hardware to Splunk for indexing purposes, and then use a combination of universal forwarders and Windows Management Instrumentation (WMI) to collect data from other machines in the enterprise.

Considerations

Deploying Splunk in a Windows enterprise requires a number of planning steps.

First, you must inventory your enterprise, beginning at the physical network, and leading up to how the machines on that network are individually configured. This

includes, but is not limited to:

- Counting the number of machines in your environment and defining a subset of those which need Splunk installed. Doing this defines the initial framework of your Splunk topology.
- Calculating your network bandwidth, both in your main site and at any remote or external sites. Doing this determines where you will install your main Splunk instance, and where and how you will use Splunk forwarders.
- Assessing the current health of your network, particularly in areas where networks are separated. Making sure your edge routers and switches are functioning properly will allow you to set a baseline for network performance both during and after the deployment.

Then, you must answer a number of questions prior to starting the deployment, including:

- **What data on your machines needs indexing? What part of this data do you want to search, report, or alert across?** This is probably the most important consideration to review. The answers to these questions determine how you address every other consideration. It determines where to install Splunk, and what types of Splunk you use in those installations. It also determines how much computing and network bandwidth Splunk will potentially use.

- **How is the network laid out? How are any external site links configured? What security is present on those links?** Fully understanding your network topology helps determine which machines you should install Splunk on, and what types of Splunk (indexers or forwarders) you should install on those machines from a networking standpoint.

A site with thin LAN or WAN links makes it necessary to consider how much Splunk data should be transferred between sites. For example, if you have a hub-and-spoke type of network, with a central site connected to branch sites, it might be a better idea to deploy forwarders on machines in the branch sites, which send data to an intermediate forwarder in each branch. Then, the intermediate forwarder would send data back to the central site. This is a less costly move than having all machines in a branch site forward their data to an indexer in the central site.

If you have external sites that have file, print or database services, you'll need to account for that traffic as well.

- **How is your Active Directory (AD) configured?** How are the operations masters roles on your domain controllers (DCs) defined? Are all domain controllers centrally located, or do you have controllers located in satellite sites? If your AD is distributed, are your bridgehead servers configured properly? Is your Inter-site Topology Generator (ISTG)-role server functioning correctly? If you are running Windows Server 2008 R2, do you have read-only domain controllers (RODCs) in your branch sites? If so, then you have to consider the impact of AD replication traffic as well as Splunk and other network traffic.

- **What other roles are the servers in your network playing?** Splunk indexers need resources to run at peak performance, and sharing servers with other resource-intensive applications or services (such as Microsoft Exchange, SQL Server and even Active Directory itself) can potentially lead to problems with Splunk on those machines. For additional information on sharing server resources with Splunk indexers, see "Introduction to capacity planning for Splunk Enterprise" in the Capacity Planning Manual.

- **How will you communicate the deployment to your users?** A Splunk installation means the environment is changing. Depending on how Splunk is rolled out, some machines will get new software installed. Users might incorrectly link these new installs to perceived problems or slowness on their individual machine. You should keep your user base informed of any changes to reduce the number of support calls related to the deployment.

Prepare your Splunk on Windows deployment

How you deploy Splunk into your existing environment depends on the needs you have for Splunk, balanced with the available computing resources you have, your physical and network layouts, and your corporate infrastructure. As there is no one specific way to deploy Splunk, there are no step-by-step instructions to follow. There are, however, some general guidelines to observe.

For a more successful Splunk deployment:

- **Prepare your network.** Before integrating Splunk into your environment:
    - Make sure that your network is functioning properly, and that all switches, routers and cabling are correctly configured.
    - Replace any broken or failing equipment.
    - Ensure any virtual LANs (VLANs) are properly set up.

- Test network throughput, particularly between sites with thin network links.

- **Prepare your Active Directory.** While AD is not a requirement to run Splunk, it's a good idea to ensure that it is functioning properly prior to your deployment. This includes but is not limited to:
  - Identifying all of your domain controllers, and the operations master roles any of them might perform. If you have RODCs at your branch sites, make sure that they have the fastest connections as possible to operations masters DCs.
  - Ensuring that AD replication is functioning correctly, and that all site links have a DC with a copy of the global catalog.
  - If your forest is divided into multiple sites, make sure your ISTG role server is functioning properly, or that you have assigned at least two bridgehead servers in your site (one primary, one backup).
  - Ensuring that your DNS infrastructure is working properly.

You might need to place DCs on different subnets on your network, and seize flexible single master operations (FSMO, or operations master) roles as necessary to ensure peak AD operation and replication performance during the deployment.

- **Define your Splunk deployment.** Once your Windows network is properly prepared, you must now determine where Splunk will go in the network. Consider the following:
  - Determine the set(s) of data that you want Splunk to index on each machine, and whether or not you need for Splunk to send alerts on any collected data.
  - Dedicate one or more machines in each network segment to handle Splunk indexing, if possible. For additional information on capacity planning for a distributed Splunk deployment, review "Introduction to capacity planning for Splunk Enterprise" in the Capacity Planning Manual.
  - Don't install full Splunk on machines that run resource-intensive services like AD (in particular, DCs that hold FSMO roles), any version of Exchange, SQL Server, or machine virtualization product such as Hyper-V or VMWare. Instead, use a universal forwarder, or connect to those machines using WMI.
  - If you're running Windows Server 2008/2008 R2 Core, remember that you'll have no GUI available to make changes using Splunk Web when you install Splunk on those machines.
  - Arrange your Splunk layout so that it uses minimal network resources, particularly across thin WAN links. Universal forwarders

greatly reduce the amount of Splunk-related traffic sent over the wire.

- **Communicate your deployment plans to your users.** It's important to advise your users about the status of the deployment, throughout the course of it. This will significantly reduce the amount of support calls you receive later.

Optimize Splunk for peak performance

Like many services, Splunk on Windows needs proper maintenance in order to run at peak performance. This topic discusses the methods that you can apply to keep your Splunk on Windows deployment running properly, either during the course of the deployment, or after the deployment is complete.

To ensure peak Splunk performance:

- **Designate one or more machines solely for Splunk operations.** Splunk scales horizontally. This means that more physical computers dedicated to Splunk, rather than more resources in a single computer, translate into better performance. Where possible, split up your indexing and searching activities across a number of machines, and only run main Splunk services on those machines. With the exception of the universal forwarder performance is reduced when you run Splunk on servers that share other services.

- **Dedicate fast disks for your Splunk indexes.** The faster the available disks on a system are for Splunk indexing, the faster Splunk will run. Use disks with spindle speeds faster than 10,000 RPM when possible. When dedicating redundant storage for Splunk, use hardware-based RAID 1+0 (also known as RAID 10). It offers the best balance of speed and redundancy. Software-based RAID configurations through the Windows Disk Management utility are not recommended.

- **Don't allow anti-virus programs to scan disks used for Splunk operations.** When anti-virus file system drivers scan files for viruses on access, performance is significantly reduced, especially when Splunk internally ages data that has recently been indexed. If you must use anti-virus programs on the servers running Splunk, make sure that *all* Splunk directories and programs are excluded from on-access file scans.

- **Use multiple indexes, where possible.** Distribute the data that in indexed by Splunk into different indexes. Sending all data to the default index can cause I/O bottlenecks on your system. Where appropriate, configure your indexes so that they point to different physical volumes on your systems, when possible. For information on how to configure indexes, read "Configure your indexes" in this manual.

- **Don't store your indexes on the same physical disk or partition as the operating system.** The disk that holds your Windows OS directory (`%WINDIR%`) or its swap file is not recommended for Splunk data storage. Put your Splunk indexes on other disks on your system.

For more information on how indexes are stored, including information on database bucket types and how Splunk stores and ages them, review "How Splunk stores indexes" in this manual.

- **Don't store the hot and warm database buckets of your Splunk indexes on network volumes.** Network latency will decrease performance significantly. Reserve fast, local disk for the hot and warm buckets of your Splunk indexes. You can specify network shares such as Distributed File System (DFS) volumes or Network File System (NFS) mounts for the cold and frozen buckets of the index, but note that searches that include data stored in the cold database buckets will be slower.

- **Maintain disk availability, bandwidth and space on your Splunk indexers.** Make sure that the disk volumes that hold Splunk's indexes maintain 20% or more free space at all times. Disk performance decreases proportionally to available space because disk seek times increase. This affects how fast Splunk indexes data, and can also determine how quickly search results, reports and alerts are returned. In a default Splunk installation, the drive(s) that contain your indexes must have at least 5000 megabytes (approximately 5 gigabytes) of free disk space, or indexing will pause.

Put Splunk onto system images

This topic explains the concepts of making Splunk a part of every Windows system image or installation process. It also guides you through the general process of integration, regardless of the imaging utilities that you use.

- For more specific information about getting Windows data into Splunk, review "About Windows data and Splunk" in the Getting Data In Manual.
- For information on distributed Splunk deployments, read "Distributed overview" in the Distributed Deployment Manual. This overview is essential reading for understanding how to set up Splunk deployments, irrespective of the operating system that you use. You can also read about Splunk's distributed deployment capabilities there.
- For information about planning larger Splunk deployments, read "Introduction to capacity planning for Splunk Enterprise" in the Capacity Planning Manual and "Deploying Splunk on Windows" in this manual.

Concepts for system integration on Windows

The main reason to integrate Splunk into Windows system images is to ensure that Splunk is available immediately when the machine is activated for use in the enterprise. This frees you from having to install and configure Splunk after activation.

In this scenario, when a Windows system is activated, it immediately launches Splunk after booting. Then, depending on the type of Splunk instance installed and the configuration given, Splunk either collects data from the machine and forwards it to an indexer (in many cases), or begins indexing data that is forwarded from other Windows machines.

System administrators can also configure Splunk instances to contact a deployment server, which allows for further configuration and update management.

In many typical environments, universal forwarders on Windows machines send data to a central indexer or group of indexers, which then allow that data to be searched, reported and alerted on, depending on your specific needs.

Considerations for system integration

Integrating Splunk into your Windows system images requires planning.

In most cases, the preferred Splunk component to integrate into a Windows system image is a universal forwarder. The universal forwarder is designed to share resources on computers that perform other roles, and does much of the work that an indexer can, at much less cost. You can also modify the forwarder's configuration using Splunk's deployment server or an enterprise-wide configuration manager with no need to use Splunk Web to make changes.

In some situations, you may want to integrate a full instance of Splunk into a system image. Where and when this is more appropriate depends on your specific needs and resource availability.

Splunk doesn't recommend that you include a full version of Splunk in an image for a server that performs any other type of role, unless you have specific need for the capability that an indexer has over a forwarder. Installing multiple indexers in an enterprise does not give you additional indexing power or speed, and can lead to undesirable results.

Before integrating Splunk into a system image, consider:

- **the amount of data you want Splunk to index, and where you want Splunk to send that data, if applicable.** This feeds directly into disk space calculations, and should be a top consideration.
- **the type of Splunk instance to install on the image or machine.** Universal forwarders have a significant advantage when installing on workstations or servers that perform other duties, but might not be appropriate in some cases.
- **the available system resources on the imaged machine.** How much disk space, RAM and CPU resources are available on each imaged system? Will it support a Splunk install?
- **the resource requirements of your network.** Splunk needs network resources, whether you're using it to connect to remote machines using WMI to collect data, or you're installing forwarders on each machine and sending that data to an indexer.
- **the system requirements of other programs installed on the image.** If Splunk is sharing resources with another server, it can take available resources from those other programs. Consider whether or not you should install other programs on a workstation or server that is running a full instance of Splunk. A universal forwarder will work better in cases like this, as it is designed to be lightweight.
- **the role that the imaged machine plays in your environment.** Will it be a workstation only running productivity applications like Office? Or will it be an operations master domain controller for your Active Directory forest?

Integrate Splunk into a System Image

Once you have determined the answers to the questions in the checklist above, the next step is to integrate Splunk into your system images. The steps listed are generic, allowing you to use your favorite system imaging or configuration tool to complete the task.

Choose one of the following options for system integration:

- Integrate a universal forwarder into a system image
- Integrate a full version of Splunk into a system image

Integrate a universal forwarder onto a system image

This topic discusses the procedure to integrate a Splunk universal forwarder into a Windows system image. For additional information about integrating Splunk Enterprise into images, see Integrate Splunk Enterprise into system images.

1. On a reference computer, install and configure Windows the way that you want, including installing Windows features, service packs, and other components.
2. Install and configure necessary applications, taking into account Splunk's system and hardware capacity requirements.
3. Install and configure the universal forwarder from the command line. You must supply at least the `LAUNCHSPLUNK=0` command line flag when you perform the installation.
4. Proceed through the graphical portion of the install, selecting the inputs, deployment servers, and/or forwarder destinations you need.
5. Once you have completed the install, open a command prompt or PowerShell window.
6. From this prompt, edit any additional configuration files that are not configurable in the installer.
7. After you edit configuration files, from the prompt, change to the universal forwarder `bin` directory.
8. Run `./splunk clone-prep-clear-config`.
9. Close the command prompt or PowerShell window.
10. In the Services Control Panel, configure the `splunkd` service to start automatically by setting its startup type to 'Automatic'.
11. Prepare the system image for domain participation using a utility such as Windows System Image Manager (WSIM). Microsoft recommends using SYSPREP or WSIM as the method to change machine Security Identifiers (SIDs) prior to cloning, as opposed to using third-party tools (such as Ghost Walker or NTSID.)
12. After you have configured the system for imaging, reboot the machine and clone it with your favorite imaging utility.

The image is now ready for deployment.

Integrate full Splunk onto a system image

This topic discusses the procedure to integrate a full version of Splunk into a Windows system image. For additional information about integrating Splunk into images, see "Put Splunk onto system images" in this manual.

To integrate a full version of Splunk into a system image:

1. Using a reference computer, install and configure Windows to your liking, including installing any needed Windows features, patches and other components.

2. Install and configure any necessary applications, taking into account Splunk's system and hardware capacity requirements.

3. Install and configure Splunk.

Important: You can install using the GUI installer, but more options are available when installing the package from the command line.

4. Once you have configured Splunk inputs, open a command prompt.

5. From this prompt, stop Splunk by changing to the `%SPLUNK_HOME%\bin` directory and issuing a `.\splunk stop`

6. Clean any event data by issuing a `.\splunk clean eventdata`.

7. Close the command prompt window.

8. Ensure that the `splunkd` and `splunkweb` services are set to start automatically by setting their startup type to 'Automatic' in the Services Control Panel.

9. Prepare the system image for domain participation using a utility such as SYSPREP (for Windows XP and Windows Server 2003/2003 R2) and/or Windows System Image Manager (WSIM) (for Windows Vista, Windows 7, and Windows Server 2008/2008 R2).

Note: Microsoft recommends using SYSPREP and WSIM as the method to change machine Security Identifiers (SIDs) prior to cloning, as opposed to using third-party tools (such as Ghost Walker or NTSID.)

10. Once you have configured the system for imaging, reboot the machine and clone it with your favorite imaging utility.

The image is now ready for deployment.

Administer Splunk Enterprise with Splunk Web

Launch Splunk Web

After Splunk is running, you can launch the Web interface, **Splunk Web**. To learn more about Splunk Web, see:

- Admin tasks with Splunk Web
- Navigating Splunk Web
- Using Splunk Search

To launch Splunk Web, navigate to:

```
http://mysplunkhost:<port>
```

Using whatever host and port you chose during installation.

The first time you log in to Splunk with an Enterprise license, the default login details are:
Username - *admin*
Password - *changeme*

Note: Splunk with a free license does not have access controls, so you will not be prompted for login information.

Note: Starting in Splunk version 4.1.4, you cannot access Splunk Free from a remote browser until you have edited `$SPLUNK_HOME/etc/local/server.conf` and set `allowRemoteLogin` to `Always`. If you are running Splunk Enterprise, remote login is disabled by default (set to `requireSetPassword`) for the admin user until you change the default password.

Admin tasks with Splunk Web

Splunk Web is Splunk's browser-based interface. Here are just a few of the things you can do in Splunk Web:

- Configure your data inputs
- Search data and report and visualize results
- Investigate problems
- Manage users natively or via LDAP strategies
- Troubleshoot Splunk deployments
- Manage clusters and peers

Refer to the system requirements for a list of supported operating systems and browsers.

Splunk Settings menu

Splunk Web provides a convenient interface for managing most aspects of Splunk operations. Most of the functions can be accessed by clicking **Settings** in the menu. From here you can:

Manage your data

Under **Settings > Data** you can do the following:

- **Data Inputs** Lets you view a list of data types and configure them. To add an input, click the **Add data** button in the Data Inputs page. For more information about how to add data, see the *Getting Data In* manual.
- **Forwarding and receiving** lets you set up your forwarders and receivers. For more information about setting up forwarding and receiving, see the Forwarding Data manual.
- **Indexes** lets you add, disable, and enable indexes.
- **Report acceleration summaries** takes you to the searching and reporting app to lets you review your existing report summaries. For more information about creating report summaries, see the *Knowledge Manager Manual*.

Manage users and user authentication

By navigating to **Settings > Users and Authentication > Access Control** you can do the following:

- Create and manage users
- Define and assign roles
- Set up LDAP authentication strategies

For more information about working with users and authentication, see the Securing Splunk manual.

Work with Apps

To see your installed **apps**, select **Apps** in the menu bar.

From this page, you can select an app from a list of those you have already installed and are currently available to you. From here you can also access the following menu options:

- **Find more Apps** lets you search for and install additional apps.
- **Manage Apps** lets you manage your existing apps.

You can also access all of your apps in the Home page.

For more information about apps, see Developing views and apps for Splunk Web.

Manage aspects of your system

The options under **Settings > System** let you do the following:

- **Server settings** lets you manage Splunk settings like ports, host name, index paths, email server, and system logging and deployment client information. For more about configuring and managing distributed environments with Splunk Web, see the Updating Splunk Components manual.
- **Server controls** lets you restart Splunk.
- **Licensing** lets you manage and renew your Splunk licenses.

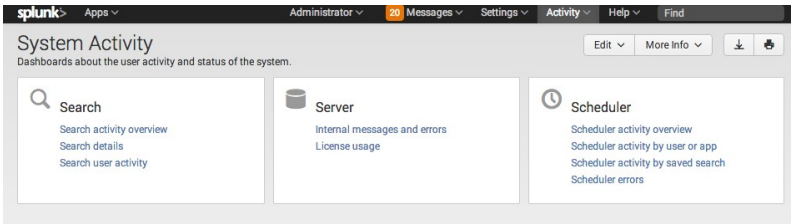
Splunk Enterprise default dashboards

Splunk Enterprise comes packaged with a set of useful dashboards. They help you to troubleshoot your system and searches and can also help you come up with ideas about how you might want to design dashboards and views of your own.

Activity dashboards

You can find the following dashboards by clicking **Activity > System Activity** in the user bar near the top of the page.

Note: These dashboards are visible only to users with Admin role permissions. See "Add and manage users" in *Securing Splunk Enterprise*. For information about setting up permissions for dashboards, see the *Knowledge Manager manual*.



- **Search activity** - This dashboard collection provides at-a-glance info about search activity for your Splunk instance. You can find out when searches are running, the amount of load they're putting on the system, which searches are the most popular, which search views and dashboards are getting the most usage, and more. The following dashboards are provided:
    - Search activity overview
    - Search details
    - Search user activity
- **Server activity** - This collection of dashboards provides metrics related to splunkd and Splunk Web performance and is handy for troubleshooting. You'll find the numbers of errors reported, lists of the most recent errors, lists of timestamping issues and unhandled exceptions, a chart displaying recent browser usage, and more. The following dashboards are provided:
    - Internal messages and errors
    - License usage
- **Scheduler activity** - This collection of dashboards gives you insight into the work of the search scheduler, which ensures that both ad hoc and scheduled searches are run in a timely manner.
    - Scheduler activity overview
    - Scheduler activity by user or app
    - Scheduler activity by saved search
    - Scheduler errors

The Summary Dashboard

The Summary dashboard is the first thing you see as you enter the Search & Reporting app. It provides a search bar and time range picker which you can use

to input and run your initial search.

When you add an input to Splunk, that input gets added relative to the app you're in. 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 review the summary dashboard and you don't see data that you're certain is in Splunk, be sure that you're looking at the right index.

You may want to add the index that an app uses to the list of default indexes for the role you're using. For more information about roles, refer to this topic about roles in Securing Splunk.For more information about Summary Dashboards, see the Search Tutorial.

Customize Splunk Web banner messages

You can add and edit notifications that display in the **Messages** menu in Splunk Web.

You need admin or system user level privileges to add or edit a notification.

To change or add a notification:

1. Select **Settings> User Interface**.

2. Click **New** to create a new message, or click **Bulletin Messages** and select the message you want to edit.

3. Edit the existing message text, or give your new message a name and message text.

4. Click **Save.** The message will now appear when the user accesses **Messages** in the menu.

Use Splunk Web with a proxy server

When Splunk Web is located behind a proxy server, users may have trouble with Splunk Web links that access the Splunk website. For example, some Splunk Web pages link directly to the download site for Splunk apps and many "learn more" links will take you to the online documentation.

To resolve this, simply set the `HTTP_PROXY` environment variable. For permanent results, you can specify the setting in the `splunk-launch.conf` configuration file, located in `$SPLUNK_HOME/etc/` on *nix systems and `%SPLUNK_HOME%\etc\` on Windows.

Note: The App Manager is not supported for use with a proxy server, if you use a proxy server with Splunk Web, you must download and update apps manually.

In `splunk-launch.conf`, add this attribute/value pair:

```
HTTP_PROXY = <IP address or host name>:<port number>
```

For example:

```
HTTP_PROXY = 10.1.8.11:8787
```

Important: If your proxy server only handles HTTPS requests, you must use the following attribute/value pair:

```
HTTPS_PROXY = <IP address or host name>:<port number>
```

For example:

```
HTTPS_PROXY = 10.1.8.11:8888
```

Administer Splunk Enterprise with configuration files

About configuration files

Splunk Enterprise configuration information is stored in **configuration files**. These files are identified by the `.conf` extension and hold the information for different aspects of your configurations. These aspects include:

- System settings
- Authentication and authorization information
- Index mappings and setting
- Deployment and cluster configurations
- Knowledge objects and saved searches

For a list of configuration files and an overview of the area each file covers, see "List of configuration files" in this manual.

Most configuration files come packaged with your Splunk software in the `$SPLUNK_HOME/etc/system/default/` directory.

Use Splunk Web to manage configuration files

When you change your configuration in Splunk Web, that change is written to a copy of the configuration file for that setting. Splunk software creates a copy of this configuration file (if it does not exist), writes the change to that copy, and adds it to a directory under `$SPLUNK_HOME/etc/...`. The directory that the new file is added to depends on a number of factors that are discussed in "Configuration file directories" in this manual. The most common directory is `$SPLUNK_HOME/etc/system/local`, which is used in the example.

If you add a new index in Splunk Web, the software performs the following actions:

1. Checks for a copy of the file.

2. If no copy exists, the software creates a copy of `indexes.conf` and adds it to a directory, such as `$SPLUNK_HOME/etc/system/local`.

3. Writes the change to the copy of `indexes.conf`.

4. Leaves the default file unchanged in `$SPLUNK_HOME/etc/system/default`.

Editing the configuration file directly

While you can do a lot of configuration from Splunk Web, you can also edit the configuration files directly for any setting. For some advanced customizations that Splunk Web does not support, edit the configuration files directly.

Note: Editing configuration files requires more frequent restarts than making your changes in Splunk Web. See When to restart Splunk after a configuration file change in this manual.

Important: Never change or copy the configuration files in the default directory. Default files must remain intact and in their original location. To change settings for a particular configuration file, you must first create a new version of the file in a non-default directory and then add the settings that you want to change. For information on the directories where you can edit configuration files, see Configuration file directories. When you first create this new version of the file, start with an empty file. Do not start from a copy of the default directory.

Before you change any configuration files:

- Learn about how the default configuration files work, and where to put the copies that you edit. See "Configuration file directories" in this manual.
- Learn about the structure of the stanzas that comprise configuration files and how the attributes you want to edit are set up. See "Configuration file structure" in this manual.
- Learn how different copies of the same configuration files in different directories are layered and combined so that you know the best place to put your copies. See "Configuration file precedence" in this manual.

After you are familiar with the configuration file content and directory structure, and understand how to leverage Splunk Enterprise configuration file precedence, see "How to edit a configuration file" to learn how to safely modify your files.

Configuration file directories

A single Splunk instance typically has multiple versions of configuration files across several of directories. You can have configuration files with the same names in your default, local, and app directories. This creates a layering effect that allows Splunk to determine configuration priorities based on factors such as

the current user and the current app.

To learn more about how configurations are prioritized by Splunk, see "Configuration file precedence".

Note: The most accurate list of settings available for a given configuration file is in the `.spec` file for that configuration file. You can find the latest version of the `.spec` and `.example` files in the "Configuration file reference", or in `$SPLUNK_HOME/etc/system/README`.

About the default files

"all these worlds are yours, except /default - attempt no editing there"

-- duckfez, 2010

The default directory contains preconfigured versions of the configuration files. The location of the default directory is `$SPLUNK_HOME/etc/system/default`.

Important: Never change or copy the configuration files in the default directory. Default files must remain intact and in their original location. The Splunk Enterprise upgrade process overwrites the default directory, so any changes that you make in the default directory are lost on upgrade. Changes that you make in non-default configuration directories, such as `$SPLUNK_HOME/etc/system/local` or `$SPLUNK_HOME/etc/apps/<app_name>/local`, persist through upgrades.

To change attribute values for a particular configuration file, you must first create a new version of the file in a non-default directory and then modify the values there. Values in a non-default directory have precedence over values in the default directory.

Note: When you first create this new version of the file, start with an empty file and add only the attributes that you need to change. Do not start from a copy of the default directory. If you copy the entire default file to a location with higher precedence, any changes to the default values that occur through future Splunk Enterprise upgrades cannot take effect, because the values in the copied file will override the updated values in the default file.

Where you can place (or find) your modified configuration files

You can layer several versions of a configuration files, with different attribute values used by Splunk according to the layering scheme described in "Configuration file precedence".

Never edit files in their default directories. Instead, create and/or edit your files in one of the configuration directories, such as `$SPLUNK_HOME/etc/system/local`. These directories are not overwritten during upgrades.

For most deployments you can use the `$SPLUNK_HOME/etc/system/local` directory to make configuration changes. However, in certain situations you may want to work with the files in other directories. The following is the configuration directory structure in `$SPLUNK_HOME/etc`:

- `$SPLUNK_HOME/etc/system/local`
  - Local changes on a site-wide basis go here; for example, settings you want to make available to all apps. If the configuration file you're looking for doesn't already exist in this directory, create it and give it write permissions.
- `$SPLUNK_HOME/etc/slave-apps/[_cluster|<app_name>]/[local|default]`
  - **For cluster peer nodes only.**
  - The subdirectories under `$SPLUNK_HOME/etc/slave-apps` contain configuration files that are common across all peer nodes.
  - **Do not** change the content of these subdirectories on the cluster peer itself. Instead, use the cluster master to distribute any new or modified files to them.
  - The `_cluster` directory contains configuration files that are not part of real apps but that still need to be identical across all peers. A typical example is the `indexes.conf` file.
  - For more information, see "Update common peer configurations" in the Managing Indexers and Clusters manual.
- `$SPLUNK_HOME/etc/apps/<app_name>/[local|default]`
  - If you're in an app when a configuration change is made, the setting goes into a configuration file in the app's `/local` directory. For example, edits for search-time settings in the default Splunk search app go here: `$SPLUNK_HOME/etc/apps/search/local/`.
  - If you want to edit a configuration file so that the change only applies to a certain app, copy the file to the app's `/local` directory (with write permissions) and make your changes there.
- `$SPLUNK_HOME/etc/users`
  - User-specific configuration changes go here.
- `$SPLUNK_HOME/etc/system/README`
  - This directory contains supporting reference documentation. For most configuration files, there are two reference files: `.spec` and `.example`; for example, `inputs.conf.spec` and `inputs.conf.example`. The `.spec` file specifies the syntax, including a list of available attributes and variables. The `.example` file contains examples of real-world usage.

Configuration file structure

Before you edit configuration files, you should familiarize yourself with the structure of the files.

Stanzas

Configuration files consist of one or more **stanzas**, or sections. Each stanza begins with a stanza header in square brackets. This header identifies the settings held within that stanza. Each setting is an attribute value pair that specifies particular configuration settings.

For example, `inputs.conf` provides an `[SSL]` that includes settings for the server certificate and password (among other things):

```
[SSL]
serverCert = <pathname>
password = <password>
```

Depending on the stanza type, some of the attributes might be required, while others could be optional.

Setting up a new stanza

When you edit a configuration file, you might be changing the default stanza, like above, or you might need to add a brand-new stanza.

Here's the basic pattern:

```
[stanza1_header]
<attribute1> = <val1>
# comment
<attribute2> = <val2>
...

[stanza2_header]
<attribute1> = <val1>
<attribute2> = <val2>
...
```

Important: Attributes are case-sensitive. For example, `sourcetype = my_app` is **not** the same as `SOURCETYPE = my_app`. One will work; the other won't.

Stanza scope

Configuration files frequently have stanzas with varying scopes, with the more specific stanzas taking precedence. For example, consider this example of an `outputs.conf` configuration file, used to configure **forwarders**:

```
[tcpout]
indexAndForward=true
compressed=true

[tcpout:my_indexersA]
autoLB=true
compressed=false
server=mysplunk_indexer1:9997, mysplunk_indexer2:9997

[tcpout:my_indexersB]
autoLB=true
server=mysplunk_indexer3:9997, mysplunk_indexer4:9997
```

Note that this example file has two levels of stanzas:

- The global `[tcpout]`, with settings that affect all tcp forwarding.
- Two `[tcpout:<target_list>]` stanzas, whose settings affect only the indexers defined in each target group.

The setting for `compressed` in `[tcpout:my_indexersA]` overrides that attribute's setting in `[tcpout]`, *for the indexers in the my_indexersA target group only.*

For more information on forwarders and `outputs.conf`, see "Configure forwarders with outputs.conf".

Configuration file precedence

For more information about configuration files, read About configuration files.

Splunk software uses **configuration files** to determine nearly every aspect of its behavior. A Splunk platform deployment can have many copies of the same configuration file. These file copies are usually layered in directories that affect either the users, an **app**, or the system as a whole.

When editing configuration files, it is important to understand how Splunk software evaluates these files and which ones take precedence.

When incorporating changes, Splunk software does the following to your configuration files:

- It merges the settings from all copies of the file, using a location-based prioritization scheme.
- When different copies have conflicting attribute values (that is, when they set the same attribute to different values), it uses the value from the file with the highest priority.
- It determines the priority of configuration files by their location in its directory structure, according to whether the file is located in a system, app, or user directory, in that order. To determine priority among the collection of apps directories, Splunk uses ASCII sort order. Files in an apps directory named "A" have a higher priority than files in an apps directory named "B", and so on.

Note: Besides resolving configuration settings amongst multiple copies of a file, Splunk software sometimes needs to resolve settings within a single file. For information on how Splunk software determines precedence within a single `props.conf` file, see "Attribute precedence within a single props.conf file".

About configuration file context

How precedence is determined depends upon the context of the file.

App or user context versus global context

To determine priority among copies of a configuration file, Splunk software first determines the directory scheme.

Splunk software uses two main schemes of directory precedence.

- App or user: Some activities, like searching, take place in an app or user context. The app and user context is vital to search-time processing, where certain knowledge objects or actions might be valid only for specific users in specific apps.
- Global: Activities like indexing take place in a global context. They are independent of any app or user. For example, configuration files that determine monitoring behavior occur outside of the app and user context and are global in nature.

Cluster peer configuration context

There's also an expanded precedence order for cluster peer node global configurations. This is because some configuration files, like `indexes.conf`, must be identical across peer nodes.

To keep them consistent, files are managed from the cluster master, which distributes them to the peer nodes so that all peer nodes contain the same versions of the files. These files have the highest precedence in a cluster peer's configuration, which is explained in the next section.

For more information about how configurations are distributed across peer nodes, see "Update common peer configurations" in the Managing Indexers and Clusters manual.

How Splunk determines precedence order

This subsection provides a conceptual understanding of precedence order and context. For ordered listings by directory name, see "Summary of directory order precedence", later in this topic.

Precedence order within global context:

When the context is global (that is, where there's no app/user context), directory priority descends in this order:

1. System local directory -- highest priority
2. App local directories
3. App default directories
4. System default directory -- lowest priority

When consuming a global configuration, such as `inputs.conf`, Splunk first uses the attributes from any copy of the file in `system/local`. Then it looks for any copies of the file located in the app directories, adding any attributes found in them, but ignoring attributes already discovered in system/local. As a last resort, for any attributes not explicitly assigned at either the system or app level, it assigns default values from the file in the `system/default` directory.

Note: As the next section describes, cluster peer nodes have an expanded order of precedence.

Precedence for cluster peer nodes

For cluster peer nodes, the global context considers some additional peer-specific ("slave-app") directories. These directories contain apps and configurations that are identical across all peer nodes. Here is the expanded precedence order for cluster peers:

1. Slave-app local directories **(cluster peers only)** -- highest priority
2. System local directory
3. App local directories
4. Slave-app default directories **(cluster peers only)**
5. App default directories
6. System default directory -- lowest priority

With cluster peers, custom settings common to all the peers (those in the slave-app local directories) have the highest precedence.

Precedence order within app or user context

When there's an app/user context, directory priority descends from user to app to system:

1. User directories for current user -- highest priority
2. App directories for currently running app (local, followed by default)
3. App directories for all other apps (local, followed by default) -- for exported settings only
4. System directories (local, followed by default) -- lowest priority

An attribute in `savedsearches.conf`, for example, might be set at all three levels: the user, the app, and the system. Splunk will always use the value of the user-level attribute, if any, in preference to a value for that same attribute set at the app or system level.

How app directory names affect precedence

Note: For most practical purposes, the information in this subsection probably won't matter, but it might prove useful if you need to force a certain order of evaluation or for troubleshooting.

To determine priority among the collection of apps directories, Splunk uses ASCII sort order. Files in an apps directory named "A" have a higher priority than files in an apps directory named "B", and so on. Also, all apps starting with an uppercase letter have precedence over any apps starting with a lowercase letter,

due to ASCII sort order. ("A" has precedence over "Z", but "Z" has precedence over "a", for example.)

In addition, numbered directories have a higher priority than alphabetical directories and are evaluated in lexicographic, not numerical, order. For example, in descending order of precedence:

```
$SPLUNK_HOME/etc/apps/myapp1
$SPLUNK_HOME/etc/apps/myapp10
$SPLUNK_HOME/etc/apps/myapp2
$SPLUNK_HOME/etc/apps/myapp20
...
$SPLUNK_HOME/etc/apps/myappApple
$SPLUNK_HOME/etc/apps/myappBanana
$SPLUNK_HOME/etc/apps/myappZabaglione
...
$SPLUNK_HOME/etc/apps/myappapple
$SPLUNK_HOME/etc/apps/myappbanana
$SPLUNK_HOME/etc/apps/myappzabaglione
...
```

Note: When determining precedence **in the app/user context,** directories for the currently running app take priority over those for all other apps, independent of how they're named. Furthermore, other apps are only examined for exported settings.

Summary of directory precedence

Putting this all together, the order of directory priority, from highest to lowest, goes like this:

Global context:

```
$SPLUNK_HOME/etc/system/local/*

$SPLUNK_HOME/etc/apps/A/local/* ... $SPLUNK_HOME/etc/apps/z/local/*

$SPLUNK_HOME/etc/apps/A/default/* ... $SPLUNK_HOME/etc/apps/z/default/*

$SPLUNK_HOME/etc/system/default/*
```

Global context - cluster peer nodes only:

```
$SPLUNK_HOME/etc/slave-apps/A/local/* ...
$SPLUNK_HOME/etc/slave-apps/z/local/*
```

```
$SPLUNK_HOME/etc/system/local/*

$SPLUNK_HOME/etc/apps/A/local/* ... $SPLUNK_HOME/etc/apps/z/local/*

$SPLUNK_HOME/etc/slave-apps/A/default/* ...
$SPLUNK_HOME/etc/slave-apps/z/default/*

$SPLUNK_HOME/etc/apps/A/default/* ... $SPLUNK_HOME/etc/apps/z/default/*

$SPLUNK_HOME/etc/system/default/*
```

Important: Within the `slave-apps/[local|default]` directories, the special `_cluster` subdirectory has a higher precedence than any app subdirectories starting with a lowercase letter (for example, `anApp`). However, it has a *lower* precedence than any apps starting with an uppercase letter (for example, `AnApp`). This is due to the location of the underscore ("_") character in the ASCII sort order.

App/user context:

```
$SPLUNK_HOME/etc/users/*

$SPLUNK_HOME/etc/apps/Current_running_app/local/*

$SPLUNK_HOME/etc/apps/Current_running_app/default/*

$SPLUNK_HOME/etc/apps/A/local/*, $SPLUNK_HOME/etc/apps/A/default/*, ...
$SPLUNK_HOME/etc/apps/z/local/*, $SPLUNK_HOME/etc/apps/z/default/* (but
see note below)

$SPLUNK_HOME/etc/system/local/*

$SPLUNK_HOME/etc/system/default/*
```

Important: In the app/user context, all configuration files for the currently running app take priority over files from all other apps. This is true for the app's local *and* default directories. So, if the current context is app C, Splunk evaluates both `$SPLUNK_HOME/etc/apps/C/local/*` and `$SPLUNK_HOME/etc/apps/C/default/*` before evaluating the local or default directories for any other apps. Furthermore, Splunk software only looks at configuration data for other apps if that data has been exported globally through the app's `default.meta` file. For more information, see Set permissions for objects in a Splunk app on the Splunk Developer Portal.

Also, note that `/etc/users/` is evaluated only when the particular user logs in or performs a search.

Example of how attribute precedence works

This example of attribute precedence uses `props.conf`. The `props.conf` file is unusual, because its context can be either global or app/user, depending on when Splunk is evaluating it. Splunk evaluates `props.conf` at both index time (global) and search time (apps/user).

Assume `$SPLUNK_HOME/etc/system/local/props.conf` contains this stanza:

```
[source::/opt/Locke/Logs/error*]
sourcetype = fatal-error
```

and `$SPLUNK_HOME/etc/apps/t2rss/local/props.conf` contains another version of the same stanza:

```
[source::/opt/Locke/Logs/error*]
sourcetype = t2rss-error
SHOULD_LINEMERGE = True
BREAK_ONLY_BEFORE_DATE = True
```

The line merging attribute assignments in `t2rss` always apply, as they only occur in that version of the file. However, there's a conflict with the `sourcetype` attribute. In the `/system/local` version, the `sourcetype` has a value of "fatal-error". In the `/apps/t2rss/local` version, it has a value of "t2rss-error".

Since this is a `sourcetype` assignment, which gets applied at index time, Splunk uses the global context for determining directory precedence. In the global context, Splunk gives highest priority to attribute assignments in `system/local`. Thus, the `sourcetype` attribute gets assigned a value of "fatal-error".

The final, internally merged version of the file looks like this:

```
[source::/opt/Locke/Logs/error*]
sourcetype = fatal-error
SHOULD_LINEMERGE = True
BREAK_ONLY_BEFORE_DATE = True
```

List of configuration files and their context

As mentioned, Splunk decides how to evaluate a configuration file based on the context that the file operates within, global or app/user. Generally speaking, files that affect data input, indexing, or deployment activities are global; files that affect search activities usually have a app/user context.

The `props.conf` and `transforms.conf` files can be evaluated in either a app/user or a global context, depending on whether Splunk is using them at index or search time.

Global configuration files

```
admon.conf
authentication.conf
authorize.conf
crawl.conf
deploymentclient.conf
distsearch.conf
indexes.conf
inputs.conf
outputs.conf
pdf_server.conf
procmonfilters.conf
props.conf -- global and app/user context
pubsub.conf
regmonfilters.conf
report_server.conf
restmap.conf
searchbnf.conf
segmenters.conf
server.conf
serverclass.conf
serverclass.seed.xml.conf
source-classifier.conf
sourcetypes.conf
sysmon.conf
tenants.conf
transforms.conf  -- global and app/user context
user-seed.conf -- special case: Must be located in /system/default
web.conf
wmi.conf
```

App/user configuration files

```
alert_actions.conf
app.conf
audit.conf
commands.conf
eventdiscoverer.conf
event_renderers.conf
eventtypes.conf
fields.conf
limits.conf
literals.conf
```

```
macros.conf
multikv.conf
props.conf -- global and app/user context
savedsearches.conf
tags.conf
times.conf
transactiontypes.conf
transforms.conf  -- global and app/user context
user-prefs.conf
workflow_actions.conf
```

Troubleshooting configuration precedence and other issues

Splunk's configuration file system supports many overlapping configuration files in many different locations. The price of this level of flexibility is that figuring out which value for which configuration option is being used in your Splunk installation can sometimes be quite complex. If you're looking for some tips on figuring out what configuration setting is being used in a given situation, read "Use btool to troubleshoot configurations" in the Troubleshooting Manual.

Attribute precedence within a single props.conf file

In addition to understanding how attribute precedence works across files, you also sometimes need to consider attribute priority within a single props.conf file.

Precedence within sets of stanzas affecting the same target

When two or more **stanzas** specify a behavior that affects the same item, items are evaluated by the stanzas' ASCII order. For example, assume you specify in `props.conf` the following stanzas:

```
[source::.../bar/baz]
attr = val1

[source::.../bar/*]
attr = val2
```

The second stanza's value for `attr` will be used, because its path is higher in the ASCII order and takes precedence.

Overriding default attribute priority in props.conf

There's a way to override the default ASCII priority in `props.conf`. Use the `priority` key to specify a higher or lower priority for a given stanza.

For example, suppose we have a source:

```
source::az
```

and the following patterns:

```
[source::...a...]
sourcetype = a

[source::...z...]
sourcetype = z
```

In this case, the default behavior is that the settings provided by the pattern "source::...a..." take precedence over those provided by "source::...z...". Thus, sourcetype will have the value "a".

To override this default ASCII ordering, use the `priority` key:

```
[source::...a...]
sourcetype = a
priority = 5

[source::...z...]
sourcetype = z
priority = 10
```

Assigning a higher priority to the second stanza causes `sourcetype` to have the value "z".

There's another attribute precedence issue to consider. By default, stanzas that match a string literally ("literal-matching stanzas") take precedence over regex pattern-matching stanzas. This is due to the default values of their `priority` keys:

- 0 is the default for pattern-matching stanzas
- 100 is the default for literal-matching stanzas

So, literal-matching stanzas will always take precedence over pattern-matching stanzas, unless you change that behavior by explicitly setting their `priority` keys.

You can use the `priority` key to resolve collisions between patterns of the same type, such as `sourcetype` patterns or `host` patterns. The `priority` key does not, however, affect precedence across spec types. For example, `source` patterns take priority over `host` and `sourcetype` patterns, regardless of priority key values.

Precedence for events with multiple attribute assignments

The `props.conf` file sets attributes for processing individual events by host, source, or sourcetype (and sometimes event type). So it's possible for one event to have the same attribute set differently for the **default fields**: host, source or sourcetype. The precedence order is:

- source
- host
- sourcetype

You might want to override the default `props.conf` settings. For example, assume you are tailing `mylogfile.xml`, which by default is labeled `sourcetype = xml_file`. This configuration will re-index the entire file whenever it changes, even if you manually specify another sourcetype, because the property is set by source. To override this, add the explicit configuration by source:

```
[source::/var/log/mylogfile.xml]
CHECK_METHOD = endpoint_md5
```

How to edit a configuration file

Before you edit a configuration file, make sure you are familiar with the following:

- To learn about where configuration files live, and where to put the ones you edit, see Configuration file directories.
- To learn about file structure and how the attributes you want to edit are set up, see Configuration file structure.
- To learn how configuration files across multiple directories are layered and combined, see Configuration file precedence.

Customize a configuration file

To customize an attribute in a configuration file, create a new file with the same name in a local or app directory. You will then add the specific attributes that you want to customize to the local configuration file.

1. Determine whether the configuration file already exists in your preferred directory, for example $SPLUNK_HOME/etc/system/local. See Configuration file precedence in this manual.
2. If the file already exists in your preferred directory, edit the existing file. Otherwise, create the file in your preferred directory. Do not copy the contents of the default configuration file into the file in your preferred directory. This is to ensure that any Splunk software upgrades properly update the default values.
3. Add only the stanzas and attributes that you want to customize to the local file.

Clear an attribute

You can clear any attribute by setting it to null. For example:

```
forwardedindex.0.whitelist =
```

This overrides any previous value that the attribute held, including any value set in its default file, causing the system to consider the value entirely unset.

Insert a comment

You can insert comments in configuration files. To do so, use the # sign:

```
# This stanza forwards some log files.
[monitor:///var/log]
```

Important: Start the comment at the left margin. Do not put the comment on the same line as the stanza or attribute:

```
[monitor:///var/log]    # This is a really bad place to put your
comment.
```

For an attribute, such as

```
a_setting = 5  #5 is the best number
```

This sets the a_setting attribute to the value "5 #5 is the best number", which may cause unexpected results.

Creating and editing configuration files on Windows and other non-UTF-8 operating systems

The Splunk platform works with configuration files with ASCII/UTF-8 encoding. On operating systems where UTF-8 is not the default character set, for example Windows, configure your text editor to write files in that format.

When to restart Splunk Enterprise after a configuration file change

When you make changes to Splunk Enterprise using the configuration files, you might need to restart Splunk Enterprise for the changes to take effect.

Note: Changes made in Splunk Web are less likely to require restarts. This is because Splunk Web automatically updates the underlying configuration file(s) and notifies the running Splunk instance (splunkd) of the changes.

This topic provides guidelines to help you determine whether to restart after a change. Whether a change requires a restart depends on a number of factors, and this topic does not provide a definitive authority. Always check the configuration file or its reference topic to see whether a particular change requires a restart. For a full list of configuration files and an overview of the area each file covers, see List of configuration files in this manual.

When to restart forwarders

If you make a configuration file change to a heavy forwarder, you must restart the forwarder, but you do not need to restart the receiving indexer. If the changes are part of a deployed app already configured to restart after changes, then the forwarder restarts automatically.

When to restart splunkweb

You must restart splunkweb to enable or disable SSL for Splunk Web access.

When to restart splunkd

As a general rule, restart splunkd after making the following types of changes.

Index changes

- Index time field extractions
- Time stamp properties

Note: When settings that affect indexing are changed through Splunk Web and the CLI, they do not require restarts and take place immediately.

See Update common peer configurations and apps in *Managing Indexers and Clusters of Indexers*.

User and role changes

Any user and role changes made in configuration files require a restart, including:

- LDAP configurations (If you make these changes in Splunk Web you can reload the changes without restarting.)
- Password changes
- Changes to role capabilities
- Splunk Enterprise native authentication changes, such as user-to-role mappings.

System changes

Changes that affect the system settings or server state require restart, such as:

- Licensing changes
- Web server configuration updates
- Changes to general indexer settings (minimum free disk space, default server name, etc.)
- Changes to General settings (e.g., port settings). See Determine which indexes.conf changes require restart in *Managing Indexers and Clusters of Indexers*.
- Changing a forwarder's output settings
- Changing the time zone in the OS of a Splunk Enterprise instance (Splunk Enterprise retrieves its local time zone from the underlying OS at startup)
- Creating a pool of search heads
- Installing some apps may require a restart. Consult the documentation for each app you are installing.

Splunk Enterprise changes that do not require a restart

Settings that apply to search-time processing take effect immediately and do not require a restart. This is because searches run in a separate process that reloads configurations. For example, lookup tables, tags, and event types are re-read for each search.

This includes (but is not limited to) changes to:

- Lookup tables
- Field extractions
- Knowledge objects
- Tags
- Event types

Files that contain search-time operations include (but are not limited to):

- `macros.conf`
- `props.conf`
- `transforms.conf`
- `savedsearches.conf` (If a change creates an endpoint you must restart.)

To view your endpoints type the following into your browser:

http://yoursplunkserver:8000/en-GB/debug/refresh
In addition, index-time props and transforms do not require restarts, as long as your indexers are receiving the data from forwarders. That is to say:

- Changes to `props.conf` and `transforms.conf` on an indexer do not require restarts.
- In an indexer cluster, changes to `props.conf` and `transforms.conf` are automatically reloaded when the peers receive the changes from the master.
- On a non-clustered indexer, changes to `props.conf` and `transforms.conf` require a reload.
- On either a clustered or non-clustered indexer, once the .conf files have reloaded, the changes take effect after a forwarder auto-LB time period.

How to reload files

To reload `transforms.conf`:

`http://yoursplunkserver:8000/en-us/debug/refresh?entity=admin/transforms-lookup` for new lookup file definitions that reside within transforms.conf

`http://yoursplunkserver:8000/en-us/debug/refresh?entity=admin/transforms-extract` for new field transforms/extractions that reside within transforms.conf

To reload `authentication.conf`, use Splunk Web. Go to **Settings > Access controls > Authentication method** and click the **Reload authentication configuration** button. This refreshes the authentication caches, but does not disconnect current users.

Restart an indexer cluster

To learn about restarts in an indexer cluster, and when and how to use a rolling restart, see Restart the entire indexer cluster or a single peer node in *Managing Indexers and Clusters of Indexers*.

Use cases

In complex situations, restarting Splunk Enterprise is the safest practice. Here are a few examples of scenarios where you might (or might not) be able to avoid a restart.

Scenario: You edit search- or index-time transforms in `props.conf` and `search.conf`

Whether to restart depends on if the change is related to a index-time setting or a search-time setting. Index-time settings include:

- line breaking
- timestamp parsing

Search-time settings relate mainly to field extraction and creation and do not require a restart. Any index-time changes still require a restart. So for example:

1. If `props.conf` and `tranforms.conf` are configured as search-time transforms on the index, you don't have to do anything. For any search-time changes, each time you run a search Splunk reloads the `props.conf` and `transforms.conf`.

2. If the search-time changes are on a heavy forwarder, you must restart that forwarder. (If the changes are part of a deployed app configured to restart after changes, then this would happen automatically.)

3. If it is an index-time transform on the indexer you must restart the indexer to

add the changes.

Scenario: You edit `savedsearches.conf` and the new search creates a REST endpoint

You must restart the indexer to integrate the new endpoint.

List of configuration files

The following is a list of some of the available spec and example files associated with each conf file. Some conf files do not have spec or example files; contact Support before editing a conf file that does not have an accompanying spec or example file.

Important: Do not edit the default copy of any conf file in `$SPLUNK_HOME/etc/system/default/`. See How to edit a configuration file.

File	Purpose
alert_actions.conf	Create an alert.
app.conf	Configure app properties
audit.conf	Configure auditing and event hashing. This feature is not available for this release.
authentication.conf	Toggle between Splunk's built-in authentication or LDAP, and configure LDAP.
authorize.conf	Configure roles, including granular access controls.
checklist.conf	Customize monitoring console health check.
collections.conf	Configure KV Store collections for apps.
commands.conf	Connect search commands to any custom search script.
crawl.conf	Configure crawl to find new data sources.
datamodels.conf	Attribute/value pairs for configuring data models.
default.meta.conf	Set permissions for objects in a Splunk app.
deploymentclient.conf	Specify behavior for clients of the deployment server.
distsearch.conf	Specify behavior for distributed search.

event_renderers.conf	Configure event-rendering properties.
eventtypes.conf	Create event type definitions.
fields.conf	Create multivalue fields and add search capability for indexed fields.
indexes.conf	Manage and configure index settings.
inputs.conf	Set up data inputs.
instance.cfg.conf	Designate and manage settings for specific instances of Splunk. This can be handy, for example, when identifying forwarders for internal searches.
limits.conf	Set various limits (such as maximum result size or concurrent real-time searches) for search commands.
literals.conf	Customize the text, such as search error strings, displayed in Splunk Web.
macros.conf	Define search macros in Settings.
multikv.conf	Configure extraction rules for table-like events (ps, netstat, ls).
outputs.conf	Set up forwarding behavior.
passwords.conf	Maintain the credential information for an app.
procmon-filters.conf	Monitor Windows process data.
props.conf	Set indexing property configurations, including timezone offset, custom source type rules, and pattern collision priorities. Also, map transforms to event properties.
pubsub.conf	Define a custom client of the deployment server.
restmap.conf	Create custom REST endpoints.
savedsearches.conf	Define ordinary reports, scheduled reports, and alerts.
searchbnf.conf	Configure the search assistant.
segmenters.conf	Configure segmentation.
server.conf	Enable SSL for Splunk's back-end (communications between Splunkd and Splunk Web) and specify certification locations.

serverclass.conf	Define deployment server classes for use with deployment server.
serverclass.seed.xml.conf	Configure how to seed a deployment client with apps at start-up time.
source-classifier.conf	Terms to ignore (such as sensitive data) when creating a source type.
sourcetypes.conf	Machine-generated file that stores source type learning rules.
tags.conf	Configure tags for fields.
telemetry.conf	Enable apps to collect telemetry data about app usage and other properties.
times.conf	Define custom time ranges for use in the Search app.
transactiontypes.conf	Add additional transaction types for transaction search.
transforms.conf	Configure regex transformations to perform on data inputs. Use in tandem with props.conf.
ui-prefs.conf	Change UI preferences for a view. Includes changing the default earliest and latest values for the time range picker.
user-seed.conf	Set a default user and password.
visualizations.conf	List the visualizations that an app makes available to the system.
viewstates.conf	Use this file to set up IU views (such as charts) in Splunk.
web.conf	Configure Splunk Web, enable HTTPS.
wmi.conf	Set up Windows management instrumentation (WMI) inputs.
workflow_actions.conf	Configure workflow actions.

Configuration parameters and the data pipeline

Data goes through several phases as it transitions from raw input to searchable events. This process is called the **data pipeline** and consists of four phases:

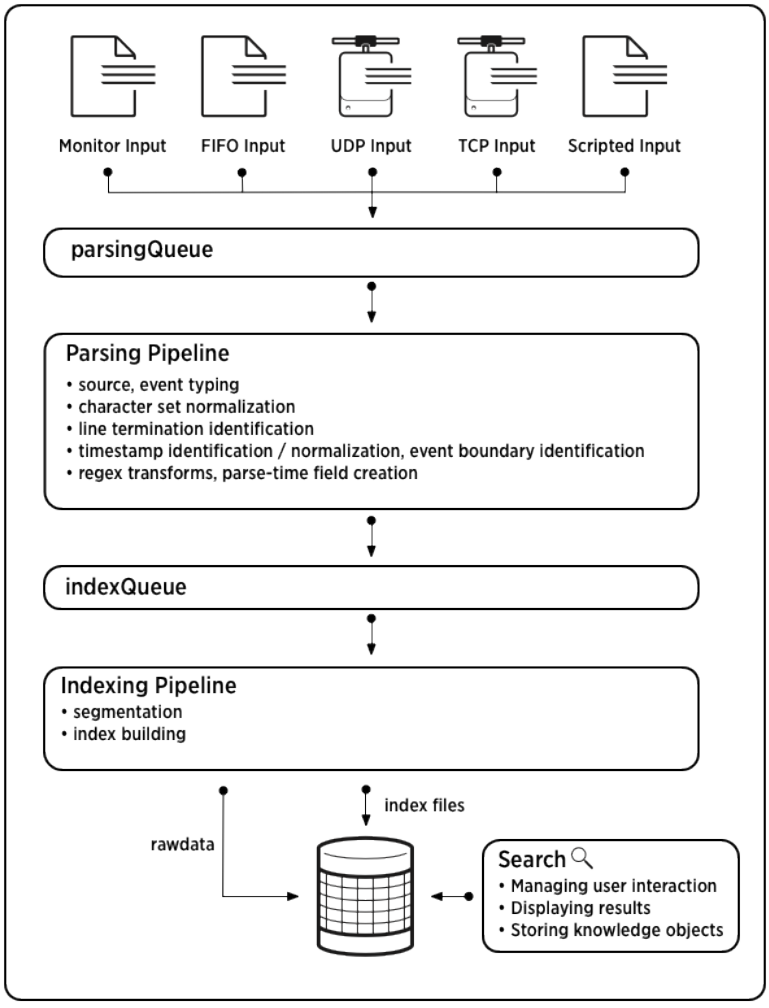
- **Input**

- **Parsing**
- **Indexing**
- **Search**

Each phase of the data pipeline relies on different configuration file parameters. Knowing which phase uses a particular parameter allows you to identify where in your Splunk deployment topology you need to set the parameter.

What the data pipeline looks like

This diagram outlines the data pipeline:



The Distributed Deployment manual describes the data pipeline in detail, in "How data moves through Splunk: the data pipeline".

How Splunk Enterprise components correlate to phases of the pipeline

One or more Splunk Enterprise components can perform each of the pipeline phases. For example, a universal forwarder, a heavy forwarder, or an indexer can perform the input phase.

Data only goes through each phase once, so each configuration belongs on only one component, specifically, the first component in the deployment that handles that phase. For example, say you have data entering the system through a set of universal forwarders, which forward the data to an intermediate heavy forwarder, which then forwards the data onwards to an indexer. In that case, the input phase for that data occurs on the universal forwarders, and the parsing phase occurs on the heavy forwarder.

Data pipeline phase	Components that can perform this role
Input	**indexer** **universal forwarder** **heavy forwarder**
Parsing	indexer heavy forwarder light/universal forwarder (in conjunction with the `INDEXED_EXTRACTIONS` attribute only)
Indexing	indexer
Search	indexer **search head**

Where to set a configuration parameter depends on the components in your specific deployment. For example, you set parsing parameters on the indexers in most cases. But if you have heavy forwarders feeding data to the indexers, you instead set parsing parameters on the heavy forwarders. Similarly, you set search parameters on the search heads, if any. But if you aren't deploying dedicated search heads, you set the search parameters on the indexers.

For more information, see "Components and the data pipeline" in the *Distributed Deployment Manual.*

How configuration parameters correlate to phases of the pipeline

This is a non-exhaustive list of configuration parameters and the pipeline phases that use them. By combining this information with an understanding of which Splunk component in your particular deployment performs each phase, you can determine where to configure each setting.

For example, if you are using universal forwarders to consume inputs, you need to configure `inputs.conf` parameters on the forwarders. If, however, your indexer is directly consuming network inputs, you need to configure those network-related `inputs.conf` parameters on the indexer.

The following items in the phases below are listed in the order Splunk applies them (ie `LINE_BREAKER` occurs before `TRUNCATE`).

Input phase

- inputs.conf
- props.conf
  - `CHARSET`
  - `NO_BINARY_CHECK`
  - `CHECK_METHOD`
  - `CHECK_FOR_HEADER`
  - `PREFIX_SOURCETYPE`
  - sourcetype
- wmi.conf
- regmon-filters.conf

Structured parsing phase

- props.conf
  - `INDEXED_EXTRACTIONS`, and all other structured data header extractions

Parsing phase

- props.conf
  - `LINE_BREAKER`, `TRUNCATE`, `SHOULD_LINEMERGE`, `BREAK_ONLY_BEFORE_DATE`, and all other line merging settings
  - `TIME_PREFIX`, `TIME_FORMAT`, `DATETIME_CONFIG` (datetime.xml), `TZ`, and all other time extraction settings and rules
  - `TRANSFORMS` which includes per-event queue filtering, per-event index assignment, per-event routing
  - `SEDCMD`
  - `MORE_THAN`, `LESS_THAN`
- transforms.conf

    - stanzas referenced by a `TRANSFORMS` clause in props.conf
    - `LOOKAHEAD`, `DEST_KEY`, `WRITE_META`, `DEFAULT_VALUE`, `REPEAT_MATCH`

Indexing phase

- props.conf
    - `SEGMENTATION`
- indexes.conf
- segmenters.conf

Search phase

- props.conf
    - `EXTRACT`
    - `REPORT`
    - `LOOKUP`
    - `KV_MODE`
    - `FIELDALIAS`
    - `EVAL`
    - `rename`
- transforms.conf
    - stanzas referenced by a `REPORT` clause in props.conf
    - `filename`, `external_cmd`, and all other lookup-related settings
    - `FIELDS`, `DELIMS`
    - `MV_ADD`
- lookup files in the lookups folders
- search and lookup scripts in the bin folders
- search commands and lookup scripts
- savedsearches.conf
- eventtypes.conf
- tags.conf
- commands.conf
- alert_actions.conf
- macros.conf
- fields.conf
- transactiontypes.conf
- multikv.conf

Other configuration settings

There are some settings that don't work well in a distributed Splunk environment. These tend to be exceptional and include:

- props.conf

- `CHECK_FOR_HEADER`, `LEARN_MODEL`, `maxDist`. These are created in the parsing phase, but they require generated configurations to be moved to the search phase configuration location.

Back up configuration information

All Splunk's configuration information is contained in **configuration files**. To back up the set of configuration files, make an archive or copy of `$SPLUNK_HOME/etc/`. This directory, along with its subdirectories, contains all the default and custom settings for your Splunk install, and all apps, including saved searches, user accounts, tags, custom source type names, and other configuration information.

Copy this directory to a new Splunk instance to restore. You don't have to stop Splunk to do this.

For more information about configuration files, read "About configuration files".

Back up the cluster master node

If you're using **index replication**, you can back up the master node's static configuration. This is of particular use when configuring a stand-by master that can take over if the primary master fails. For details, see "Configure the master" in the Managing Indexers and Clusters manual.

Check the integrity of your Splunk software files

Most files that Splunk software ships with should not be modified by end users or administrators. However, many users mistakenly modify these files. For example, someone might edit a configuration file in the default directory, or files might be corrupted by hardware flaws, filesystem problems, a mangled installation, or an errant script.

File validation can identify when the contents of the files of a Splunk software instance have been modified in a way that is not valid. You can run this check manually, and it also runs automatically on startup.

Run the check manually

You might want to run the integrity check manually under any of the following conditions:

- You have problems after an upgrade.
- You have symptoms that make you suspect that there may have been a storage system problem.
- You suspect or wish to guard against the common error of edits to the default .conf files.
- As part of a regular system check. See Customize the health check in the *Monitoring Splunk Enterprise* manual.

To run the check manually with default settings, from the installation directory, type `./splunk validate files`. You can manually run the integrity check with two controls.

- You can specify the file describing the correct file contents with `-manifest`. You might want to do this to check against an old manifest from a prior installation after a botched upgrade, to validate that the files are simply stale. You can use any valid manifest file. A manifest file ships in the installation directory with a new Splunk Enterprise download.
- You can constrain the test to only files that end with .conf by using `-type conf`. This is the set of messages the startup-time check prints to the terminal.

Options for automatic verification

The check runs at startup in two parts.

First, as part of the pre-flight check before splunkd starts, the check quickly validates only the default conf files and writes a message to your terminal.

Next, after splunkd starts, the check validates all files shipped with Splunk Enterprise (default conf files, libraries, binaries, data files, and so on). This more complete check writes the results to `splunkd.log` as well as to the bulletin message system in Splunk Web. You can configure it in `limits.conf`.

Options for the second part of the check in `limits.conf` include the following:

- run and log
- run, log, and emit a message to Splunk Web
- disable it

See limits.conf.spec.

Reading all the files provided with the installation has a moderate effect on I/O performance. If you need to restart Splunk software several times in a row, you might wish to disable this check temporarily to improve I/O performance.

Files are validated against the manifest file in the installation directory. If this file is removed or altered, the check cannot work correctly.

Interpret results of an integrity check

If an integrity check returns an error, here are some tips to get you started resolving the problem.

- If the integrity check complains about conf files in default directories, determine how these files became changed and avoid this practice in the future. Modified default conf files will be overwritten on upgrade, creating hard-to-identify problems. See How to edit a configuration file for more details on how to edit configuration files in Splunk software.
- If it complains about files in `$SPLUNK_HOME/bin` or `$SPLUNK_HOME/lib`, or on Windows `%SPLUNK_HOME%\Python2.7\`, you probably need to reinstall. First try to find out how Splunk software was installed locally and determine whether this process could have resulted in a mix of files from different versions. AIX can cause this problem by holding library files open even after the Splunk service has been shut down. On most platforms this type of problem can occur when a Splunk product is upgraded while it is still running. If you cannot determine how this situation occurred, or how to resolve it, work with Splunk Support to identify the issue.
- If it cannot read some files, Splunk software may have been run as two or more different users or security contexts. Files created at install time under one user or context might not be readable by the service now running as another context. Alternatively, you might have legitimately modified the access rules to these files, but this is far less common.
- If the integrity check reports that it cannot read or comprehend the manifest, the manifest might be simply missing from `$SPLUNK_HOME`, or you have access problems to it, or the file may be corrupted. You might want to evaluate whether all the files from the installation package made it to the installation directory, and that the manifest contents are the same as the ones from the package. The manifest is not required for Splunk software to function, but the integrity check cannot function without it.
- If the integrity check reports all or nearly all files are incorrect, splunkd and `etc/splunk.version` might be in disagreement with the rest of the installation. Try to determine how this could have happened. It might be

that the majority of the files are the ones you intended to be present.
- If the pattern is not described above, you might need to apply local analysis and troubleshooting skills possibly in concert with Splunk Support.

Interaction with monitoring console health check

The monitoring console health check queries the `server/status/installed-file-integrity` endpoint. This endpoint is populated with results when the integrity check runs at startup. See server/status/installed-file-integrity in the *REST API Reference Manual*.

If Splunk Enterprise starts with the integrity check disabled in `limits.conf`, then REST file integrity information is not available. In addition, manual runs do not update the results.

See Access and customize health check in *Monitoring Splunk Enterprise*.

Administer Splunk Enterprise with the command line interface (CLI)

About the CLI

You can use the Splunk platform command line interface (CLI) to monitor, configure, and execute searches. The CLI help exists in the product and is accessible through a terminal or shell interface. This topic discusses how to access this information.

Access the CLI

The Splunk platform CLI commands are located in `$SPLUNK_HOME/bin` (or `%SPLUNK_HOME%\bin` on Windows hosts.)

You can find the Splunk installation path on your instance through Splunk Web by clicking **Settings > Server settings > General settings**.

To access the Splunk platform CLI, you need:

- A shell prompt, command prompt, or PowerShell session
- Access to a Splunk platform instance or forwarder, or
- Permission to access the correct port on a remote Splunk platform instance.

CLI help documentation

If you have administrator privileges, you can use the CLI not only to search but also to configure and monitor your Splunk instance or instances. The CLI commands used for configuring and monitoring Splunk are not search commands. Search commands are arguments to the `search` and `dispatch` CLI commands. Some commands require you to authenticate with a username and password or specify a target Splunk server.

You can look up help information for the CLI using:

UNIX	Windows
`./splunk help`	`./splunk help`

For more information about how to access help for specific CLI commands or tasks, see "Get help with the CLI" and "Administrative CLI commands" in this manual.

Work with the CLI on *nix

If you have administrator or root privileges, you can simplify CLI access by adding the top level directory of your Splunk platform installation, `$SPLUNK_HOME/bin`, to your shell path.

This example works for Linux/BSD/Solaris users who installed Splunk Enterprise in the default location:

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

This example works for Mac users who installed Splunk Enterprise in the default location:

```
# export SPLUNK_HOME=/Applications/Splunk
# export PATH=$SPLUNK_HOME/bin:$PATH
```

Now you can invoke CLI commands using:

./splunk <command>

To set the `$SPLUNK_HOME` environment variable while working in a CLI session:

- In *nix: `source /opt/splunk/bin/setSplunkEnv`
- In Windows: `splunk.exe envvars > setSplunkEnv.bat & setSplunkEnv.bat`

Mac OS X requires elevated privileges to access system files or directories

Mac OS X requires superuser level access to run any command that accesses system files or directories. Run CLI commands using **sudo** or "su -" for a new shell as root. The recommended method is to use sudo. (By default the user "root" is not enabled but any administrator user can use sudo.)

Work with the CLI on Windows

To run CLI commands in Splunk Enterprise on Windows, use PowerShell or the command prompt as an administrator.

1. Open a PowerShell window or command prompt as an administrator.
2. Change to the Splunk Enterprise `bin` directory.
3. Run a Splunk command by typing in `splunk` followed by the subcommand and any required arguments.

```
C:\Program Files\Splunk\bin> splunk status
splunkd is running.
splunk helpers are running.
```

You can run many commands and perform many tasks from the CLI. For help on using the CLI, see Get help with the CLI.

Set Splunk environment variables on Windows

You do not need to set Splunk environment variables to use the CLI on Windows. If you want to use environment variables to run CLI commands, you must set the variables manually, because Windows does not set the variables by default.

Set Splunk environment variables temporarily

1. Open a PowerShell window or command prompt.
2. Enter the following command from within either the PowerShell window or command prompt to set environment variables temporarily, or use the Environment Variables dialog box in Computer Properties to set the variables permanently.

PowerShell	Command prompt
`$splunk_home=C:\Program Files\Splunk`	`set SPLUNK_HOME="C:\Program Files\Splunk"`

3. Use the variable to run Splunk commands.

PowerShell	Command prompt
`$splunk_home\bin\splunk status`	`%SPLUNK_HOME%\bin\splunk add forward-server 192.168.1.100:9997 -auth admin:changeme`

Set Splunk environment variables permanently

After you complete this procedure, Windows uses the values you set for the variables until you either change or delete the variable entries.

To set environment variables permanently, see Add or change environment variables on MS TechNet.

Answers

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

Get help with the CLI

This topic discusses how to access Splunk's built-in CLI help reference, which contains information about the CLI commands and how to use them. This topic also briefly discusses the universal parameters, which are parameters that you can use with any CLI command.

Access CLI help reference

If you need to find a CLI command or syntax for a CLI command, use Splunk's built-in CLI help reference.

To start, you can access the default help information with the `help` command:

./splunk help

This will return a list of objects to help you access more specific CLI help topics, such as administrative commands, clustering, forwarding, licensing, searching, etc.

Universal parameters

Some commands require that you authenticate with a username and password, or specify a target host or app. For these commands you can include one of the universal parameters: `auth`, `app`, or `uri`.

```
./splunk [command] [object] [-parameter <value> | <value>]... [-app]
[-owner] [-uri] [-auth]
```

Parameter	Description
app	Specify the App or namespace to run the command; for search, defaults to the Search App.
auth	Specify login credentials to execute commands that require you to be logged in.
owner	Specify the owner/user context associated with an object; if not specified, defaults to the currently logged in user.
uri	Excute a command on any specified (remote) Splunk server.

app

In the CLI, `app` is an object for many commands, such as `create app` or `enable app`. But, it is also a parameter that you can add to a CLI command if you want to run that command on a specific app.

Syntax:

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

For example, when you run a search in the CLI, it defaults to the Search app. If want to run the search in another app:

./splunk search "eventype=error | stats count by source" -deatach f -preview t -app unix

auth

If a CLI command requires authentication, Splunk will prompt you to supply the username and password. You can also use the `-auth` flag to pass this information inline with the command. The `auth` parameter is also useful if you need to run a command that requires different permissions to execute than the currently logged-in user has.

Note: `auth` must be the last parameter specified in a CLI command argument.

Syntax:

```
./splunk command object [-parameter value]... -auth username:password
```

uri

If you want to run a command on a remote Splunk server, use the `-uri` flag to specify the target host.

Syntax:

```
./splunk command object [-parameter value]... -uri specified-server
```

Specify the target Splunk server with the following format:

```
[http|https]://name_of_server:management_port
```

You can specify an IP address for the `name_of_server`. Both IPv4 and IPv6 formats are supported; for example, the `specified-server` may read as: 127.0.0.1:80 or "[2001:db8::1]:80". By default, splunkd listens on IPv4 only. To enable IPv6 support, refer to the instructions in "Configure Splunk for IPv6".

Example: The following example returns search results from the remote "splunkserver" on port 8089.

./splunk search "host=fflanda error 404 *.gif" -auth admin -uri https://splunkserver:8089

For more information about the CLI commands you can run on a remote server, see the next topic in this chapter.

Useful help topics

When you run the default Splunk CLI help, you will see these objects listed.

Administrative CLI commands

You can use the CLI for administrative functions such as adding or editing inputs, updating configuration settings, and searching. If you want to see the list of administrative CLI commands type in:

./splunk help commands

These commands are discussed in more detail in "Administrative CLI commands", the next topic in this manual.

CLI help for clustering

Index replication, which is also referred to as clustering, is a Splunk feature that consists of clusters of indexers configured to replicate data to achieve several goals: data availability, data fidelity, disaster tolerance, and improved search performance.

You can use the CLI to view and edit clustering configurations on the cluster master or cluster peer. For the list of commands and parameters related to clustering, type in:

./splunk help clustering

For more information, read "Configure the cluster with the CLI" in the *Managing Indexers and Clusters* manual.

CLI help for Splunk controls

Use the CLI to start, stop, and restart Splunk server (`splunkd`) and web (`splunkweb`) processes or check to see if the process is running. For the list of controls, type in:

./splunk help controls

For more information, read "Start and stop Splunk" in the Admin Manual.

CLI help for data management

When you add data to Splunk, Splunk processes it and stores it in an **index**. By default, data you feed to Splunk is stored in the **main** index, but you can use the CLI to create and specify other indexes for Splunk to use for different data inputs. To see the list of objects and commands to manage indexes and datastores, type in:

./splunk help datastore
./splunk help index

For more information, read "About managing indexes", "Create custom indexes", and "Remove indexes and data from Splunk" in the *Managing Indexers and Clusters* manual.

CLI help for distributed search deployments

Use the CLI to view and manage your distributed search configurations. For the list of objects and commands, type in:

./splunk help distributed

For information about distributed search, read "About distributed search" in the *Distributed Search* manual.

CLI help for forwarding and receiving

Splunk deployments can include dozens or hundreds of forwarders forwarding data to one or more receivers. Use the CLI to view and manage your data forwarding configuration. For the list of forwarding objects and commands, type in:

./splunk help forwarding

For more information, read "About forwarding and receiving" in the *Forwarding Data* manual.

CLI help for search and real-time search

You can also use the CLI to run both historical and real-time searches. Access the help page about Splunk search and real-time search with:

./splunk help search
./splunk help rtsearch

Also, use objects `search-commands`, `search-fields`, and `search-modifiers` to access the respective help descriptions and syntax:

./splunk help search-commands
./splunk help search-fields
./splunk help search-modifiers

Note: The Splunk CLI interprets spaces as breaks. Use dashes between multiple words for topic names that are more than one word.

To learn more about searching your data with the CLI, refer to "About CLI searches" and "Syntax for CLI searches" in the Search Reference Manual and "Real-time searches and reports in the CLI" in the Search Manual.

Administrative CLI commands

This topic discusses the administrative CLI commands, which are the commands used to manage or configure your Splunk server and distributed deployment.

For information about accessing the CLI and what is covered in the CLI help, see the previous topic, "Get help with the CLI". If you're looking for details about how to run searches from the CLI, refer to "About CLI searches" in the Search Reference Manual.

Your Splunk role configuration dictates what actions (commands) you can execute. Most actions require you to be a Splunk admin. Read more about setting up and managing Splunk users and roles in the "About users and roles" topic in the Admin Manual.

Splunk CLI command syntax

The general syntax for a CLI command is this:

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

Note the following:

- Some commands don't require an object or parameters.
- Some commands have a default parameter that can be specified by its value alone.

Commands, objects, and examples

A **command** is an action that you can perform. An **object** is something you perform an action on.

Command	Objects	Examples
add	exec, forward-server, index, licenser-pools, licenses, master, monitor, oneshot, saved-search, search-server, tcp, udp, user	**1.** Adds monitor directory and file inputs to source `/var/log`. `./splunk add monitor /var/log/`
		2. Adds another master to the list of instances the searchhead searches across.

		`./splunk add cluster-master https://127.0.0.1:8089 -secret testsecret -multisite false'`
anonymize	source	**1.** Replaces identifying data, such as usernames and IP addresses, in the file located at `/tmp/messages`. `./splunk anonymize file -source /tmp/messages`
		2. Anonymizes `Mynames.txt` using name-terms, a file containing a list of common English personal names. `./splunk anonymize file -source /tmp/messages -name_terms $SPLUNK_HOME/bin/Mynames.txt`
apply	cluster-bundle	**1.** Makes validated bundle active on peers. `./splunk apply cluster-bundle`
		2. Skip-validation is an optional argument to skip bundle validation on the master and peers. ./splunk apply cluster-bundle --skip-validation</code>
clean	all, eventdata, globaldata, inputdata, userdata, kvstore	**1.** Removes data from Splunk installation. `eventdata` refers to exported events indexed as raw log files. `./splunk clean eventdata`
		2. `globaldata` refers to host tags and source type aliases. `./splunk clean globaldata`
cmd	btool, classify, locktest, locktool, parsetest, pcregextest, regextest,	**1.** Runs the `splunk btool inputs list` command string with various environment variables set. Run

	searchtest, signtool, walklex	`splunk envvars` to see which environment variables are set. `./splunk cmd btool inputs list`
		2. Shows contents of the bin directory. `./splunk cmd /bin/ls`
create	app	**1.** Builds myNewApp from a template. `./splunk create app myNewApp -template sample_app`
createssl	NONE	
diag	NONE	
disable	app, boot-start, deploy-client, deploy-server, dist-search, index, listen, local-index, maintenance-mode, perfmon, webserver, web-ssl, wmi	**1.** Disables the maintenance mode on peers in indexer clustering. Must be invoked at the master. `'./splunk disable maintenance-mode'`
		2. Disables the logs1 collection. `./splunk disable eventlog logs1`
display	app, boot-start, deploy-client, deploy-server, dist-search, jobs, listen, local-index	**1.** Displays status information, such as enabled/disabled, for all apps. `./splunk display app`
		2. Displays status information for the unix app. `./splunk display app unix`
edit	app, cluster-config, shcluster-config, exec, index, licenser-localslave, licenser-groups, monitor, saved-search,	**1.** Edits the current clustering configuration. `./splunk edit cluster-config -mode slave -site site2`

	search-server, tcp, udp, user	**2.** Edits monitored directory inputs in `/var/log` and only reads from the end of this file. `./splunk edit monitor /var/log -follow-only true`
enable	app, boot-start, deploy-client, deploy-server, dist-search, index, listen, local-index, maintenance-mode, perfmon, webserver, web-ssl, wmi	**1.** Sets the maintenance mode on peers in indexer clustering. Must be invoked at the master. `'./splunk enable maintenance-mode'`
		2. Enables the `col1` collection. `./splunk enable perfmon col1`
export	eventdata, user data	**1.** Exports data out of your Splunk server into `/tmp/apache_raw_404_logs`. `./splunk export eventdata -index my_apache_data -dir /tmp/apache_raw_404_logs -host localhost -terms "404 html"`
fsck	repair, scan, clear-bloomfilter	
help	NONE	
import	userdata	**1.** Imports user accounts data from directory `/tmp/export.dat`. `./splunk import userdata -dir /tmp/export.dat`
install	app	**1.** Installs the app from foo.tar to the local Splunk server. `./splunk install app foo.tar`
		2. Installs the app from foo.tgz to the local Splunk server. `./splunk install app foo.tgz`
list	cluster-buckets, cluster-config,	**1.** Lists all active monitored directory and file inputs. This

	cluster-generation, cluster-peers, deploy-clients, excess-buckets, exec, forward-server, index, inputstatus, licenser-groups, licenser-localslave, licenser-messages, licenser-pools, licenser-slaves, licenser-stacks, licenses, jobs, master-info, monitor, peer-info, peer-buckets, perfmon, saved-search, search-server, tcp, udp, user, wmi	displays files and directories currently or recently monitored by splunkd for change. `./splunk list monitor`
		2. Lists all licenses across all stacks. `./splunk list licenses`
login,logout	NONE	
offline	NONE	**1.** Used to shutdown the peer in a way that does not affect existing searches. The master rearranges the primary peers for buckets, and fixes up the cluster state in case the enforce-counts flag is set. `./splunk offline`
		2. Because the `--enforce-counts` flag is used, the cluster is completely fixed up before this peer is taken down. `./splunk offline --enforce-counts`
package	app	**1.** Packages the stubby app and returns its uri. `./splunk package app stubby`
rebuild	NONE	
refresh	deploy-clients	

reload	ad, auth, deploy-server, index, listen, monitor, registry, script, tcp, udp, perfmon, wmi	**1.** Reloads your deployment server, in entirety or by server class. `./splunk reload deploy-server`
		2. Reloads my_serverclass. `./splunk reload deploy-server -class my_serverclass`
remove	app, cluster-peers, excess-buckets, exec, forward-server, index, jobs, licenser-pools, licenses, monitor, saved-search, search-server, tcp, udp, user	**1.** Removes the cluster master from the list of instances the searchhead searches across. Uses testsecret as the secret/pass4SymmKey. `'./splunk remove cluster-master https://127.0.0.1:8089 -secret testsecret'`
		2. Removes the Unix app. `./splunk remove app unix`
rolling-restart	cluster-peers, shcluster-members	
rtsearch	app, batch, detach, earliest_time, header, id, index_earliest, index_latest, max_time, maxout, output, preview, rt_id, timeout, uri, wrap	**1.** Runs a real-time search that does not line-wrap for individual lines. `./splunk rtsearch 'error' -wrap false`
		2. Runs a real-time search. Use `rtsearch` exactly as you use the traditional search command. `./splunk rtsearch 'eventtype=webaccess error | top clientip'`
search	app, batch, detach, earliest_time, header, id, index_earliest, index_latest, latest_time, max_time, maxout,	**1.** Uses the wildcard as the search object. Triggers an asynchronous search and displays the job id and ttl for the search.

	output, preview, timeout, uri, wrap	`./splunk search '*' -detach true`
		2. Uses `eventtype=webaccess error` as the search object. Does not line wrap for individual lines that are longer than the terminal width. `./splunk search 'eventtype=webaccess error' -wrap 0`
set	datastore-dir, deploy-poll, default-hostname, default-index, minfreemb, servername, server-type, splunkd-port, web-port, kvstore-port	**1.** Sets the force indexing ready bit. `./splunk set indexing-ready`
		2. Sets `bologna:1234` as the deployment server to poll updates from. `./splunk set deploy-poll bologna:1234`
show	config, cluster-bundle-status, datastore-dir, deploy-poll, default-hostname, default-index, jobs, minfreemb, servername, splunkd-port, web-port, kvstore-port	**1.** Shows current logging levels. `./splunk show log-level`
		2. Shows which deployment server Splunk Enterprise is configured to poll from. `./splunk show deploy-poll`
spool	NONE	
start,stop,restart	splunkd, splunkweb	
status	splunkd, splunkweb	
validate	index	**1.** Uses main as the index to validate. Verifies index paths specified in `indexes.conf`. `./splunk validate index main`
version	NONE	

Exporting search results with the CLI

You can use the CLI to export large numbers of search results. For information about how to export search results with the CLI, as well as information about the other export methods offered by Splunk Enterprise, see "Export search results" in the *Search Manual*.

Troubleshooting with the CLI

Splunk's CLI also includes tools that help with troubleshooting Splunk issues. These tools are invoked using the Splunk CLI command `cmd`:

./splunk cmd <tool>

For the list of CLI utilities, see "Command line tools for use with Support" in the Troubleshooting Manual.

Use the CLI to administer a remote Splunk Enterprise instance

You can use the `uri` parameter with any CLI command to send that command to another Splunk Enterprise instance and view the results on your local server.

This topic discusses:

- Syntax for using the `uri` parameter.
- CLI commands that you cannot use remotely.

Note: Remote CLI access is disabled by default for the admin user until you have changed its default password.

Enable remote access

If you are running Splunk Free (which has no login credentials), remote access is disabled by default until you've edited `$SPLUNK_HOME/etc/system/local/server.conf` and set the value:

```
allowRemoteLogin=always
```

Note: The `add oneshot` command works on local instances but cannot be used remotely.

For more information about editing configuration files, refer to About configuration files in this manual.

Send CLI commands to a remote server

The general syntax for using the `uri` parameter with any CLI command is:

```
./splunk command object [-parameter <value>]... -uri <specified-server>
```

The `uri` value, `specified-server` is formatted as:

```
[http|https]://name_of_server:management_port
```

Also, the `name_of_server` can be the fully resolved domain name or the IP address of the remote Splunk Enterprise instance.

Important: This `uri` value is the `mgmtHostPort` value that you defined in `web.conf` on the remote Splunk Enterprise instance. For more information, see the web.conf reference in this manual.

For general information about the CLI, see About the CLI and Get help with the CLI in this manual.

Search a remote instance

The following example returns search results from the remote "splunkserver".

```
./splunk search "host=fflanda error 404 *.gif" -uri
https://splunkserver:8089
```

For details on syntax for searching using the CLI, refer to About CLI searches in the *Search Reference Manual*.

View apps installed on a remote instance

The following example returns the list of apps that are installed on the remote "splunkserver".

```
./splunk display app -uri https://splunkserver:8089
```

Change your default URI value

You can set a default URI value using the SPLUNK_URI environment variable. If you change this value to be the URI of the remote server, you do not need to include the `uri` parameter each time you want to access that remote server.

To change the value of SPLUNK_URI, type either:

```
$ export SPLUNK_URI=[http|https]://name_of_server:management_port     #
For Unix shells
C:\> set SPLUNK_URI=[http|https]://name_of_server:management_port     #
For Windows shell
```

For the examples above, you can change your SPLUNK_URI value by typing:

```
$ export SPLUNK_URI=https://splunkserver:8089
```

CLI commands you cannot run remotely

With the exception of commands that control the server, you can run all CLI commands remotely. These server control commands include:

- Start, stop, restart
- Status, version

You can view all CLI commands by accessing the CLI help reference. See Get help with the CLI in this manual.

Customize the CLI login banner

If you provide CLI access to data, you may need to customize your login banner to notify your users of monitoring, their legal obligations, and penalties for misuse. You can also add additional security (in the form of basic authentication) for your CLI logins.

To create a custom login banner and add basic authentication, add the following stanzas to your local server.conf file:

```
[httpServer]
cliLoginBanner = <string>
allowBasicAuth = true|false
basicAuthRealm = <string>
```

- For `cliLoginBanner = <string>`

Create a message that you want your user to see in the Splunk CLI, such as access policy information, before they are prompted for authentication credentials. The default value is no message.

To create a multi-line banner, place the lines in a comma separated list, putting each line in double-quotes. For example:

```
cliLoginBanner="Line 1","Line 2","Line 3"
```

To include a double quote within the banner text, use two quotes in a row. For example:

```
cliLoginBanner="This is a line that ""contains quote characters""!"
```

- For `allowBasicAuth = true|false`:

Set this value to `true` if you want to require clients to make authenticated requests to the Splunk server using "HTTP Basic" authentication in addition to Splunk's existing (`authtoken`) authentication. This is useful for allowing programmatic access to REST endpoints and for allowing access to the REST API from a web browser. It is not required for the UI or CLI. The default value is `true`.

- For `basicAuthRealm = <string>`:

If you have enabled `allowBasicAuth`, use this attribute to add a text string that can be presented in a Web browser when credentials are prompted. You can display a short message that describes the server and/or access policy. The text: "/splunk" displays by default.

Start Splunk Enterprise and perform initial tasks

Start and stop Splunk Enterprise

This topic provides brief instructions for starting and stopping Splunk Enterprise.

Start Splunk Enterprise on Windows

On Windows, Splunk Enterprise installs by default into `C:\Program Files\Splunk`. Many examples in the Splunk documentation use `$SPLUNK_HOME` to indicate the Splunk installation directory. You can replace the string `$SPLUNK_HOME` (and the Windows variant `%SPLUNK_HOME%`) with `C:\Program Files\Splunk` if you installed Splunk Enterprise into the default directory.

Splunk Enterprise installs with two services, `splunkd` and `splunkweb`. In normal operation, only `splunkd` runs, handling all Splunk Enterprise operations, including the Splunk Web interface. To change this, you must put Splunk Enterprise in legacy mode. Read "Start Splunk Enterprise on Windows in legacy mode."

You can start and stop Splunk on Windows in one of the following ways:

1. Start and stop Splunk Enterprise processes via the Windows Services control panel (accessible from `Start -> Control Panel -> Administrative Tools -> Services`)

- Server daemon and Web interface: `splunkd`
- Web interface (in legacy mode only): `splunkweb`. In normal operation, this service starts, then immediately quits when it receives a start request.

2. Start and stop Splunk Enterprise services from a command prompt by using the `NET START <service>` or `NET STOP <service>` commands:

- Server daemon and Web interface: `splunkd`
- Web interface (in legacy mode only): `splunkweb`. In normal operation, this service starts, then immediately quits when it receives a start request.

3. Start, stop, or restart both processes at once by going to `%SPLUNK_HOME%\bin` and typing

```
> splunk [start|stop|restart]
```

Start Splunk Enterprise on Windows in legacy mode

If you want run Splunk Enterprise in legacy mode, where `splunkd` and `splunkweb` both run, you must change a configuration parameter.

Important: Do not run Splunk Web in legacy mode permanently. Use legacy mode to temporarily work around issues introduced by the new integration of the user interface with the main splunkd service. Once you correct the issues, return Splunk Web to normal mode as soon as possible.

To put Splunk Enterprise in legacy mode:

1. From a command prompt, go to `%SPLUNK_HOME%\etc\system\local`.

2. Edit `%SPLUNK_HOME%\etc\system\local\web.conf`, or create a new file named web.conf in `%SPLUNK_HOME%\etc\system\local` if one does not already exist. See How to edit a configuration file.

3. In `web.conf`, set the `appserverPorts` and `httpport` attributes as follows:

```
[settings]
appServerPorts = 0
httpport = 8000
```

4. Save the file and close it.

5. Restart Splunk Enterprise. The `splunkd` and `splunkweb` services start and remain running.

6. Log into Splunk Enterprise by browsing to `http://<server name>:<httpport>` and entering your credentials.

To restore normal Splunk Enterprise operations, edit `%SPLUNK_HOME%\etc\system\local\web.conf` to remove the `appServerPorts` and `httpport` attributes.

Start Splunk Enterprise on UNIX

Splunk Enterprise installs with one process on *nix, `splunkd`. In normal operation, only `splunkd` runs, handling all Splunk Enterprise operations, including the Splunk Web interface. To change this, you must put Splunk Enterprise in legacy mode. See "Start Splunk Enterprise on Unix in legacy mode."

Start Splunk Enterprise

From a shell prompt on the Splunk Enterprise server host, run this command:

```
# splunk start
```

Note: If you have configured Splunk Enterprise to start at boot time, you should start it using the service command. This ensures that the user configured in the init.d script starts the software.

```
# service splunk start
```

This starts `splunkd` (indexer and the Splunk Web interface).

To start them individually, type:

```
# splunk start splunkd
```

or

(in legacy mode only) `# splunk start splunkweb`

Note: If either the `startwebserver` attribute is disabled, or the `appServerPorts` attribute is set to anything other than 0 in `web.conf`, then manually starting `splunkweb` does not do anything. The `splunkweb` process will not start in either case. See Start Splunk Enterprise on Unix in legacy mode."

To restart Splunk Enterprise (`splunkd` or `splunkweb`) type:

```
# splunk restart
```

```
# splunk restart splunkd
```

(in legacy mode only) `# splunk restart splunkweb`

Start Splunk Enterprise on Unix in legacy mode

If you want run Splunk Enterprise in such a way that `splunkd` and `splunkweb` both run, you must put Splunk Enterprise into legacy mode.

To put Splunk Enterprise in legacy mode:

1. From a shell prompt, go to `$SPLUNK_HOME/etc/system/default`.

2. Make a copy of `web.conf` and place it into `$SPLUNK_HOME/etc/system/local`.

3. Edit `web.conf` in `$SPLUNK_HOME/etc/system/local`.

4. In `web.conf`, set the `appserverPorts` and `httpport` attributes as follows:

```
[settings]
appServerPorts = 0
httpport = 8000
```

5. Save the file and close it.

6. Restart Splunk Enterprise (see "Start Splunk Enterprise on Unix"). The `splunkd` and `splunkweb` services start and remain running.

7. Log into Splunk Enterprise by browsing to `http://<server name>:<httpport>` and entering your credentials.

To restore normal Splunk Enterprise operations: edit `%SPLUNK_HOME%\etc\system\local\web.conf` and remove the `appServerPorts` and `httpport` attributes.

Stop Splunk Enterprise

To shut down Splunk Enterprise, run this command:

```
# splunk stop
```

To stop `splunkd` and Splunk Web individually, type:

```
# splunk stop splunkd
```

or

(in legacy mode only) `# splunk stop splunkweb`

Check if Splunk is running

To check if Splunk Enterprise is running, type this command at the shell prompt on the server host:

```
# splunk status
```

You should see this output:

```
splunkd is running (PID: 3162).
splunk helpers are running (PIDs: 3164).
```
If Splunk Enterprise runs in legacy mode, you will see an additional line in the output:

```
splunkweb is running (PID: 3216).
```
Note: On Unix systems, you must be logged in as the user who runs Splunk Enterprise to run the `splunk status` command. Other users cannot read the necessary files to report status correctly.

If `splunk status` decides that the service is running it will return the status code 0, or success. If `splunk status` determines that the service is not running it will return the Linux Standard Base value for a non-running service, 3. Other values likely indicate `splunk status` has encountered an error.

You can also use `ps` to check for running Splunk Enterprise processes:

```
# ps aux | grep splunk | grep -v grep
```

Solaris users should use the `-ef` arguments to `ps` instead of `aux`:

```
# ps -ef | grep splunk | grep -v grep
```

Restart Splunk Enterprise from Splunk Web

You can also restart Splunk from Splunk Web:

1. Navigate to **System > Server controls**.

2. Click **Restart Splunk**.

This will restart the `splunkd` and (in legacy mode only) the `splunkweb` processes.

Configure Splunk to start at boot time

On Windows, Splunk starts by default at machine startup. To disable this, see "Disable boot-start on Windows" at the end of this topic.

On *nix platforms, you must configure Splunk to start at boot time.

Enable boot-start on *nix platforms

Splunk provides a utility that updates your system boot configuration so that Splunk starts when the system boots up. This utility creates a suitable `init` script (or makes a similar configuration change, depending on your OS).

As root, run:

```
$SPLUNK_HOME/bin/splunk enable boot-start
```

If you don't start Splunk as root, you can pass in the `-user` parameter to specify which user to start Splunk as. For example, if Splunk runs as the user bob, then as root you would run:

```
$SPLUNK_HOME/bin/splunk enable boot-start -user bob
```

If you want to stop Splunk from running at system startup time, run:

```
$SPLUNK_HOME/bin/splunk disable boot-start
```

More information is available in `$SPLUNK_HOME/etc/init.d/README` and if you type `help boot-start` from the command line.

Note for Mac users

Splunk automatically creates a script and configuration file in the directory: `/System/Library/StartupItems`. This script is run at system start, and automatically stops Splunk at system shutdown.

Note: If you are using a Mac OS, you **must** have root level permissions (or use **sudo**). You need administrator access to use **sudo**.

Example:

Enable Splunk to start at system start up on Mac OS using:

just the CLI:

```
./splunk enable boot-start
```

the CLI with **sudo**:

```
sudo ./splunk enable boot-start
```

Disable boot-start on Windows

By default, Splunk starts automatically when you start your Windows machine. You can configure the Splunk processes (`splunkd` and `splunkweb`) to start manually from the Windows Services control panel.

Install your license

The first time you download Splunk, you are asked to register.

Your registration authorizes you to receive a temporary (60 day) Enterprise trial license, which allows a maximum indexing volume of 500 MB/day. This license is included with your download.

The Enterprise license enables the following features:

- Multiple user accounts and access controls.
- Distributed search and data routing.
- Deployment management.

For more information about Splunk licensing, read How Splunk licensing works in this manual.

Where is your new license?

When you request a new license, you should receive the license in an email from Splunk. You can also access that new license in your `splunk.com` My Orders page.

To install and update your licenses via Splunk Web, navigate to **Settings > Licensing** and follow these instructions.

Change default values

Before you begin configuring Splunk Enterprise for your environment, check through the following default settings.

Set or change environment variables

You can change how Splunk Enterprise starts by setting environment variables on your operating system.

On *nix, use the `setenv` or `export` commands to set a particular variable. For example:

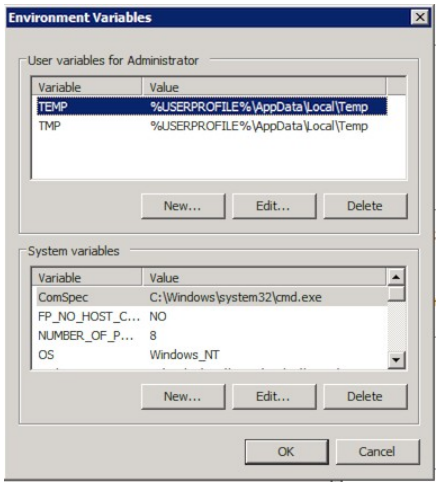
```
# export SPLUNK_HOME = /opt/splunk02/splunk
```

To set the environment permanently, edit the appropriate shell initialization file. Add entries for the variables you want Splunk Enterprise to use when it starts up.

On Windows, use the `set` environment variable in either a command prompt or PowerShell window:

```
C:\> set SPLUNK_HOME = "C:\Program Files\Splunk"
```

To set the environment permanently, use the "Environment Variables" window to add the entry to the "User variables" list.



There are several environment variables that are available:

Environment variable	Purpose
`SPLUNK_HOME`	The fully qualified path to the Splunk Enterprise

	installation directory.
SPLUNK_DB	The fully qualified path to the directory that contains the Splunk Enterprise index directories.
SPLUNK_BINDIP	The IP address on the system that Splunk Enterprise should bind to on startup to accept connections. Useful for when a host has more than one live IP address.
SPLUNK_IGNORE_SELINUX	Tells Splunk Enterprise to attempt to start when running in Linux host with SELinux enabled. By default, Splunk Enterprise quits immediately when it detects that SELinux is active. This variable defeats that check and can be used in scenarios where you have configured SELinux to allow Splunk Enterprise to work.
SPLUNK_OS_USER	Tells Splunk Enterprise to assume the credentials of the user you specify, regardless of what user you started it as. For example, if you specify the user 'splunk' on your system and start Splunk Enterprise as root, it adopts the privileges of the 'splunk' user and any files written by those processes will be owned by the 'splunk' user.
SPLUNK_SERVER_NAME	The name of the splunkd service (on Windows) or process (on *nix). Do not set this variable unless you know what you are doing.
SPLUNK_WEB_NAME	The name of the splunkweb service (on Windows) or process (on *nix). Do not set this variable unless you know what you are doing.

You can also edit these environment variables for each instance by editing `splunk-launch.conf` or, in some cases, `web.conf`. This is handy when you run more than one Splunk software instance on a host. See "splunk-launch.conf".

Change the admin default password

Splunk Enterprise has a default administration account and password, admin/changeme. Splunk recommends strongly that you change the default in order to keep your system secure. Your password should be complex and follow general password best practices:

- Use a combination of words, numbers, symbols, and both upper- and lower-case letters.

- Complexity is important, but length is vital. We recommend a minimum of 10 characters.
- Do not choose passwords based upon details that might not be confidential, such as your birth date, your Social Security or phone number, or names of family members.
- Do not use words that can be found in the dictionary.
- Do not use a password you use or have used elsewhere.

Change the password using Splunk Web

To change the admin default password:

1. Log into Splunk Web as the admin user.
2. Click **Settings** in the top-right of the interface.
3. Click **Access controls** in the Users and Authentication section of the screen.
4. Click **Users**.
5. Click the **admin** user.
6. Update the password, and click **Save**.

Change the password using the CLI

The Splunk CLI command is:

```
splunk edit user
```

Important: You must authenticate with the existing password before you can change it. Log into Splunk Enterprise via the CLI or use the `-auth` parameter. For example, this command changes the admin password from *changeme* to *foo*:

```
splunk edit user admin -password foo -role admin -auth admin:changeme
```

Note: On *nix operating systems, the shell interprets some special characters as command directives. You must either escape these characters by preceding them with `\` individually, or enclose the password in single quotes (`'`). For example:

```
splunk edit user admin -password 'FFL14io!23ur$' -role admin -auth
admin:changeme
```

or

```
splunk edit user admin -password FFL14io!23ur\$ -role admin -auth
admin:changeme
```

On Windows, use the caret (^) to escape reserved shell characters, or enclose the password in double-quotes ("). For example:

```
splunk edit user admin -password "FFL14io!23ur>" -role admin -auth
admin:changeme
```

or

```
splunk edit user admin -password FFL14io!23ur^> -role admin -auth
admin:changeme
```

Note: You can also reset all of your passwords across servers at once. See "Deploy secure passwords across multiple servers for the procedure.

Change network ports

Splunk Enterprise configures a few ports at installation time:

- **The HTTP/HTTPS port.** This port provides the socket for Splunk Web. It defaults to 8000.
- **The appserver port.** 8065 by default.
- **The management port.** This port is used to communicate with the `splunkd` daemon. Splunk Web talks to `splunkd` on this port, as does the command line interface and any distributed connections from other servers. This port defaults to 8089.
- **The KV store port.** 8191 by default.

Important: During installation, you might have set these ports to values other than the defaults.

Note: Splunk instances **receiving** data from **forwarders** must be configured with an additional port, the receiver port. They use this port to listen for incoming data from forwarders. This configuration does not occur during installation. The default receiver port is 9997. For more information, see "Enable a receiver" in the Forwarding Data Manual.

Use Splunk Web

To change the ports from their installation settings:

1. Log into Splunk Web as the admin user.
2. Click **Settings** in the top-right of the interface.
3. Click the **Server settings** link in the System section of the screen.
4. Click **General settings**.

5. Change the value for either **Management port** or **Web port**, and click **Save**.

Use Splunk CLI

To change the port settings via the Splunk CLI, use the CLI command `set`. For example, this command sets the Splunk Web port to 9000:

```
splunk set  web-port 9000
```

This command sets the splunkd port to 9089:

```
splunk set  splunkd-port 9089
```

Change the default Splunk server name

The Splunk server name setting controls both the name displayed within Splunk Web and the name sent to other Splunk Servers in a distributed setting.

The default name is taken from either the DNS or IP address of the Splunk Server host.

Use Splunk Web

To change the Splunk server name:

1. Log into Splunk Web as the admin user.
2. Click **Settings** in the top-right of the interface.
3. Click the **Server settings** link in the System section of the screen.
4. Click **General settings**.
5. Change the value for **Splunk server name**, and click **Save**.

Use Splunk CLI

To change the server name via the CLI, use the `set servername` command. For example, this command sets the server name to *foo*:

```
splunk set servername foo
```

Changing the datastore location

The datastore is the top-level directory where the Splunk Server stores all indexed data.

Note: If you change this directory, the server does not migrate old datastore files. Instead, it starts over again at the new location.

To migrate your data to another directory follow the instructions in "Move an index".

Use Splunk Web

To change the datastore location:

1. Log into Splunk Web as the admin user.
2. Click **Settings** in the top-right of the interface.
3. Click the **System settings** link in the System section of the screen.
4. Click **General settings**.
5. Change the path in **Path to indexes**, and click **Save**.
6. Use the CLI to restart Splunk Enterprise. Navigate to `$SPLUNK_HOME/bin/` (*nix) or `%SPLUNK_HOME%\bin` (Windows) and run this command:

```
splunk restart
```

Important: Do not use the restart function inside Settings. This will not have the intended effect of causing the index directory to change. You *must* restart from the CLI.

Use Splunk CLI

To change the datastore directory via the CLI, use the `set datastore-dir` command. For example, this command sets the datastore directory to `/var/splunk/`:

```
splunk set datastore-dir /var/splunk/
```

Set minimum free disk space

The minimum free disk space setting controls how low disk space in the datastore location can fall before Splunk software stops indexing.

Splunk software resumes indexing when more space becomes available.

Use Splunk Web

To set minimum free disk space:

1. Log into Splunk Web as the admin user.
2. Click **Settings** in the top-right of the interface.
3. Click the **System settings** link in the System section of the screen.
4. Click **General settings**.
5. Change the value for **Pause indexing if free disk space falls below**, and click **Save**.

Use Splunk CLI

To change the minimum free space value via the CLI, use the `set minfreemb` command. For example, this command sets the minimum free space to 2000 MB:

```
splunk set minfreemb 2000
```

Set the default time range

The default time range for ad hoc searches in the Search & Reporting App is set to **All time**. An administrator can set the default time range globally, across all apps. The setting is stored in `SPLUNK_HOME/etc/apps/user-prefs/local/user-prefs.conf` file in the `[general_default]` stanza.

This setting applies to all Search pages in Splunk Apps, not just the Search & Reporting App. This setting applies to all user roles.

Note: This setting does not apply to dashboards.

Use Splunk Web

1. Log into Splunk Web as the admin user.

2. Click **Settings**.

3. In the System section, click **Server settings**.

4. Click **Search Preferences**.

5. From the **Default search time range** drop-down, select the time that you want to use and click **Save**.

Time range settings in the `ui_prefs.conf` file

You might already have a time range setting in the `ui-prefs.conf` file for a specific application or user. The settings in the `ui-prefs.conf` file take precedence over any settings that you make to the global default time range using Splunk Web.

However, if you want to use the global default time range for all users and applications, consider removing the settings you have in the `ui-prefs.conf` file.

Other default settings

The Splunk Web Settings General Settings screen has a few other default settings that you might want to change. Explore the screen to see the range of options.

See also

About configuration files

user-prefs.conf

ui-prefs.conf

Bind Splunk to an IP

You can force Splunk to bind its ports to a specified IP address. By default, Splunk will bind to the IP address 0.0.0.0, meaning all available IP addresses.

Changing Splunk's bind IP only applies to the Splunk daemon (splunkd), which listens on:

- TCP port 8089 (by default)
- any port that has been configured as for:
    - SplunkTCP inputs
    - TCP or UDP inputs

To bind the Splunk Web process (splunkweb) to a specific IP, use the `server.socket_host` setting in web.conf.

Temporarily

To make this a temporary change, set the environment variable `SPLUNK_BINDIP=<ipaddress>` before starting Splunk.

Permanently

If you want this to be a permanent change in your working environment, modify `$SPLUNK_HOME/etc/splunk-launch.conf` to include the `SPLUNK_BINDIP` attribute and `<ipaddress>` value. For example, to bind Splunk ports to 127.0.0.1 (for local loopback only), `splunk-launch.conf` should read:

```
# Modify the following line to suit the location of your Splunk install.
# If unset, Splunk will use the parent of the directory this
configuration
# file was found in
#
# SPLUNK_HOME=/opt/splunk
SPLUNK_BINDIP=127.0.0.1
```

Important: The `mgmtHostPort` attribute in `web.conf` has a default value of `127.0.0.1:8089`. Therefore, if you change `SPLUNK_BINDIP` to any value besides `127.0.0.1`, you must also change `mgmtHostPort` to use the same IP address. For example, if you make this change in `splunk-launch.conf`:

```
SPLUNK_BINDIP=10.10.10.1
```

you must also make this change in `web.conf` (assuming the management port is `8089`):

```
mgmtHostPort=10.10.10.1:8089
```

See web.conf for more information on the `mgmtHostPort` attribute.

IPv6 considerations

Starting in version 4.3, the `web.conf mgmtHostPort` setting has been extended to allow it to take IPv6 addresses if they are enclosed in square brackets. Therefore, if you configure splunkd to only listen on IPv6 (via the setting in `server.conf` described in "Configure Splunk for IPv6" in this manual), you must change this from `127.0.0.1:8089` to `[::1]:8089`.

Configure Splunk for IPv6

This topic discusses Splunk's support for IPv6 and how to configure it. Before following the procedures in this topic, you may want to review:

- "About configuration files" in this manual to learn about how Splunk's configuration files work
- "Get data from TCP and UDP ports" in the Getting Data In manual
- "server.conf" in this manual to see the reference of options available in the `server.conf` configuration file
- "inputs.conf" in this manual to see the reference of options available in the `inputs.conf` configuration file

Starting in version 4.3, Splunk supports IPv6. Users can connect to Splunk Web, use the CLI, and forward data over IPv6 networks.

IPv6 platform support

All Splunk-supported OS platforms (as described in "Supported OSes" in the Installation Manual) are supported for use with IPv6 configurations except for the following:

- HPUX PA-RISC
- Solaris 8, and 9
- AIX

Configure Splunk to listen on an IPv6 network

You have a few options when configuring Splunk to listen over IPv6. You can configure Splunk to:

- connect to IPv6 addresses only and ignore all IPv4 results from DNS
- connect to both IPv4 and IPv6 addresses and
    - try the IPv6 address first
    - try the IPv4 address first
- connect to IPv4 addresses only and ignore all IPv6 results from DNS

To configure how Splunk listens on IPv6: Edit a copy of `server.conf` in `$SPLUNK_HOME/etc/system/local` to add the following:

```
listenOnIPv6=[yes|no|only]
```

- `yes` means that splunkd will listen for connections from both IPv6 and IPv4.
- `no` means that splunkd will listen on IPv4 only, **this is the default setting**.
- `only` means that Splunk will listen for incoming connections on IPv6 only.

`connectUsingIpVersion=[4-first|6-first|4-only|6-only|auto]`

- `4-first` means splunkd will try to connect to the IPv4 address first and if that fails, try IPv6.
- `6-first` is the reverse of `4-first`. This is the policy most IPv6-enabled client apps like web browsers take, but can be less robust in the early stages of IPv6 deployment.
- `4-only` means that splunkd will ignore any IPv6 results from DNS.
- `6-only` means that splunkd will Ignore any IPv4 results from DNS.
- `auto` means that splunkd picks a reasonable policy based on the setting of `listenOnIPv6`. **This is the default value**.
    - If splunkd is listening only on IPv4, this behaves as though you specified `4-only`.
    - If splunkd is listening only on IPv6, this behaves as though you specified `6-only`.
    - If splunkd is listening on both, this behaves as though you specified `6-first`.

Important: These settings only affect DNS lookups. For example, a setting of `connectUsingIpVersion = 6-first` will not prevent a stanza with an explicit IPv4 address (like "server=10.1.2.3:9001") from working.

If you have just a few inputs and don't want to enable IPv6 for your entire deployment

If you've just got a few data sources coming over IPv6 but don't want to enable it for your entire Splunk deployment, you can add the `listenOnIPv6` setting described above to any `[udp]`, `[tcp]`, `[tcp-ssl]`, `[splunktcp]`, or `[splunktcp-ssl]` stanza in `inputs.conf`. This overrides the setting of the same name in `server.conf` for that particular input.

Forwarding data over IPv6

Your Splunk **forwarders** can forward over IPv6; the following are supported in `outputs.conf`:

- The `server` setting in `[tcpout]` stanzas can include IPv6 addresses in the standard `[host]:port` format.

- The `[tcpout-server]` stanza can take an IPv6 address in the standard `[host]:port` format.
- The `server` setting in `[syslog]` stanzas can include IPv6 addresses in the standard `[host]:port` format.

Distributed search configuration for IPv6

Your Splunk **distributed search** deployment can use IPv6; the following are supported in `distsearch.conf`:

- The `servers` setting can include IPv6 addresses in the standard `[host]:port` format
- However, `heartbeatMcastAddr` has not been updated to support IPv6 addresses; this setting is deprecated in Splunk 4.3 and will be removed from the product in a future release.

Access to Splunk Web over IPv6

If your network policy allows or requires IPv6 connections from web browsers, you can configure the splunkweb service to behave differently than splunkd. Starting in 4.3, `web.conf` supports a `listenOnIPv6` setting. This setting behaves exactly like the one in `server.conf` described above, but applies only to Splunk Web.

The existing `web.conf mgmtHostPort` setting has been extended to allow it to take IPv6 addresses if they are enclosed in square brackets. Therefore, if you configure splunkd to only listen on IPv6 (via the setting in `server.conf` described above), you must change this from `127.0.0.1:8089` to `[::1]:8089`.

The Splunk CLI and IPv6

The Splunk CLI can communicate to splunkd over IPv6. This works if you have set `mgmtHostPort` in `web.conf`, defined the `$SPLUNK_URI` environment variable, or use the `-uri` command line option. When using the `-uri` option, be sure to enclose IPv6 IP address in brackets and the entire address and port in quotes, for example: `-uri "[2001:db8::1]:80"`.

IPv6 and SSO

If you are using IPv6 with SSO, you do not use the square bracket notation for the `trustedIP` property, as shown in the example below. This applies to both `web.conf` and `server.conf`.

In the following `web.conf` example, the `mgmtHostPort` attribute uses the square bracket notation, but the `trustedIP` attribute does not:

```
[settings]
mgmtHostPort = [::1]:8089
startwebserver = 1
listenOnIPv6=yes
trustedIP=2620:70:8000:c205:250:56ff:fe92:1c7,::1,2620:70:8000:c205::129
SSOMode = strict
remoteUser = X-Remote-User
tools.proxy.on = true
```

For more information on SSO, see "Configure Single Sign-on" in the Securing Splunk Enterprise manual.

Secure your configuration

If you haven't already, this is a good time to make sure that Splunk and your data are secure. Taking the proper steps to secure Splunk reduces the attack surface and mitigates the risk and impact of most vulnerabilities.

Some key actions you should take after installation:

- Set up users and roles. You can configure users using Splunks native authentication and/or use LDAP to manage users. See About user authentication
- Set up certificate authentication (SSL). Splunk ships with a set of default certificates that should be replaced for secure authentication. We provide guidelines and further instructions for adding SSL encryption and authentication and Configure secure authentication.

The Securing Splunk Enterprise manual provides more information about ways you can secure Splunk. Including a checklist for hardening your configuration. See Securing Splunk Enterprise for more information.

Share performance data

You can opt in to automatically share certain data about your license usage and deployment performance with Splunk, Inc. Splunk uses this data to make decisions about future product development, and will not share your information with any third parties.

Opt in or out

You can choose to send both, either, or neither of two types of data:

- **License usage data** describing your active licenses and the amount of data you index,
- **Anonymized usage data** about your deployment performance.

The first time you run Splunk Web on a search head as an admin or equivalent, you are presented with a modal. The options on the modal are as follows:

- Click **Skip** to suppress the modal permanently for the user who clicks **Skip**. Use this option to defer the decision to a different admin.
- Click **OK** to suppress the modal permanently for all users.

You can also opt in or out at any time by navigating to **Settings > Instrumentation**.

If you opt out, the searches that gather the data on your system do not run, and no data is sent.

The ability to enable or disable instrumentation is controlled by the `edit_telemetry_settings` capability.

What data is collected

For either type of data, you can view what data has been sent in Splunk Web.

1. Navigate to **Settings > Instrumentation**.
2. Under the relevant data category ("Anonymized usage data" or "License usage data"), click **View Log**.
3. Click **View Data**.

This log of data is available only after the first run of the collection (see Feature footprint). To inspect the type of data that gets sent before opting in on your production environment, you can opt in on your sandbox environment.

Anonymized usage data is not tied to customer accounts, and is used only in aggregate for analysis. Note that anonymized usage data is not encrypted when it is collected. Data received is securely stored within on-premise servers at Splunk, with access restricted to aggregate analyses only. License IDs collected

are used only to verify that data is received from a valid Splunk product, and to help analyze how different Splunk products are being deployed across the population of users.

The following table describes the data collected if you opt in to both programs. The data is in JSON format tagged with a field named "component."

Description	Component(s)	Note
Active license group and subgroup	`licensing.stack`	
Total license stack quota, total license pool consumption, license stack type	`licensing.stack`	
License pool quota, license pool consumption	`licensing.stack`	
License IDs	`licensing.stack`	Always sent, but persisted only for users opting in to license usage reporting.
Number of nodes in indexer cluster, replication factor and search factor for indexer cluster	`deployment.clustering.indexer`	
GUID, host, Number of cores by type (virtual/physical), CPU architecture, memory size, storage (partition) capacity, OS/version, Splunk version	`deployment.node`	For each indexer or search head
Number of hosts, Number of Splunk software instances, OS/version, CPU architecture, Splunk software version, distribution of forwarding volume	`deployment.forwarders`	For forwarders

Core utilization, storage utilization, memory usage, indexing throughput, search latency	`deployment.node`
	`performance.indexing`
	`performance.search`

Indexing volume, Number of events, Number of hosts, source type name	`usage.indexing.sourcetype`
Number of active users	`usage.users.active`
Number of searches of each type, distribution of concurrent searches	`usage.search.type` `usage.search.concurrent`
App name, page name, locale, Number of users, Number of page loads	`usage.app.page`

Data samples

Click **Expand** to view examples of the data that is collected.

Component	Data category	Example

`deployment.clustering.indexer`	Clustering configuration	<pre>{
		"host": "docteam-unix-5",
		"summaryReplication": true,
		"siteReplicationFactor": null,
		"enabled": true,
		"multiSite": false,
		"searchFactor": 2,
		"siteSearchFactor": null,
		"timezone": "-0700",
		"replicationFactor": 3
		}</pre>

`deployment.forwarders`	Forwarder architecture, forwarding volume	<pre>{
		"hosts": 168,
		"instances": 497,
		"architecture": "x86_64",
		"os": "Linux",
		"splunkVersion": "6.5.0",
		"type": "uf",
		"kb": {
		"min": 389,
		"max": 2291497,</pre>

		"total": 189124803, "p10": 40960, "p20": 139264, "p30": 216064, "p40": 269312, "p50": 318157, "p60": 345088, "p70": 393216, "p80": 489472, "p90": 781312 } }

deployment.node	Host architecture, utilization	{
		"guid": "123309CB-ABCD-4BB9-9B6A-185316600F23",
		"host": "docteam-unix-3",
		"os": "Linux",
		"osExt": "Linux",
		"osVersion": "3.10.0-123.el7.x86_64",
		"splunkVersion": "6.5.0",
		"cpu": {
		"coreCount": 2,
		"utilization": {
		"min": 0.01,
		"p10": 0.01,
		"p20": 0.01,
		"p30": 0.01,
		"p40": 0.01,
		"p50": 0.02,
		"p60": 0.02,
		"p70": 0.03,
		"p80": 0.03,
		"p90": 0.05,
		"max": 0.44
		},
		"virtualCoreCount": 2,
		"architecture": "x86_64"
		},
		"memory": {
		"utilization": {
		"min": 0.26,
		"max": 0.34,
		"p10": 0.27,
		"p20": 0.28,
		"p30": 0.28,
		"p40": 0.28,
		"p50": 0.29,
		"p60": 0.29,
		"p70": 0.29,
		"p80": 0.3,
		"p90": 0.31

		<pre>{
		        },
		        "capacity": 3977003401
		    },
		    "disk": {
		        "fileSystem": "xfs",
		        "capacity": 124014034944,
		        "utilization": 0.12
		    }
		}</pre>

licensing.stack	Licensing quota and consumption	<pre>{
		    "type": "download-trial",
		    "guid":
		"4F735357-F278-4AD2-BBAB-139A85A75DBB",
		    "product": "enterprise",
		    "name": "download-trial",
		    "licenseIDs": [
		        "553A0D4F-3B7B-4AD5-B241-89B94386A0
		    ],
		    "quota": 524288000,
		    "pools": [
		        {
		            "quota": 524288000,
		            "consumption": 304049405
		        }
		    ],
		    "consumption": 304049405,
		    "subgroup": "Production",
		    "host": "docteam-unix-9"
		}</pre>

performance.indexing	Indexing throughput and volume	<pre>{
		    "host": "docteam-unix-5",
		    "thruput": {
		        "min": 412,
		        "max": 9225,
		        "total": 42980219,
		        "p10": 413,
		        "p20": 413,
		        "p30": 431,
		        "p40": 450,
		        "p50": 474,
		        "p60": 488,
		        "p70": 488,
		        "p80": 488,
		        "p90": 518
		    }
		}</pre>

performance.search	Search runtime statistics	<pre>{
		    "latency": {
		        "min": 0.01,
		        "max": 1.33,</pre>

		"p10": 0.02, "p20": 0.02, "p30": 0.05, "p40": 0.16, "p50": 0.17, "p60": 0.2, "p70": 0.26, "p80": 0.34, "p90": 0.8 } }

usage.app.page	App page users and views	{
		"app": "search",
		"locale": "en-US",
		"occurrences": 1,
		"page": "datasets",
		"users": 1
		}

usage.indexing.sourcetype	Indexing by source type	{
		"name": "vendor_sales",
		"bytes": 2026348,
		"events": 30245,
		"hosts:" 1
		}

usage.search.concurrent	Search concurrency	{
		"host": "docteam-unix-5"
		"searches": {
		"min": 1,
		"max": 11,
		"p10": 1,
		"p20": 1,
		"p30": 1,
		"p40": 1,
		"p50": 1,
		"p60": 1,
		"p70": 1,
		"p80": 2,
		"p90": 3
		}
		}

usage.search.type	Searches by type	{
		"ad-hoc": 1428,
		"scheduled": 225
		}

usage.users.active	Active users	{
		"active": 23
		}

What data is not collected

The following kinds of data are not collected:

- Usernames or passwords.
- Indexed data that you ingest into your Splunk platform instance.

Why send license usage data

Certain license programs require that you report your license usage. The easiest way to do this is to opt in to automatically send this information to Splunk.

If you do not opt in to automatic license data sharing, you can send this data manually. In Splunk Web navigate to **Settings > Instrumentation** and follow the instructions for exporting the data to your local directory.

Feature footprint

The data is summarized and sent once per day, starting at 3:05 AM.

About searches

If you opt in, one instance in your Splunk Enterprise deployment collects data through ad hoc searches. All searches run in sequence, starting at 3:05 AM. All searches are triggered with a scripted input. See Configure the priority of scheduled reports.

Which node runs the searches

Only one node in your deployment runs the searches to collect the usage data. Which instance that is depends on the details of your deployment:

- In an indexer clustering environment, the searches run on the cluster master.
- If search head clustering is enabled but not indexer clustering, the searches run on the search head captain.
- If your deployment does not use clustering, the searches run on a search head.

About internal log files

If you enable license usage reporting, the first time product instrumentation runs, it creates a new file in `$SPLUNK_HOME/var/log/splunk`. The file is called `license_usage_summary.log` and is limited in size to 25 MB. The file is indexed to a new internal index, `_telemetry`. The `_telemetry` index is retained for two years by default and is limited in size to 256 MB.

After the searches run, the data is packaged and sent to Splunk, Inc.

The app resides in the filesystem at `$SPLUNK_HOME/etc/apps/splunk_instrumentation`.

Configure Splunk licenses

How Splunk Enterprise licensing works

Splunk Enterprise takes in data from sources you designate and processes it so that you can analyze it. We call this process indexing. For information about the exact indexing process, see How Splunk software handles your data in *Getting Data In*.

Splunk Enterprise licenses specify how much data you can index per calendar day (from midnight to midnight by the clock on the **license master**).

Any host in your Splunk Enterprise infrastructure that performs indexing must be licensed to do so. You can either run a standalone indexer with a license installed locally, or you can configure one of your Splunk Enterprise instances as a **license master** and set up a **license pool** from which other indexers, configured as **license slaves**, can draw.

In addition to indexing volume, access to some Splunk Enterprise features requires an Enterprise license. For more information about different types of licenses, read Types of Splunk licenses.

About the connection between the license master and license slaves

When a license master instance is configured, and license slaves are added to it, the license slaves communicate their usage to the license master every minute. If the license master is unreachable for any reason, the license slave starts a 72 hour timer. If the license slave cannot reach the license master for 72 hours, search is blocked on the license slave (although indexing continues). Users cannot search data in the indexes on the license slave until that slave can reach the license master again.

Splunk Enterprise license lifecycle

When you first install a downloaded copy of Splunk Enterprise, that instance uses a 60 day Enterprise Trial license. This license allows you to try out all of the features in Splunk Enterprise for 60 days, and to index up to 500 MB of data per day.

Once the 60 day trial expires (and if you have not purchased and installed an Enterprise license), you are given the option to switch to Splunk Free. Splunk Free includes a subset of the features of Splunk Enterprise and is intended for use in standalone deployments and for short-term forensic investigations. It allows you to index up to 500 MB of data a day indefinitely.

Important: Splunk Free does not include authentication, scheduled searches, or alerting. This means that any user accessing your installation (via Splunk Web or the CLI) will not have to provide credentials. Additionally, scheduled saved searches or alerts will no longer fire.

If you want to continue using Splunk Enterprise features after the 60 day Trial expires, you must purchase an Enterprise license. Contact a Splunk sales rep to learn more.

Once you have purchased and downloaded an Enterprise license, you can install it on your instance and access Splunk Enterprise features. Read "Types of Splunk licenses" in this manual for information about Enterprise features.

For information about upgrading an existing license, see Migrate to the new Splunk Enterprise licenser in the *Installation Manual*.

Types of Splunk software licenses

Each Splunk software instance requires a license. Splunk licenses specify how much data a given Splunk platform instance can index and what features you have access to. This topic discusses the various license types and options.

In general, there are a few types of licenses:

- The Enterprise license enables all Enterprise features, such as authentication and distributed search. As of Splunk Enterprise 6.5.0, new Enterprise licenses are no-enforcement licenses.
- The Free license allows you a limited indexing volume and disables authentication, but is perpetual.
- The Forwarder license allows you to forward, but not index, data and enables authentication.
- The Beta license typically enables Enterprise features, but is restricted to Splunk Beta releases.
- A license for a premium app is used in conjunction with an Enterprise or Cloud license to access an app's functionality.

Also discussed in this topic are licensing considerations for a deployment including distributed search or indexer clustering.

For information about upgrading a pre-4.2 license, see Migrate to the new Splunk Enterprise licenser in the *Installation Manual*.

Splunk Enterprise licenses

Splunk Enterprise is the standard Splunk software license. It allows you to use all Splunk Enterprise features, including authentication, distributed search, deployment management, scheduling of alerts, and role-based access controls. Enterprise licenses are available for purchase and can be any indexing volume. Contact Splunk Sales for more information.

The following are additional types of Enterprise licenses, which include all the same features:

No-enforcement license

If your license master is running Splunk Enterprise 6.5.0 or later, you can use a no-enforcement Enterprise license. This new license type allows users to keep searching even if you acquire five warnings in a 30 day window. Your license master still considers itself in violation, but search is not blocked.

Enterprise trial license

When you download Splunk for the first time, you are asked to register. Your registration authorizes you to receive an Enterprise **trial** license, which allows a maximum indexing volume of 500 MB/day. The Enterprise trial license expires 60 days after you start using Splunk. If you are running with a Enterprise trial license and your license expires, Splunk requires you to switch to a Splunk Free license.

Once you have installed Splunk software, you can choose to run it with the Enterprise trial license until the license expires, purchase an Enterprise license, or switch to the Free license, which is included.

Note: The Enterprise trial license is also sometimes referred to as "download-trial."

Sales trial license

If you are working with Splunk Sales, you can request trial Enterprise licenses of varying size and duration. The Enterprise trial license expires 60 days after you

start using Splunk. If you are preparing a pilot for a large deployment and have requirements for a longer duration or higher indexing volumes during your trial, contact Splunk Sales or your sales rep directly with your request.

Dev/Test licenses

With certain license programs you might have access to Dev/Test licenses to operate Splunk software in a non-production environment. If a deployment is using a Dev/Test license, all users see a Dev/Test stamp on the left side of the navigation bar in Splunk Web. The Dev/Test license can be used only for a single instance Splunk Enterprise deployment on version 6.5.0 or later.

Caution: A Dev/Test license does not stack with an Enterprise license. If you install a Dev/Test license with an Enterprise license, the Enterprise license file will be replaced.

Free license

The Free license includes 500 MB/day of indexing volume, is free (as in beer), and has no expiration date.

The following features that are available with the Enterprise license are **disabled in Splunk Free:**

- Multiple user accounts and role-based access controls
- Distributed search
- Forwarding in TCP/HTTP formats (you can forward data to other Splunk software instances, but not to non-Splunk software instances)
- Deployment management (including for clients)
- Alerting/monitoring
- Authentication and user management, including native authentication, LDAP, and scripted authentication.
    - ♦ There is no login. The command line or browser can access and control all aspects of Splunk software with no user/password prompt.
    - ♦ You cannot add more roles or create user accounts.
    - ♦ Searches are run against all public indexes, 'index=*' and restrictions on search such as user quotas, maximum per-search time ranges, search filters are not supported.
    - ♦ The capability system is disabled, all capabilities are enabled for all users accessing Splunk software.

See More about Splunk Free.

Compare license features

Consult this table for a comparison of major license types.

Behavior or functionality	Enterprise pre-6.5.0	No-enforcement Enterprise	DevTest personalized	Enterprise Trial	Free
Blocks search while in violation	yes	no	varies	yes	yes
Logs internally and displays message in Splunk Web when in warning or violation	yes	yes	yes	yes	yes
Stacks with other licenses	yes	yes	no	yes	yes
Full Enterprise feature set	yes	yes	no	yes	no

Forwarder license

This license allows forwarding (but not indexing) of unlimited data, and also enables security on the instance so that users must supply username and password to access it. (The free license can also be used to forward an unlimited amount of data, but has no security.)

Forwarder licenses are included with Splunk; you do not have to purchase them separately.

Splunk offers several forwarder options:

- The **universal forwarder** has the license enabled/applied automatically; no additional steps are required post-installation.
- The light forwarder uses the same license, but you must manually enable it by changing to the Forwarder license group.
- The heavy forwarder must also be manually converted to the Forwarder license group. If any indexing is to be performed, the instance should instead be given access to an Enterprise license **stack**. Read Groups, stacks, pools, and other terminology for more information about Splunk license terms.

Beta license

Splunk's Beta releases require a different license that is not compatible with other Splunk releases. Also, if you are evaluating a Beta release of Splunk, it will not run with a Free or Enterprise license. Beta licenses typically enable Enterprise features, they are just restricted to Beta releases. If you are evaluating a Beta version of Splunk, it will come with its own license.

Licenses for search heads (for distributed search)

A **search head** is a Splunk instance that distributes searches to other Splunk indexers. Although search heads don't usually index any data locally, you will still want to use a license to restrict access to them.

There is no special type of license specifically for search heads, that is to say, there is no "Search head license". However, **you must have an Enterprise license to configure a search head**. Splunk recommends that you add the search heads to an Enterprise **license pool** even if they are not expected to index any data. Read Groups, stacks, pools, and other terminology and Create or edit a license pool.

Note: If your existing search head has a pre-4.2 forwarder license installed, the forwarder license will not be read after you upgrade.

Licenses for indexer cluster nodes (for index replication)

As with any Splunk deployment, your licensing requirements are driven by the volume of data your indexers process. Contact your Splunk sales representative to purchase additional license volume.

There are just a few license issues that are specific to index replication:

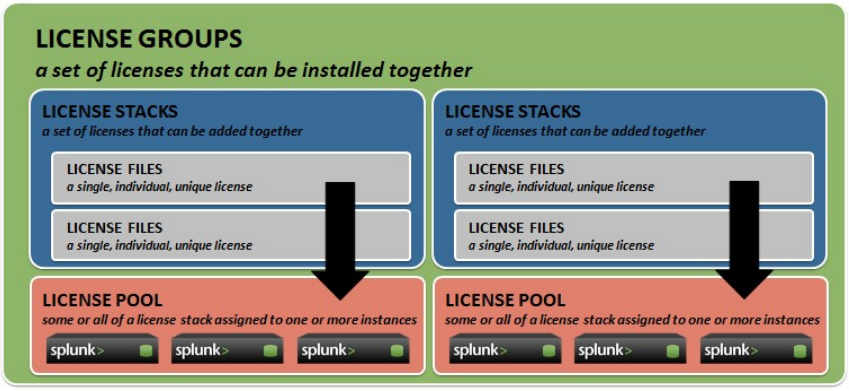
- All cluster nodes, including masters, peers, and search heads, need to be in an Enterprise license pool, even if they're not expected to index any data.
- Cluster nodes must share the same licensing configuration.
- Only incoming data counts against the license; replicated data does not.
- You cannot use index replication with a free license.

Read more about System requirements and other deployment considerations in *Managing Indexers and Clusters of Indexers*.

Groups, stacks, pools, and other terminology

You can aggregate compatible Splunk Enterprise licenses into stacks of available license volume and define pools of indexers to use license volume from a given stack.

Splunk Free users: This functionality is only relevant for Enterprise licenses. If you're running a standalone instance of Splunk Free, groups, pools, and stacks are not needed.



Stacks

Certain types of Splunk licenses can be aggregated together, or **stacked** so that the available license volume is the sum of the volumes of the individual licenses.

This means you can increase your indexing volume capacity over time as you need to without having to swap out licenses. Instead, you simply purchase additional capacity and add it to the appropriate stack.

- Enterprise licenses and sales trial licenses can be stacked together, and with each other.
- The Enterprise **trial** license that is included with the standard Splunk Enterprise download package cannot be included in a stack. The Enterprise trial license is designed for standalone use and is its own group. Until you install an Enterprise or sales trial license, you will not be able to create a stack or define a pool for other indexers to use.
- The Splunk Free license cannot be stacked with other licenses, including Splunk Free licenses.
- The forwarder license cannot be stacked with other licenses, including forwarder licenses.

- The Dev/Test license cannot be stacked with other licenses, including an Enterprise license. If you install a Dev/Test license with an Enterprise license, the Enterprise license will be deleted.

Groups

A **license group** contains one or more stacks. A stack can be a member of only one group, and only one group can be "active" in your Splunk installation at a time. Specifically this means that a given license master can only administer pools of licenses of one group type at a time. The groups are:

- Enterprise/sales trial group -- This group allows stacking of purchased Enterprise licenses, and sales trial licenses (which are Enterprise licenses with a set expiry date, NOT the same thing as the downloaded Enterprise trial).
- Enterprise trial group -- This is the default group when you first install a new Splunk platform instance. You cannot combine multiple Enterprise trial licenses into a stack and create pools from it. **If you switch to a different group, you will not be able to switch back to the Enterprise trial group.**
- Free group -- This group exists to accommodate Splunk Free installations. When an Enterprise trial license expires after 60 days, that Splunk instance is converted to the Free group. You cannot combine multiple Splunk Free licenses into a stack and create pools from it.
- Forwarder group -- This group exists for the purposes of configuring a Splunk instance as a universal forwarder or light forwarder. These types of forwarders do not perform any indexing, and therefore aren't really managed via the Licensing pages in Manager, but do belong to a license group. If you change the license group of a Splunk instance to the Forwarder group, it assumes that Splunk instance is configured as a forwarder and will not be indexing any data. See **forwarders** and "Forwarder licenses" for more information.

Subgroups

A subgroup can have one of several values, including DevTest or Production. You cannot stack two licenses with different subgroups.

Subgroups are introduced in Splunk Enterprise 6.5.0. A license with no subgroup, such as a license issued before Splunk Enterprise 6.5.0, is treated as though its subgroup is Production.

Pools

You can define a **pool** of license volume from a given license **license stack** and specify other indexing Splunk instances as members of that pool for the purposes of volume usage and tracking.

A license pool is made up of a single **license master** and zero or more **license slave** instances of Splunk configured to use licensing volume from a set license or **license stack**.

License slaves

A license slave is a member of one or more license pools. A license slave's access to license volume is controlled by its license master.

License master

A license master controls one or more license slaves. From the license master, you can define pools, add licensing capacity, and manage license slaves.

Install a license

This topic discusses installing new licenses. You can install multiple licenses on a Splunk platform **license master**. **Note:** If you install a Dev/Test license with an Enterprise license, the Enterprise license file will be replaced.

Before you proceed, you might want to review these topics:

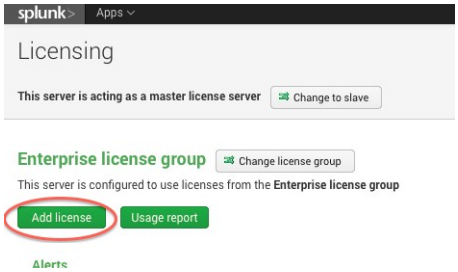
- Read How Splunk Enterprise licensing works for an introduction to Splunk software licensing.
- Read Types of Splunk software licenses to compare license types and learn which licenses can be combined, and which cannot.
- Read Groups, stacks, pools, and other terminology for more information about Splunk license terminology.

Add a new license

To add a new license:

1. Navigate to **Settings > Licensing**.

2. Click **Add license**.



3. Either click **Choose file** and browse for your license file and select it, or click **copy & paste the license XML directly...** and paste the text of your license file into the provided field.

4. Click **Install**. If this is the first Enterprise license that you are installing, you must restart Splunk Enterprise. Your license is installed.

Configure a license master

This topic discusses configuring a Splunk instance as a **license master**. Before you proceed, you may want to review these topics:

- Read "How Splunk licensing works" in this manual for an introduction to Splunk licensing.
- Read "Groups, stacks, pools, and other terminology" in this manual for more information about Splunk license terms.

Kinds of license masters

There are two basic styles of license master:

- **Standalone** license master
  - If you have a single Splunk indexer and want to manage its licenses, you can run it as its own license master, install one or more Enterprise licenses on it and it will manage itself as a license slave.
  - When you first download and install Splunk Enterprise, it includes a 500 MB 60-day Enterprise Trial license. This instance is automatically configured as a standalone license master, and you cannot create a pool or define any **license slaves** for this type of license. If you want to create one or more stacks or pools and

assign multiple indexers to them, you must purchase and install an Enterprise license. To install a license, follow the instructions in "Install a license" in this manual.

- **Central** license master
  - If you have more than one indexer and want to manage their access to purchased license capacity from a central location, configure a central license master and add the indexers to it as **license slaves**.
  - If the license master is also an indexer, it will be its own license master as well, but Splunk recommends that if you have a **search head**, you designate it as the license master.
  - If you have a large environment with multiple search heads, you might want to have some or all search heads that are not the license master distribute searches to the license master, for two reasons:
    - You can run searches against the license logs.
    - If an unusual condition occurs on the search head (for example you have a time-limited license and it will expire in 5 days), this condition will be visible on the search head when running a search, as part of the info messages attached to search results.

License master and slave compatibility

A license master must always be of equal or later version than its license slaves.

License master version	Compatible license slave versions
6.1.x	5.x, 6.0.x, and 6.1.x
6.2.x	5.x, 6.0.x, 6.1.x, 6.2.x
6.3.x	5.x, 6.0.x, 6.1.x, 6.2.x, 6.3.x
6.4.x	5.x, 6.0.x, 6.1.x, 6.2.x, 6.3.x, 6.4.x

Configure a central license master

By default, a standalone instance of Splunk is its own license master. To configure a central license master, install one or more Enterprise licenses.

Once an Enterprise license is installed, you can create one or more stacks and pools to access the installed license, and manage them from the license master.

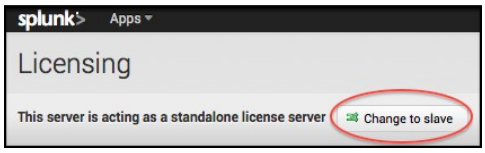
Configure a license slave

This topic discusses configuring a Splunk indexer as a **license slave**. Before you proceed, you may want to review these topics:

- Read "How Splunk licensing works" in this manual for an introduction to Splunk licensing.
- Read "Groups, stacks, pools, and other terminology" in this manual for more information about Splunk license terms.
- ReadConfigure a license master" in this manual for instructions on setting up a license master.
- Read "Manage licenses from the CLI" in this manual for help with performing these tasks from the command line.

1. On the indexer you want to configure as a license slave, log into Splunk Web and navigate to **Settings > Licensing**.

2. Click **Change to Slave**.



3. Switch the radio button from **Designate this Splunk instance, *<this indexer>*, as the master license server** to **Designate a different Splunk instance as the master license server**.

4. Specify the license master to which this license slave should report. You must provide either an IP address or a hostname and the **Splunk management port**, which is 8089 by default.

Note: The IP address can be specified in IPv4 or IPv6 format. For detailed information on IPv6 support, read "Configure Splunk for IPv6" in this manual.

5. Click **Save**. If this instance does not already have an Enterprise license installed, you must restart Splunk. This indexer is now configured as a license slave.

To switch back, navigate to **Settings > Licensing** and click **Switch to local master**. If this instance does not already have an Enterprise license installed, you must restart Splunk for this change to take effect.

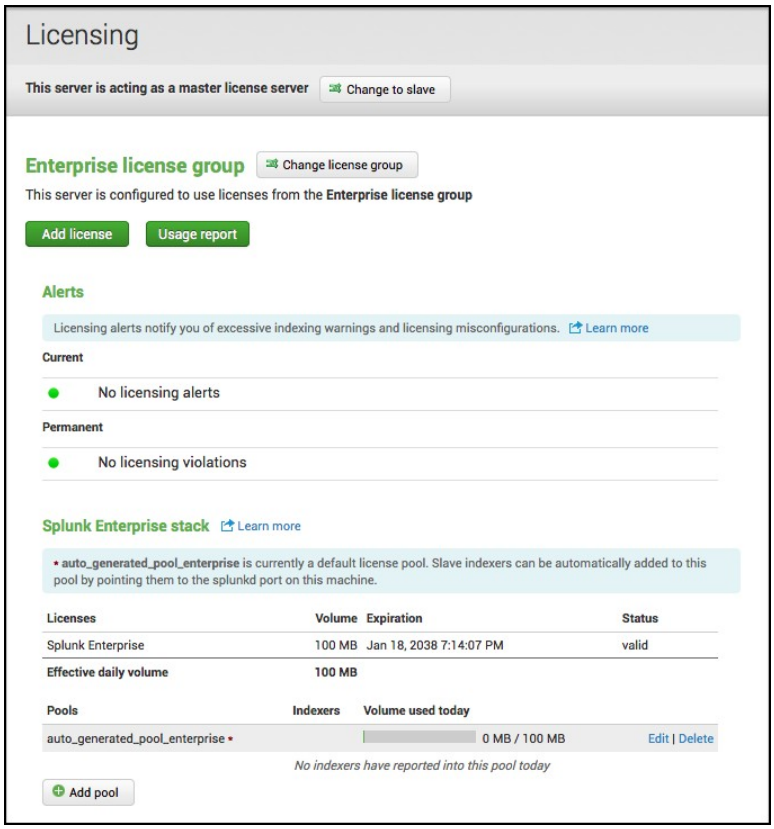
Create or edit a license pool

This topic discusses creating a license pool from one or more installed licenses, as well as editing an existing license pool. Before you proceed, you may want to review these topics:

- Read "How Splunk licensing works" in this manual for an introduction to Splunk licensing.
- Read "Groups, stacks, pools, and other terminology" in this manual for more information about Splunk license terms.
- Read "Install a license" to learn more about installing licenses.
- Read "Manage licenses from the CLI" in this manual for help with performing these tasks from the command line.

When you first download and install Splunk, it includes a 500 MB 60 day Enterprise Trial license. This instance of Splunk is automatically configured as a stand-alone **license master**, and you cannot create a pool or define any **license slaves** for this type of license. If you want to create one or more stacks or pools and assign multiple indexers to them, you must purchase and install an Enterprise license.

In the following example of **Settings > Licensing**, a 100 MB Enterprise license has just been installed onto a brand new Splunk installation:



When you install an Enterprise license onto a brand new Splunk server, Splunk automatically creates an Enterprise **license stack** called Splunk Enterprise Stack from it and defines a default **license pool** for it called `auto_generated_pool_enterprise`.

The default configuration for this default pool adds any license slave that connects to this license master to the pool. You can edit the pool to change this configuration, to add more indexers to it, or create a new license pool from this stack.

To edit an existing license pool

1. Next to the license pool you want to edit, click **Edit**. The Edit license pool page is displayed.

2. If desired, change the allocation or alter how indexers are permitted to access this pool. You can also change the description, but not the name of the pool.

3. Click **Submit**.

To create a new license pool

Important: Before you can create a new license pool from the default Enterprise stack, you must make some indexing volume available by either editing the `auto_generated_pool_enterprise` pool and reducing its allocation, or deleting the pool entirely. Click **Delete** next to the pool's name to delete it.

1. Click 

toward the bottom of the page. The Create new license pool page is displayed.

2. Specify a name and optionally, a description for the pool.

3. Set the allocation for this pool. The allocation is how much of the overall stack's licensing volume is available for use by the indexers who belong to this pool. The allocation can be a specific value, or the entire amount of indexing volume available in the stack, as long as it is not allocated to any other pool.

4. Specify how indexers are to access this pool. The options are:

- Any indexer in your environment that is configured as license slave can connect to this license pool and use the license allocation within it.
- Only indexers that you specify can connect to this pool and use the license allocation within it.

5. To allow a specific indexer to draw from the pool, click the plus sign next to the name of the indexer in the list of Available indexers to move it into the list of Associated indexers.

Add an indexer to a license pool

This topic discusses adding indexers to existing **license pools**. Before you proceed, you may want to review these topics:

- Read "How Splunk licensing works" in this manual for an introduction to Splunk licensing.
- Read "Groups, stacks, pools, and other terminology" in this manual for more information about Splunk license terms.

How indexers access license pools

Access to a license pool's **stack** is controlled by that pool's **license master**. A pool can be configured to allow access only to specific indexers, or to all indexers who connect to it by specifying the URI and management port of the license master.

Add a specific indexer

Follow these two basic steps to give a specific indexer access to a given license pool's stack:

1. Configure the indexer to be a license slave and give it the license master's URI and management port. To do this, follow the instructions in "Configure a license slave" in this manual.

2. Configure the pool on the license manager to accept access from that indexer. To do this, follow the instructions in "Create or edit a license pool" to edit a license pool, choose the radio button option to only allow access to **Specific indexers** and then click the plus sign next to the names of the indexer in the list of Available indexers to move them into the list of Associated indexers.

Add any indexer that connects

Follow these steps to give all indexers who connect to this license master access to a given license pool's stack:

1. Configure the indexer to be a license slave and give it the license master's URI and management port. To do this, follow the instructions in "Configure a license slave" in this manual.

2. Configure the pool on the license master to accept access from any indexer. To do this, follow the instructions in "Create or edit a license pool" to edit a license pool, and choose the radio button option to allow access from **Any indexer that connects**.

Manage licenses from the CLI

This topic describes using the Splunk CLI to monitor and manage your Splunk licenses. Before you continue, review these topics:

- Read "How Splunk licensing works" in this manual for an introduction to Splunk licensing.
- Read "Groups, stacks, pools, and other terminology" in this manual for more information about Splunk license terms.

This topic only covers what CLI commands you can use to interact with Splunk's licenser-related objects. Some of these commands also have required and optional arguments that you can specify for each object. For the complete syntax and usage examples, refer to Splunk's CLI help.

- Read "About the CLI" in this manual for an introduction to using the Splunk command line interface.

For information on managing licenses through Splunk's REST API, refer to "Licenses" in the REST API Reference Manual.

CLI licenser commands and objects

Using the Splunk CLI, you can add, edit, list, and remove licenses and licenser-related objects. The available commands are:

Command	Object(s)	Description
add	licenses, licenser-pools	Add a license or a pool of licenses to a license stack. This command is only available if you have an Enterprise license.
edit	licenser-localslave, licenser-pools	Edit the attributes of a local licenser-slave node or a pool of licenses within a license stack. This command is only available if you have an Enterprise license.
list	licenser-groups, licenser-localslave, licenser-messages, licenser-pools, licenser-slaves, licenser-stacks, licenses	Depending on the licenser-related object specified, lists either the attributes of that object or members of that object.
remove	licenser-pools, licenses	Remove licenses or license pools from a license stack.

License-related objects are:

Object	Description
licenser-groups	the different license groups you can switch to.
licenser-localslave	a local indexer's configuration.
licenser-messages	the alerts or warnings about the state of your licenses.
licenser-pools	a pool, or virtual license. A stack can be divided into various pools, with multiple slaves sharing the quota of each pool.
licenser-slaves	all the slaves that have contacted the master.
licenser-stacks	this object represents a stack of licenses. A stack contains licenses of the same type and are cumulative.
licenses	all licenses for this Splunk instance.

Common licenser-related tasks

The following are examples of common licenser-related tasks.

Managing licenses

To add a new license to the license stack, specify the path to the license file:

```
./splunk add licenses
/opt/splunk/etc/licenses/enterprise/enterprise.lic
```

To list all the licenses in a license stack:

```
./splunk list licenses
```

List also displays the properties of each license, including the features it enables (features), the license group and stack it belongs to (group_id, stack_id), the indexing quote it allows (quota), and the license key that is unique for each license (license_hash).

If a license expires, you can remove it from the license stack. To remove a license from the license stack, specify the license's hash:

```
./splunk remove licenses
BM+S8VetLnQEb1F+5Gwx9rR4M4Y91AkIE=781882C56833F36D
```

Managing license pools

You can create a license pool from one or more licenses in a license stack (if you have an Enterprise license). Basically, a license stack can be carved up into multiple licenser pools. Each pool can have more than one license slave sharing the quota of the pool.

To see all the license pools in all the license stacks:

```
./splunk list licenser-pools
```

To add a license pool to the stack, you need to: name the pool, specify the stack that you want to add it to, and specify the indexing volume allocated to that pool:

```
./splunk add licenser-pools pool01 -quota 10mb -slaves guid1,guid2
-stack_id enterprise
```

You can also specify a description for the pool and the slaves that are members of the pool (these are optional).

You can edit the license pool's description, indexing quota, and slaves:

```
./splunk edit licenser-pools pool01 -description "Test" -quota 15mb
-slaves guid3,guid4 -append_slaves true
```

This basically adds a description for the pool, "Test", changes the quota from 10mb to 15mb, adds slaves guid3 and guid4 to the pool (instead of overwriting or replace guid1 and guid2).

To remove a license pool from a stack, specify the name:

```
./splunk remove licenser-pools pool01
```

Managing license slaves

A license slave is a member of one or more license pools. The license slaves access to license volume is controlled by its license master.

To list all the license slaves that have contacted the license master:

```
./splunk list licenser-slaves
```

To list all the properties of the local license slave:

```
./splunk list licenser-localslave
```

To add a license slave, edit the attributes of that local license slave node (specify the uri of the splunkd license master instance or 'self'):

```
./splunk edit licenser-localslave -master_uri 'https://master:port'
```

Monitoring license status

You can use the list command to view messages (alerts or warnings) about the state of your licenses.

```
./splunk list licenser-messages
```

Manage Splunk licenses

Manage your licenses

This topic discusses managing Splunk Enterprise licenses. Before you proceed, you may want to review these topics:

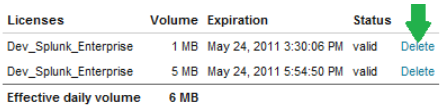
- Read How Splunk licensing works in this manual.
- Read Groups, stacks, pools, and other terminology in this manual.
- Read Manage licenses from the CLI in this manual for help with performing some of these tasks from the command line.

For information about upgrading an existing license, see Migrate to the new Splunk licenser in the *Installation Manual*.

Delete a license

If a license expires, you can delete it. To delete one or more licenses:

1. On the license master, navigate to **System > Licensing**.



Licenses	Volume	Expiration	Status	
Dev_Splunk_Enterprise	1 MB	May 24, 2011 3:30:06 PM	valid	Delete
Dev_Splunk_Enterprise	5 MB	May 24, 2011 5:54:50 PM	valid	Delete
Effective daily volume	**6 MB**			

2. Click **Delete** next to the license you want to delete.

3. Click **Delete** again to confirm.

Note: You cannot delete the last license in a list of licenses on a license master.

View license usage

You can monitor license usage across your deployment with the License Usage Report View. Access the view in **System > Licensing > Usage Report**. Read more about the License Usage Report View in the next chapter.

About license violations

This topic discusses license violations, how they occur, and how to resolve them. Before you proceed, you might want to review these topics:

- Read Types of Splunk software licenses for information about the new no-enforcement license.
- Read How Splunk Enterprise licensing works for an introduction to Splunk Enterprise licensing.

What are license violations and warnings?

Warnings and violations occur when you exceed the maximum indexing volume allowed for your license.

If you exceed your licensed daily volume on any one calendar day, you get a violation *warning*. If you have 5 or more warnings on an enforced Enterprise license, or 3 warnings on a Free license, in a rolling 30-day period, you are in *violation* of your license. Unless you are using a Splunk Enterprise 6.5.0 or later no-enforcement license, search is disabled for the offending **pool(s)**. Other pools remain searchable, as long as the total license usage from all pools is less than the total license quota for the license master.

Search capabilities return when you have fewer than 5 (Enterprise) or 3 (Free) warnings in the previous 30 days, or when you apply a temporary reset license (available for Enterprise only). To obtain a reset license, contact your sales representative. See Install a license.

Starting with Splunk Enterprise 6.5.0, Enterprise customers can request a no-enforcement license. This license warns you when you exceed your license quota or are in license violation, but it does not disable search. Even during a violation period, search remains enabled. See Types of Splunk software licenses for details.

Note: Summary indexing volume does not count against your license, although in the event of a license violation, summary indexing halts like any other noninternal search behavior. Internal indexes (for example, `_internal` and `_introspection`) do not count against your license volume.

If you get a license warning, you have until midnight (going by the time on the license master) to resolve it before it counts against the total number of warnings within the rolling 30 day period.

During a license violation period:

- **Splunk software does not stop indexing your data.**
- If you are using a pre-6.5.0 license, Splunk software blocks search while you are in license violation.
- If you are using a new no-enforcement license, search continues even while you are in license violation.
- Searches to the `_internal` index are never disabled. This means that you can access the Monitoring Console or run searches against `_internal` to diagnose the licensing problem.

What license warnings look like

If indexers in a pool exceed the license volume allocated to that pool, you will see a message in **Messages** on any page in Splunk Web.

Clicking the link in the message takes you to **Settings > Licensing**, where the warning displays under the **Alerts** section of the page. Click a warning to get more information about it.

A similar message displays on license slaves when a violation has occurred.

Here are some of the conditions that generate a licensing alert:

- When a slave becomes an orphan, there is an alert (transient and fixable before midnight).
- When a pool has maxed out, there is an alert (transient and fixable before midnight).
- When a stack has maxed out, there is an alert (transient and fixable before midnight).
- When a warning is given to one or more slaves, there is an alert. The alert stays as long as the warning is still valid within that last 30-day period.

About the connection between the license master and license slaves

When you configure a license master instance and add license slaves to it, the license slaves communicate their usage to the license master every minute. If the license master is down or unreachable for any reason, the license slave starts a 72 hour timer. If the license slave cannot reach the license master for 72 hours, search is blocked on the license slave (although indexing continues). Users cannot search data in the indexes on the license slave until that slave can reach the license master again.

To find out if a license slave has been unable to reach the license master, look for an event that contains `failed to transfer rows` in splunkd.log or search for it in the `_internal` index.

How to avoid license violations

To avoid license violations, monitor your license usage and ensure you have sufficient license volume to support it. If you do not have sufficient license volume, you need to either increase your license or decrease your indexing volume.

The distributed management console contains alerts that you can enable, including one that monitors license usage. See Platform alerts in *Monitoring Splunk Enterprise*.

Use the **License Usage** report to see details about and troubleshoot index volume in your deployment. Read about the license usage report view in the next chapter.

Correcting license warnings

If Splunk software tells you to correct your license warning before midnight, your quota is probably already exceeded for the day. This is called a "soft warning." The daily license quota resets at midnight (at which point the soft warning becomes a "hard warning"). You have until then to fix your situation and ensure that you will not go over quota tomorrow, too.

Once data is already indexed, there is no way to un-index data to give you "wiggle room" back on your license. You need to get additional license room in one of these ways:

- Purchase a bigger license.
- Rearrange license pools if you have a pool with extra license room.
- Request a no-enforcement Enterprise license if your license master is running Splunk Enterprise 6.5.0 or later.

If you cannot do any of these, prevent a warning tomorrow by using less of your license. Take a look at the License Usage Report View to learn which data sources are contributing the most to your quota.

Once you identify a data culprit, decide whether or not you need all the data it is emitting. If not, read Route and filter data in the *Forwarding Data* manual.

Answers

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

Swap the license master

This procedure assumes that you have already configured a license pool. What if you want to turn one of your license slaves into your pool's license master?

This topic spells out the steps to do that. Big picture, first you promote a slave to master. Then you demote the old master to a slave. Details follow.

1. Remove the new license master from the licensing pool and set it up as a master.

- Log into license slave (which will become new master).
- Navigate to **Settings > Licensing**.
- Follow the prompts to configure it as a new license master.
- Restart Splunk.

2. On the new license master, add the license keys. Check that the license keys match up to the old license master.

3. Make the other license slaves in the pool point to the new license master.

- On each of the slaves, navigate to **Settings > Licensing**.
- Change the master license server URI to refer to the new license master and click **Save**.
- Restart Splunk on the license slave whose entry you just updated.

4. Check that one of the license slaves is connected to the new license master.

5. Demote the old license master to a slave:

- On the old license master, navigate to **Settings > Licensing > Change to slave**.
- Ignore the restart prompt.
- On the "Change to slave" screen, point the new slave to the new license master by clicking **Designate a different Splunk Enterprise instance as**

the master license server.

6. On the new license slave, stop Splunk Enterprise and delete the old license file(s) under the /opt/splunk/etc/licenses/enterprise/ folder. (Otherwise you'll have duplicate licenses and will get errors and/or warnings.)

7. On the new license slave, start Splunk Enterprise and confirm that it connects to the new license master.

License Usage Report View

About the Splunk Enterprise license usage report view

Introduction to the license usage report view

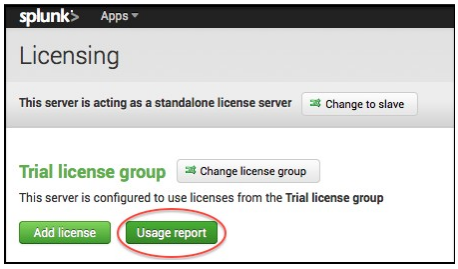
The license usage report view (LURV) on your deployment's license master is consolidated resource for questions related to your Splunk license capacity and indexed volume. Directly from the Splunk Licensing page, you can see your daily indexing volume, any license warnings, and a view of the last 30 days of your license usage with multiple reporting options.

LURV displays detailed license usage information for your license pool. The dashboard is logically divided into two parts: one displays information about today's license usage, and any warning information in the current rolling window; the other shows historic license usage during the past 30 days.

For every panel in LURV, you can click "Open in search" at the bottom left of the panel to interact with the search.

Access the license usage report view

Find LURV in **Settings > Licensing > Usage report** on your deployment's license master. (If your deployment is only one instance, your instance is its own license master.)



Today tab

When you first arrive at LURV, you'll see five panels under the "Today" tab. These panels show the status of license usage and the warnings for the day that hasn't yet finished. The licenser's day ends at midnight in whichever time zone

the license master is set to.

All the panels in the "Today" tab query the Splunk REST API.

Today's license usage panel

This panel gauges license usage for today, as well as the total daily license quota across all pools.

Today's license usage per pool panel

This panel shows the license usage for each pool as well as the daily license quota for each pool.

Today's percentage of daily license quota used per pool panel

This panel shows what percentage of the daily license quota has been indexed by each pool. The percentage is displayed on a logarithmic scale.

Pool usage warnings panel

This panel shows the warnings, both soft and hard, that each pool has received in the past 30 days (or since the last license reset key was applied). Read "About license violations" in this manual to learn more about soft and hard warnings, and license violations.

Slave usage warnings panel

For each license slave, this panel shows: the number of warnings, pool membership, and whether the slave is in violation.

Previous 30 Days tab

Clicking on the "Previous 30 Days" tab reveals five more panels and several drop-down options.

All visualizations in these panels limit the number of host, source, source type, index, pool (any field you split by) that are plotted. If you have more than 10 distinct values for any of these fields, the values after the 10th are labeled "Other." We've set the maximum number of values plotted to 10 using `timechart`. We hope this gives you enough information most of the time without making the visualizations difficult to read.

These panels all use data collected from `license_usage.log`, `type=RolloverSummary` (daily totals). If your **license master** is down at its local midnight, it will not generate a RolloverSummary event for that day, and you will not see that day's data in these panels.

Split-by: no split, indexer, pool

These three split-by options are self-explanatory. Read about adding an indexer to a license pool and about license pools in previous chapters in this manual.

Split-by: source, source type, host, index

There are two things you should understand about these four split-by fields: report acceleration and squashing.

Report acceleration

Splitting by source, source type, and host uses `license_usage.log type=Usage`, which provides real-time usage statistics at one-minute intervals. We recommend accelerating the report that powers these split-by options **on your license master**. (Without acceleration, the search can be very slow, since it searches through 30 days worth of data that gets generated at a rate of one event per minute -- that's a lot of events!)

Acceleration for this report is disabled by default. To accelerate the report, click the link that shows up in the info message when you select one of these split-by values. You can also find the workflow for accelerating in **Settings > Searches and reports > License usage data cube**. Read "Accelerate reports" in the Reporting Manual.

Note that report acceleration can take up to 10 minutes to start after you select it for the first time. Then Splunk will take some amount time to build the acceleration summary -- typically a few to tens of minutes, depending on the amount of data it's summarizing. Only after the acceleration is finished building will you see faster performance for these split-by options.

But after the first acceleration run, subsequent reports will build on what's already there, keeping the report up-to-date (and the reporting fast). You should only have a long wait the very first time you turn on report acceleration.

Important: Enable report acceleration only on your license master.

Configure how frequently the acceleration runs in savedsearches.conf, with `auto_summarize`. The default is every 10 minutes. Keep it frequent, to keep the workload small and steady. We put in a cron for every 10 minutes at the 3 minute mark. This is configurable in `auto_summarize.cron_schedule`.

Squashing

Every indexer periodically reports to license manager stats of the data indexed: broken down by source, source type, host, and index. If the number of distinct (source, source type, host, index) tuples grows over the `squash_threshold`, Splunk squashes the {host, source} values and only reports a breakdown by {sourcetype, index}. This is to prevent explosions in memory and license_usage.log lines.

Because of squashing on the other fields, only the split-by source type and index will guarantee full reporting (every byte). Split by source and host do not guarantee full reporting necessarily, if those two fields represent many distinct values. Splunk reports the entire quantity indexed, but not the names. So you lose granularity (that is, you don't know who consumed that amount), but you still know what the amount consumed is.

Squashing is configurable (with care!) in server.conf, in the `[license]` stanza, with the `squash_threshold` setting. You can increase the value, but doing so can use a lot of memory, so consult a Splunk Support engineer before changing it.

LURV will always tell you (with a warning message in the UI) if squashing has occurred.

If you find that you need the granular information, you can get it from metrics.log instead, using `per_host_thruput`.

Top 5 by average daily volume

The "Top 5" panel shows both average and maximum daily usage of the top five values for whatever split by field you've picked from the Split By menu.

Note that this selects the top five average (not peak) values. So, for example, say you have more than five source types. Source type F is normally much smaller than the others but has a brief peak. Source type F's **max** daily usage is very high, but its **average** usage might still be low (since it has all those days of very low usage to bring down its average). Since this panel selects the top five **average** values, source type F might still not show up in this view.

Use LURV

Read the next topic for a tip about configuring an alert based on a LURV panel.

Use the license usage report view

This topic is about using the license usage report view (LURV). To learn about the view, read the previous topic, "About Splunk's license usage report view."

Set up an alert

You can turn any of the LURV panels into an alert. For example, say you want to set up an alert for when license usage reaches 80% of the quota.

Start at the **Today's percentage of daily license usage quota used** panel. Click "Open in search" at the bottom left of a panel. Append

```
| where '% used' > 80
```

then select **Save as > Alert** and follow the alerting wizard.

Splunk Enterprise comes with some alerts preconfigured that you can enable. See "Platform alerts" in the *Distributed Management Console Manual*.

Troubleshoot LURV: no results in 30 days panel

A lack of results in the panels of the "Last 30 days" view of the License Usage Report View indicates that the license master instance on which this page is viewed is unable to find events from its own `$SPLUNK_HOME/var/log/splunk/license_usage.log` file when searching.

This typically has one of two causes:

- The **license master** is configured to forward its events to the indexers (read more about this best practice in the Distributed Search Manual) but it has not been configured to be a search head. This is easily remedied by adding all indexers to whom the license master is forwarding events as search peers.
- The license master is not reading (and therefore, indexing) events from its own `$SPLUNK_HOME/var/log/splunk` directory. This can happen if the `[monitor://$SPLUNK_HOME/var/log/splunk]` default data input is disabled

for some reason.

You might also have a gap in your data if your license master is down at midnight.

Administer the app key value store

About the app key value store

The app key value store (or KV store) provides a way to save and retrieve data within your Splunk apps, thereby letting you manage and maintain the state of the application.

Here are some ways that Splunk apps might use the KV Store:

- Tracking workflow in an incident-review system that moves an issue from one user to another.
- Keeping a list of environment assets provided by users.
- Controlling a job queue.
- Managing a UI session by storing the user or application state as the user interacts with the app.
- Storing user metadata.
- Caching results from search queries by Splunk or an external data store.
- Storing checkpoint data for modular inputs.

For information on using the KV store, see app key value store documentation for Splunk app developers.

How KV store works with your deployment

The KV store stores your data as key-value pairs in collections. Here are the main concepts:

- **Collections** are the containers for your data, similar to a database table. Collections exist within the context of a given app.
- **Records** contain each entry of your data, similar to a row in a database table.
- **Fields** correspond to key names, similar to the columns in a database table. Fields contain the values of your data as a JSON file. Although it is not required, you can enforce data types (number, boolean, time, and string) for field values.
- **_key** is a reserved field that contains the unique ID for each record. If you don't explicitly specify the _key value, the app auto-generates one.
- **_user** is a reserved field that contains the user ID for each record. This field cannot be overridden.

- **Accelerations** improve search performance by making searches that contain accelerated fields return faster. Accelerations store a small portion of the collection's data set in an easy-to-traverse form.

The KV store files reside on search heads.

In a search head cluster, if any node receives a write, the KV store delegates the write to the **KV store captain**. The KV store keeps the reads local, however.

System requirements

KV store is available and supported on all Splunk Enterprise 64-bit builds. It is not available on 32-bit Splunk Enterprise builds. KV store is also not available on universal forwarders. See the Splunk Enterprise system requirements.

KV store uses port 8191 by default. You can change the port number in server.conf's `[kvstore]` stanza. For information about other ports that Splunk Enterprise uses, see "System requirements and other deployment considerations for search head clusters" in the *Distributed Search Manual*.

For information about other configurations that you can change in KV store, see the "KV store configuration" section in server.conf.spec.

About Splunk FIPS

To use FIPS with KV store, see the "KV store configuration" section in server.conf.spec.

If Splunk FIPS is not enabled, those settings will be ignored.

If you enable FIPS but do not provide the required settings (`caCertFile`, `sslKeysPath`, and `sslKeysPassword`), KV store does not run. Look for error messages in splunkd.log and on the console that executes `splunk start`.

Determine whether your apps use KV store

KV store is enabled by default on Splunk Enterprise 6.2+.

Apps that use the KV store typically have collections.conf defined in `$SPLUNK_HOME/etc/apps/<app name>/default`. In addition, transforms.conf will have references to the collections with external_type = kvstore

Use the KV store

To use the KV store:

1. Create a collection and optionally define a list of fields with data types using configuration files or the REST API.

2. Perform create-read-update-delete (CRUD) operations using search lookup commands and the Splunk REST API.

3. Manage collections using the REST API.

Monitor the KV store on your Splunk Enterprise deployment

You can monitor your KV store performance through two views in the monitoring console. One view provides insight across your entire deployment. See KV store: Deployment in *Monitoring Splunk Enterprise*.

The instance-scoped view gives you detailed information about KV store operations on each search head. See KV store: Instance in *Monitoring Splunk Enterprise*.

Resync the KV store

When a KV store member fails to transform its data with all of the write operations, then the KV store member might be stale. To resolve this issue, you must resynchronize the member.

Identify a stale KV store member

You can check the status of the KV store using the command line.

1. Log into the shell of any KV store member.
2. Navigate to the `bin` subdirectory in the Splunk Enterprise installation directory.
3. Type `./splunk show kvstore-status`. The command line returns a summary of the KV store member you are logged into, as well as information about every other member in the KV store cluster.
4. Look at the `replicationStatus` field and identify any members that have neither "KV store captain" nor "Non-captain KV store member" as values.

Resync stale KV store members

If more than half of the members are stale, you can either recreate the cluster or resync it from one of the members. See Back up KV store for details about restoring from backup.

To resync the cluster from one of the members, use the following procedure. This triggers the recreation of the KV store cluster, when all of the members of current existing KV store cluster resynchronize all data from the current member (or from the member specified in -source sourceId). The command to resync the KV store cluster can be invoked only from the node that is operating as search head cluster captain.

1. Determine which node is currently the search head cluster captain. Use the CLI command `splunk show shcluster-status`.
2. Log into the shell on the search head cluster captain node.
3. Run the command `splunk resync kvstore [-source sourceId]`. The source is an optional parameter, if you want to use a member other than the search head cluster captain as the source.
4. Enter your admin login credentials.
5. Wait for a confirmation message on the command line.
6. Use the `splunk show kvstore-stats` command to verify that the cluster is resynced.

If fewer than half of the members are stale, resync each member individually.

1. Stop the search head that has the stale KV store member.
2. Run the command `splunk clean kvstore --local`.
3. Restart the search head. This triggers the initial synchronization from other KV store members.
4. Run the command `splunk show kvstore-status` to verify synchronization.

Back up KV Store

This topic describes how to safely back up and restore your KV Store.

Back up the KV Store

Before performing these steps make sure to be familiar with the standard backup and restore tools and procedures used by your organization.

1. To back up KV store data, first shut down the Splunk instance from which the KV Store will be backed up.

2. Back up all files in the path that is specified in the `dbPath` parameter of the `[kvstore]` stanza in the `server.conf` file.

3. On a single node, back up the `kvstore` folder found in your `$SPLUNK_DB` path. By default the path is `/var/lib/splunk/kvstore`.

If using a search head cluster, back up the KV Store data on any cluster member.

Restore the KV Store data

Note: In order to successfully restore KV Store data, the KV Store collection `collections.conf` must already exist on the Splunk instance the KV Store will be restored to. If you create the collection `collections.conf` after restoring the KV Store data, then the KV Store data will be lost.

To restore the KV Store data to the same search head cluster that it was backed up from, restore the `kvstore` folder on each cluster member. For example, in a three-member search head cluster:

1. Back up the KV Store data from a member of the search head cluster.

2. Stop each cluster member.

3. Restore the backed-up KV Store data folder to each cluster member.

4. Start each cluster member.

Restore the KV Store data to a new member being added to the search head cluster

Restore the KV Store data to the new member and add the new member to the cluster. For example, in a three-member search head cluster:

1. Back up the KV Store data from a member of the search head cluster.

2. On the search head that you want to add to the search head cluster:

a. Add the member to the cluster. See "Add a cluster member" in the *Distributed Search* manual.

b. Stop the member.

c. Restore the KV Store data.

d. Start the new member.

Restore the KV Store data from an old search head cluster to a new search head cluster

Note: This procedure assumes you are creating a new search head cluster with new Splunk Enterprise instances.

1. Back up the KV Store data from a search head in the current (old) search head cluster.

2. To restore the KV Store data onto a new search head cluster , the search head cluster must be initialized with one member and before bootstrapping the one member restore the KV Store data folder, then add the rest of the search heads to the search head cluster environment. This example uses a three-node old search head cluster environment and three-node new search head cluster environment:

- Back up the data from a search head in the old search head cluster.
- On a search head that will be in the new search head cluster environment.
- Create the KV Store collection using the same collection name as the KV Store data you are restoring.
- Initialize the search head cluster with `replication_factor=1`
- Stop the Splunk instance and restore the KV Store data.
- Clean the KV Store cluster. This removes cluster information from previous clusters:

splunk clean kvstore -cluster

- Start the Splunk instance and bootstrap with just this one search head.
- Once the KV Store has been restored onto the search head that will be in the new search head cluster environment, to which you can now add the other new search head cluster members.
- Once complete, go in and change the `replication_factor` on each search head to the desired replication factor number and perform a rolling restart.

KV store troubleshooting tools

KV store status

You can check the status of the KV store by using the command line or by making a REST API GET request.

Using the command line from any KV store member, in `$SPLUNK_HOME/bin` type: `./splunk show kvstore-status`. See About the CLI for information about using the CLI in Splunk software.

The following is a list of possible statuses.

KV store status	Definition
Startup	Member is just starting, give it time
KV store captain	Member has been elected KV store captain
Non-captain KV store member	Healthy non-captain member of KV store cluster
Initial sync	This member is resynchronizing data from one of the other KV store cluster members. If this happens too often or if this member is stuck in this state, check mongod.log and splunkd.log on this member, and verify connection to this member and connection speed.
Down	Member has been stopped.
Removed	Member has been removed from the KV store cluster, or is in the process of being removed.
Rollback / Recovering / Unknown status	Member might have a problem. Check mongod.log and splunkd.log on this member

Sample response:

```
This member:
                                    date : Tue Jul 21 16:42:24 2016
                                 dateSec : 1466541744.143000
                                disabled : 0
                                    guid :
6244DF36-D883-4D59-AHD3-5276FCB4BL91
                           oplogEndTimestamp : Tue Jul 21 16:41:12 2016
                        oplogEndTimestampSec : 1466541672.000000
                         oplogStartTimestamp : Tue Jul 21 16:34:55 2016
```

```
                  oplogStartTimestampSec : 1466541295.000000
                                    port : 8191
                              replicaSet : splunkrs
                       replicationStatus : KV store captain
                              standalone : 0
                                  status : ready

 Enabled KV store members:
        10.140.137.128:8191
                                    guid :
6244DF36-D883-4D59-AHD3-5276FCB4BL91
                              hostAndPort : 10.140.137.128:8191
        10.140.137.119:8191
                                    guid :
8756FA39-F207-4870-BC5D-C57BABE0ED18
                              hostAndPort : 10.140.137.119:8191
        10.140.136.112:8191
                                    guid :
D6190F30-C59A-423Q-AB48-80B0012317V5
                              hostAndPort : 10.140.136.112:8191

 KV store members:
        10.140.137.128:8191
                             configVersion : 1
                              electionDate : Tue Jul 21 16:42:02 2016
                            electionDateSec : 1466541722.000000
                              hostAndPort : 10.140.134.161:8191
                               optimeDate : Tue Jul 21 16:41:12 2016
                             optimeDateSec : 1466541672.000000
                           replicationStatus : KV store captain
                                  uptime : 108
        10.140.137.119:8191
                             configVersion : 1
                               hostAndPort : 10.140.134.159:8191
                             lastHeartbeat : Tue Jul 21 16:42:22 2016
                          lastHeartbeatRecv : Tue Jul 21 16:42:22 2016
                       lastHeartbeatRecvSec : 1466541742.490000
                           lastHeartbeatSec : 1466541742.937000
                                 optimeDate : Tue Jul 21 16:41:12 2016
                             optimeDateSec : 1466541672.000000
                                    pingMs : 0
                           replicationStatus : Non-captain KV store member
                                  uptime : 107
        10.140.136.112:8191
                             configVersion : -1
                               hostAndPort : 10.140.133.82:8191
                             lastHeartbeat : Tue Jul 21 16:42:22 2016
                          lastHeartbeatRecv : Tue Jul 21 16:42:00 2016
                       lastHeartbeatRecvSec : 1466541720.503000
                           lastHeartbeatSec : 1466541742.959000
                                 optimeDate : ZERO_TIME
                             optimeDateSec : 0.000000
```

```
                    pingMs : 0
        replicationStatus : Down
                    uptime : 0
```

Using the REST API, you can use cURL to make a GET request:

```
curl -k -u user:pass
https://<host>:<mPort>/services/server/info
```

See Basic Concepts in the *REST API User Manual* for more information about the REST API.

KV store logging

The KV store logs error and warning messages in internal logs, including splunkd.log and mongod.log. These error messages post to the bulletin board in Splunk Web. See What Splunk software logs about itself for an overview of internal log files.

Recent KV store error messages also appear in the REST `/services/messages` endpoint. You can use cURL to make a GET request for the endpoint, as follows:

```
curl -k -u user:pass https://<host>:<mPort>/services/messages
```

For more information about introspection endpoints, see System endpoint descriptions in the *REST API Reference Manual.*

Monitor KV store performance

You can monitor your KV store performance through two views in the monitoring console. One view provides insight across your entire deployment. See KV store: Deployment in *Monitoring Splunk Enterprise*.

The instance-scoped view gives you detailed information about KV store operations on each search head. See KV store: Instance in *Monitoring Splunk Enterprise*.

Meet Splunk apps

Apps and add-ons

Users often ask for definitions of app and add-on in an effort to determine what differentiates them from each other. There are no definitive criteria that universally distinguish an app from an add-on. Both are packaged sets of configuration that you install on your instance of Splunk Enterprise, and both make it easier to integrate with, or ingest data from, other technologies or vendors.

- **Apps** generally offer extensive user interfaces that enable you to work with your data, and they often make use of one or more add-ons to ingest different types of data.
- **Add-ons** generally enable Splunk Enterprise, or a Splunk app, to ingest or map a particular type of data.

To an Admin user, the difference should matter very little as both apps and add-ons function as tools to help you get data into Splunk Enterprise, then efficiently use it. To an app developer, the difference matters more: see dev.splunk.com for guidance on developing an app.

App

An **app** is an application that runs on Splunk Enterprise. Out of the box, Splunk Enterprise includes one basic, default app that enables you to work with your data: the Search and Reporting app. To address use cases beyond the basic, you can install many other apps, some free, some paid, on your instance of Splunk Enterprise. Examples include Splunk App for Microsoft Exchange, Splunk App for Enterprise Security, and Splunk DB Connect. An app may make use of one or more add-ons to facilitate how it collects or maps particular types of data.

Add-on

An **add-on** runs on Splunk Enterprise to provide specific capabilities to apps, such as getting data in, mapping data, or providing **saved searches** and macros. Examples include Splunk Add-on for Checkpoint OPSEC LEA, Splunk Add-on for Box, and Splunk Add-on for McAfee.

App and add-on support and certification

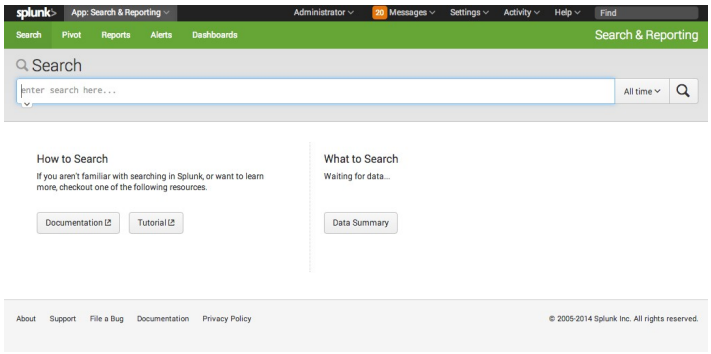
Anyone can develop an app or add-on for Splunk software. Splunk and members of our community create apps and add-ons and share them with other users of Splunk software via Splunkbase, the online app marketplace. Splunk *does not* support all apps and add-ons on Splunkbase. Labels in Splunkbase indicate who supports each app or add-on.

- The Splunk Support team accepts cases and responds to issues only for the apps and add-ons which display a **Splunk Supported** label on Splunkbase.
- Some developers support their own apps and add-ons. These apps and add-ons display a **Developer Supported** label on Splunkbase.
- The Splunk developer community supports apps and add-ons which display a **Community Supported** label on Splunkbase.

Further, app developers can obtain Splunk Certification for their app or add-on. This means that Splunk has examined an app or add-on and found that it conforms to best practices for Splunk development. Certification does not, however, mean that Splunk supports an app or add-on. For example, an add-on created by a community developer that is published on Splunkbase and certified by Splunk is not supported by Splunk. Look for a **Splunk Supported** label on Splunkbase to determine that Splunk supports an app or add-on.

Search and Reporting app

The first time you install and log into Splunk, you land in Splunk Home. The Home page displays Click on Apps in the apps that have been pre-installed for you.



By default, Splunk provides the Search and Reporting app. This interface provides the core functionality of Splunk and is designed for general-purpose use. This app displays at the top of your Home Page when you first log in and provides a search field so that you can immediately starting using it.

Once in the Search and Reporting app (by running a search or clicking on the app in the Home page) you can use the menu bar options to select the following:

- **Search:** Search your indexes. See the "Using Splunk Search" in the Search Tutorial for more information.
- **Pivot:** Use data models quickly design and generate tables, charts, and visualizations for your data. See the Pivot Manual for more information.
- **Reports:** Turn your searches into reports. "Saving and sharing reports" in the Search Tutorial for more information.
- **Alerts:** Set up alerts for your Splunk searches and reports. See the Alerting Manual for more information
- **Dashboards:** Leverage predefined dashboards or create your own. See Dashboards and Visualizations manual.

Configure Splunk Web to open in an app

You can configure Splunk Web so that it opens in a specific app of your choosing instead of Splunk Home. You can make Splunk Web open to open in a specific app for all users, or match apps to specific users.

Bypass Splunk Home for a single user

You can configure Splunk Web so that when a user logs in, they go straight to an app of your choosing, rather than Splunk Home.

To make the Search app the default landing app for a user:

1. Create a file called `user-prefs.conf` in the user's local directory:

```
etc/users/<user>/user-prefs/local/user-prefs.conf
```

- For the `admin` user the file would be in:

  ```
  etc/users/admin/user-prefs/local/user-prefs.conf
  ```

- For the `test` user, it would be in:

  ```
  etc/users/test/user-prefs/local/user-prefs.conf
  ```

2. Put the following line in the `user-prefs.conf` file:

```
default_namespace = search
```

Bypass Splunk Home for all users

You can specify a default app for all users to land in when they log in. For example, if you want the Search app to be the global default, edit `$SPLUNK_HOME/etc/apps/user-prefs/local/user-prefs.conf` and specify:

```
[general_default]
 default_namespace = search
```

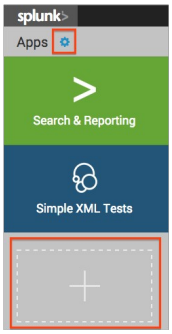
Note: Users who do not have permission to access the Search app will see an error.

Where to get more apps and add-ons

You can find new apps and add-ons on Splunkbase: **https://splunkbase.splunk.com/**. You can also browse new apps from the Splunk Enterprise home page.

If you are connected to the internet

If your Splunk Enterprise server or your client machine are connected to the internet, you can navigate to the app browser from the home page.



- You can click the **+** sign below your last installed app to go directly to the app browser.
- You can also click the gear next to **Apps** to go to the apps manager page. Click **Browse more apps** to go to the app browser.

Important: If Splunk Web is located behind a proxy server, you might have trouble accessing Splunkbase. To solve this problem, you need to set the `HTTP_PROXY` environment variable, as described in "Specify a proxy server".

If you are not connected to the internet

If your Splunk Enterprise server and client do not have internet connectivity, you must download apps from Splunkbase and copy them over to your server:

1. From a computer connected to the internet, browse Splunkbase for the app or add-on you want.

2. Download the app or add-on.

3. Once downloaded, copy it to your Splunk Enterprise server.

4. Put it in your `$SPLUNK_HOME/etc/apps` directory.

5. Untar and ungzip your app or add-on, using a tool like `tar -xvf` (on *nix) or WinZip (on Windows). Note that Splunk apps and add-ons are packaged with a .SPL extension although they are just tarred and gzipped. You may need to force your tool to recognize this extension.

6. You may need to restart Splunk Enterprise, depending on the contents of the app or add-on.

7. Your app or add-on is now installed and will be available from Splunk Home (if it has a web UI component).

App deployment overview

This topic provides an overview of the methods you can use to deploy Splunk apps and add-ons in common Splunk software environments.

For more detailed app and add-on deployment information, see your specific Splunk app documentation, or see "Where to install Splunk add-ons" in the *Splunk Add-ons* manual.

Prerequisites

You must have an existing Splunk platform deployment on which to install Splunk apps and add-ons.

Deployment methods

There are several ways to deploy apps and add-ons to the Splunk platform. The correct deployment method to use depends on the following characteristics of your specific Splunk software deployment:

- Deployment architecture (single-instance or distributed)
- Cluster types (search head clusters and/or indexer clusters)
- Location (on-premise or in Splunk Cloud)

Deployment architectures

There are two basic Splunk Enterprise deployment architectures:

- **Single-instance deployment:** In a single-instance deployment, one Splunk Enterprise instance acts as both search head and indexer.
- **Distributed deployment:** A distributed deployment can include multiple Splunk Enterprise **components**, including search heads, indexers, and forwarders. See "Scale your deployment with Splunk Enterprise components" in the *Distributed Deployment Manual*. A distributed deployment can also include standard individual components and/or clustered components, including search head clusters, indexer clusters, and multi-site clusters. See "Distributed Splunk Enterprise overview" in the *Distributed Deployment Manual*.

Single-instance deployment

To deploy an app on a single instance, download the app from **Splunkbase** to your local host, then install the app using **Splunk Web**.

Some apps currently do not support installation through Splunk Web. Make sure to check the installation instructions for your specific app prior to installation.

Distributed deployment

You can deploy apps in a distributed environment using the following methods:

- Install apps manually on each component using Splunk Web, or install apps manually from the command line.
- Install apps using the **deployment server**. The deployment server automatically distributes new apps, app updates, and certain configuration updates to search heads, indexers, and forwarders. See "About deployment server and forwarder management" in *Updating Splunk Enterprise Instances*.

Alternately, you can deploy apps using a third-party configuration management tool, such as:

- Chef
- Puppet
- Salt
- Windows configuration tools

For the most part, you must install Splunk apps on search heads, indexers, and forwarders. To determine the Splunk Enterprise components on which you must install the app, see the installation instructions for the specific app.

Deploy apps to clusters

Splunk distributed deployments can include these cluster types:

- **Search head clusters**
- **Indexer clusters**

You deploy apps to both indexer and search head cluster members using the **configuration bundle** method.

Search head clusters

To deploy apps to a search head cluster, you must use the **deployer**. The deployer is a Splunk Enterprise instance that distributes apps and configuration updates to search head cluster members. The deployer cannot be a search head cluster member and must exist outside the search head cluster. See "Use the deployer to distribute apps and configuration updates" in the *Distributed Search* manual.

Caution: Do not deploy a configuration bundle to a search head cluster from any instance other then the deployer. If you run the `apply schcluster-bundles` command on a non-deployer instance, such as a cluster member, the command deletes all existing apps and user-generated content on all search head cluster members!

Indexer clusters

To deploy apps to peer nodes (indexers) in an indexer cluster, you must first place the apps in the proper location on the indexer cluster master, then use the configuration bundle method to distribute the apps to peer nodes. You can apply the configuration bundle to peer nodes using Splunk Web or the CLI. For more information, see "Update common peer configurations and apps" in *Managing Indexers and Clusters of Indexers*.

While you cannot use the deployment server to deploy apps to peer nodes, you can use it to distribute apps to the indexer cluster master. For more information, see "Use deployment server to distribute apps to the master" in *Managing Indexers and Clusters of Indexers*.

Deploy apps to Splunk Cloud

If you want to deploy an app or add-on to Splunk Cloud, contact Splunk support for guidance. The support team can deploy the app or add-on on components of the deployment that are not exposed to Splunk Cloud subscribers.

Deploy add-ons to Splunk Light

You can install and enable a limited selection of add-ons to configure new data inputs on your instance of Splunk Light. See "Configure an add-on to add data" in the *Getting Started Manual* for Splunk Light.

App architecture and object ownership

Apps are commonly built from Splunk **knowledge objects**. Splunk knowledge objects include saved searches, event types, tags -- data types that enrich your Splunk deployment and make it easier to find what you need.

Note: Occasionally you may save objects to add-ons as well, though this is not common. Apps and add-ons are both stored in the apps directory. On the rare instance that you would need to save objects to an add-on, you would manage the add-on the same as described for apps in this topic.

Any user logged into Splunk Web can create and save knowledge objects to the user's directory under the app the user is "in" (assuming sufficient permissions). This is the default behavior -- whenever a user saves an object, it goes into the user's directory in the currently running app. The user directory is located at `$SPLUNK_HOME/etc/users/<user_name>/<app_name>/local`. Once the user has saved the object in that app, it is available only to that user when they are in that app unless they do one of the following:

- Promote the object so that it is available to all users who have access
- Restrict the object to specific roles or users (still within the app context)
- Mark the object as globally available to all apps, add-ons and users (unless you've explicitly restricted it by role/user)

Note: Users must have write permissions for an app or add-on before they can promote objects to that level.

Promote and share Splunk knowledge

Users can share their Splunk knowledge objects with other users through the Permissions dialog. This means users who have read permissions in an app or add-on can see the shared objects and use them. For example, if a user shares a saved search, other users can see that saved search, but only within the app in which the search was created. So if you create a saved search in the app "Fflanda" and share it, other users of Fflanda can see your saved search if they have read permission for Fflanda.

Users with write permission can promote their objects to the app level. This means the objects are copied from their user directory to the app's directory -- from:

```
$SPLUNK_HOME/etc/users/<user_name>/<app_name>/local/
```

to:

```
$SPLUNK_HOME/etc/apps/<app_name>/local/
```

Users can do this only if they have write permission in the app.

Make Splunk knowledge objects globally available

Finally, upon promotion, users can decide if they want their object to be available globally, meaning all apps are able to see it. Again, the user must have permission to write to the original app. It's easiest to do this in Splunk Web, but you can also do it later by moving the relevant object into the desired directory.

To make globally available an object "A" (defined in "B.conf") that belongs to user "C" in app "D":

1. Move the stanza defining the object A from `$SPLUNK_HOME/etc/users/C/D/B.conf` into `$SPLUNK_HOME/etc/apps/D/local/B.conf`.

2. Add a setting, `export = system`, to the object A's stanza in the app's `local.meta` file. If the stanza for that object doesn't already exist, you can just add one.

For example, to promote an event type called "rhallen" created by a user named "fflanda" in the *Nix app so that it is globally available:

1. Move the [rhallen] stanza from `$SPLUNK_HOME/etc/users/fflanda/unix/local/eventtypes.conf` to `$SPLUNK_HOME/etc/apps/unix/local/eventtypes.conf`.

2. Add the following stanza:

```
[eventtypes/rhallen]
export = system
```

to `$SPLUNK_HOME/etc/apps/unix/metadata/local.meta`.

Note: Adding the `export = system` setting to `local.meta` isn't necessary when you're sharing event types from the Search app, because it exports all of its events globally by default.

What objects does this apply to?

The knowledge objects discussed here are limited to those that are subject to access control. These objects are also known as app-level objects and can be viewed by selecting **Apps > Manage Apps** from the User menu bar. This page is available to all users to manage any objects they have created and shared. These objects include:

- Saved searches and Reports
- Event types
- Views and dashboards
- Field extractions

There are also system-level objects available only to users with admin privileges (or read/write permissions on the specific objects). These objects include:

- Users
- Roles
- Auth
- Distributed search
- Inputs
- Outputs
- Deployment
- License
- Server settings (for example: host name, port, etc)

Important: If you add an input, Splunk adds that input to the copy of `inputs.conf` that belongs to the app you're currently in. This means that if you navigated to your app directly from Search, your input will be added to `$SPLUNK_HOME/etc/apps/search/local/inputs.conf`, which might not be the behavior you desire.

App configuration and knowledge precedence

When you add knowledge to Splunk, it's added in the context of the app you're in when you add it. When Splunk is evaluating configurations and knowledge, it evaluates them in a specific order of precedence, so that you can control what knowledge definitions and configurations are used in what context. Refer to About configuration files for more information about Splunk configuration files and the order of precedence.

Manage app and add-on objects

When an **app** or **add-on** is created by a Splunk user, a collection of objects is created that make up the app or add-on. These objects can include **views**, commands, navigation items, **event types**, **saved searches**, **reports**, and more. Each of these objects have permissions associated with them to determine who can view or alter them. By default, the admin user has **permissions** to alter all the objects in the Splunk system.

Refer to these topics for more information:

- For an overview of apps and add-ons, refer to "What are apps and add-ons?" in this manual.
- For more information about app and add-on permissions, refer to "App architecture and object ownership" in this manual.
- To learn more about how to create your own apps and add-ons, refer to the Developing Views and Apps for Splunk Web manual.

View and manage app or add-on objects in Splunk Web

You can use Splunk Web to view the objects in your Splunk deployment in the following ways:

- To see all the objects for all the apps/add-ons on your system at once: **Settings > All configurations**.
- To see all the saved searches and report objects: **Settings > Searches and reports**.
- To see all the event types: **Settings > Event types**.
- To see all the field extractions: **Settings > Fields**.

You can:

- View and manipulate the objects on any page with the **sorting arrows** ⇕
- Filter the view to see only the objects from a given app or add-on, owned by a particular user, or those that contain a certain string, with the **App context bar**.

Use the Search field on the App context bar to search for strings in fields. By default, Splunk searches for the string in all available fields. To search within a particular field, specify that field. Wildcards are supported.

Note: For information about the individual search commands on the Search command page, refer to the **Search Reference Manual**.

Update an app or add-on in the CLI

To update an existing app on your Splunk instance using the CLI:

```
./splunk install app <app_package_filename> -update 1 -auth
<username>:<password>
```

Splunk updates the app or add-on based on the information found in the installation package.

Disable an app or add-on using the CLI

To disable an app via the CLI:

```
./splunk disable app [app_name] -auth <username>:<password>
```

Note: If you are running Splunk Free, you do not have to provide a username and password.

Uninstall an app or add-on

To remove an installed app from a Splunk installation:

1. (Optional) Remove the app or add-on's indexed data. Typically, Splunk does not access indexed data from a deleted app or add-on. However, you can use Splunk's CLI clean command to remove indexed data from an app before deleting the app. See Remove data from indexes with the CLI command.

2. Delete the app and its directory. This should be located in `$SPLUNK_HOME/etc/apps/<appname>`. You can run the following command in the CLI:

./splunk remove app [appname] -auth <username>:<password>
3. You may need to remove user-specific directories created for your app or add-on by deleting the files (if any) found here:
`$SPLUNK_HOME/splunk/etc/users/*/<appname>`

4. Restart Splunk.

Managing app and add-on configurations and properties

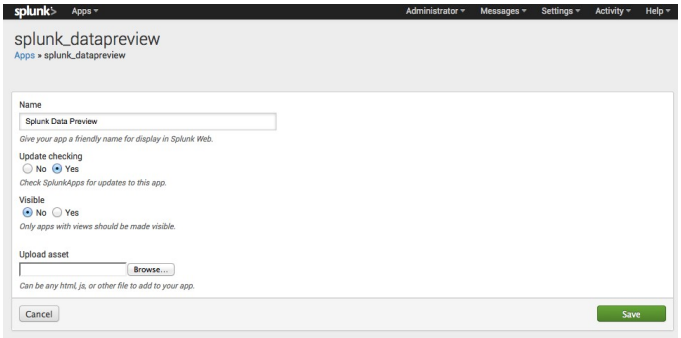
You can manage the configurations and properties for apps installed in your Splunk Enterprise instance from the Apps menu. Click on **Apps** in the User bar to select one of your installed apps or manage an app. From the Manage Apps page, you can do the following:

- Edit permissions for an app or add-on
- Enable or disable an app or add-on
- Perform actions, such as launch the app, edit the properties, and view app objects

Edit app and add-on properties

The edits you make to configuration and properties depend on whether you are the owner of the app or a user.

Select **Apps > Manage Apps** then click **Edit properties** for the app or add-on you want to edit. You can make the following edits for apps installed in this Splunk Enterprise instance.



- **Name:** Change the display name of the app or add-on in Splunk Web.
- **Update checking:** By default, update checking is enabled. You can override the default and disable update checking. See Checking for app an add-on updates below for details.
- **Visible:** Apps with views should be visible. Add-ons, which often do not have a view, should disable the visible property.

- **Upload asset:** Use this field to select a local file asset files, such as an HTML, JavaScript, or CSS file that can be accessed by the app or add-on. You can only upload one file at a time from this panel.

Refer to Develop Splunk apps on the Splunk Developer Portal for details on the configuration and properties of apps and add-ons.

Checking for updates

You can configure Splunk Enterprise whether to check Splunkbase for updates to an app or add-on. By default, checking for updates is enabled. You can disable checking for updates for an app by editing this property from **Settings > Apps > Edit properties**.

However, if this property is not available in Splunk Web, you can also manually edit the apps `app.conf` file to disable checking for updates. Create or edit the following stanza in `$SPLUNK_HOME/etc/apps/<app_name>/local/app.conf` to disable checking for updates:

```
[package]
check_for_updates = 0
```

Note: Edit the local version of `app.conf`, not the default version. This avoids overriding your setting with the next update of the app.

Splunk and Hadoop

About Splunk Analytics for Hadoop

Splunk Enterprise lets you configure remote HDFS datastores as virtual indexes so that Splunk can natively report on data residing in Hadoop. Once your virtual index is properly configured, you can report and visualize data residing in remote Hadoop datastores.

To learn more, see Meet Splunk Analytics for Hadoop.

Manage users

About users and roles

You can create users with passwords and assign them to **roles** that you have created. Splunk Enterprise Free does not support user authentication.

Splunk Enterprise comes with a single default user, the **admin** user. The default password for the admin user is **changeme**. As the password implies, you should change this password immediately after you install the software.

Create users

Splunk Enterprise supports three types of authentication systems, which are described in the *Securing Splunk Enterprise* manual.

- **Native authentication.** See "Set up user authentication with Splunk Enterprise native authentication" for more information.
- **LDAP.** Splunk supports authentication with its internal authentication services or your existing LDAP server. See "Set up user authentication with LDAP" for more information.
- **Scripted authentication API.** Use scripted authentication to connect Splunk native authentication with an external authentication system, such as RADIUS or PAM. See "Set up user authentication with external systems" for more information.

About roles

Users are assigned to roles. A role contains a set of **capabilities**. Capabilities specify what actions are available to roles. For example, capabilities determine whether someone with a particular role is allowed to add inputs or edit saved searches. The various capabilities are listed in "About defining roles with capabilities" in the *Securing Splunk Enterprise* manual.

By default, Splunk Enterprise comes with the following roles predefined:

- admin -- this role has the most capabilities assigned to it.
- power -- this role can edit all shared objects (saved searches, etc) and

alerts, tag events, and other similar tasks.
- user -- this role can create and edit its own saved searches, run searches, edit its own preferences, create and edit event types, and other similar tasks.
- can_delete -- This role allows the user to delete by keyword. This capability is necessary when using the delete search operator.

Note Do not edit the predefined roles. Instead, create custom roles that inherit from the built-in roles, and modify the custom roles as required.

For detailed information on roles and how to assign users to roles, see the chapter "Users and role-based access control" in the *Securing Splunk Enterprise* manual.

Find existing users and roles

To locate an existing user or role in Splunk Web, use the Search bar at the top of the Users or Roles page in the Access Controls section by selecting **Settings > Access Controls**. Wildcards are supported. By default Splunk Enterprise searches in all available fields for the string that you enter. To search a particular field, specify that field. For example, to search only email addresses, type "email=<*email address or address fragment*>:, or to search only the "Full name" field, type "realname=<*name or name fragment*>. To search for users in a given role, use "roles=".

Configure user language and locale

When a user logs in, Splunk automatically uses the language that the user's browser is set to. To switch languages, change the browser's locale setting. Locale configurations are browser-specific.

Splunk detects locale strings. A locale string contains two components: a language specifier and a localization specifier. This is usually presented as two lowercase letters and two uppercase letters linked by an underscore. For example, "en_US" means US English and "en_GB" means British English.

The user's locale also affects how dates, times, numbers, etc., are formatted, as different countries have different standards for formatting these entities.

Splunk provides built-in support for these locales:

```
de_DE
en_GB
en_US
fr_FR
it_IT
ja_JP
ko_KR
zh_CN
zh_TW
```

If you want to add localization for additional languages, refer to "Translate Splunk" in the Developer manual for guidance. You can then tell your users to specify the appropriate locale in their browsers.

How browser locale affects timestamp formatting

By default, timestamps in Splunk are formatted according the browser locale. If the browser is configured for US English, the timestamps are presented in American fashion: `MM/DD/YYYY:HH:MM:SS`. If the browser is configured for British English, then the timestamps will be presented in the European date format: `DD/MM/YYYY:HH:MM:SS`.

For more information on timestamp formatting, see "Configure timestamp recognition" in the Getting Data In manual.

Override the browser locale

The locale that Splunk uses for a given session can be changed by modifying the url that you use to access Splunk. Splunk urls follow the form `http://host:port/locale/...`. For example, when you access Splunk to log in, the url may appear as `http://hostname:8000/en-US/account/login` for US English. To use British English settings, you can change the locale string to `http://hostname:8000/en-GB/account/login`. This session then presents and accepts timestamps in British English format for its duration.

Requesting a locale for which the Splunk interface has not been localized results in the message: `Invalid language Specified`.

Refer to "Translate Splunk" in the Developer Manual for more information about localizing Splunk.

Configure user session timeouts

The amount of time that elapses before a Splunk user's session times out depends on the interaction among three timeout settings:

- The `splunkweb` session timeout.
- The `splunkd` session timeout.
- The browser session timeout.

The `splunkweb` and `splunkd` timeouts determine the maximum idle time in the interaction between browser and Splunk. The browser session timeout determines the maximum idle time in interaction between user and browser.

The `splunkweb` and `splunkd` timeouts generally have the same value, as the same field sets both of them. To set the timeout in Splunk Web:

1. Click **Settings** in the upper right-hand corner of Splunk Web.

2. Under System, click **Server settings**.

3. Click **General settings**.

4. In the **Session timeout** field, enter a timeout value.

5. Click **Save**.

This sets the user session timeout value for both `splunkweb` and `splunkd`. Initially, they share the same value of 60 minutes. They will continue to maintain identical values if you change the value through Splunk Web.

If, for some reason, you need to set the timeouts for `splunkweb` and `splunkd` to different values, you can do so by editing their underlying configuration files, `web.conf` (`tools.sessions.timeout` attribute) and `server.conf` (`sessionTimeout` attribute). For all practical purposes, there's no reason to give them different values. In any case, if the user is using SplunkWeb (`splunkweb`) to access the Splunk instance (`splunkd`), the smaller of the two timeout attributes prevails. So, if `tools.sessions.timeout` in `web.conf` has a value of "90" (minutes), and `sessionTimeout` in `server.conf` has a value of "1h" (1 hour; 60 minutes), the session will timeout after 60 minutes.

In addition to setting the `splunkweb`/`splunkd` session value, you can also specify the timeout for the user browser session by editing the `ui_inactivity_timeout` value in `web.conf`. The Splunk browser session will time out once this value is

reached. The default is 60 minutes. If `ui_inactivity_timeout` is set to less than 1, there's no timeout -- the session will stay alive while the browser is open.

The countdown for the `splunkweb`/`splunkd` session timeout does not begin until the browser session reaches its timeout value. So, to determine how long the user has before timeout, add the value of `ui_inactivity_timeout` to the smaller of the timeout values for `splunkweb` and `splunkd`. For example, assume the following:

- `splunkweb` timeout: 15m
- `splunkd` timeout: 20m
- browser (`ui_inactivity_timeout`) timeout: 10m

The user session stays active for 25m (15m+10m). After 25 minutes of no activity, the user will be prompted to login again.

Note: If you change a timeout value, either in Splunk Web or in configuration files, you must restart Splunk for the change to take effect.

Configuration file reference

alert_actions.conf

The following are the spec and example files for alert_actions.conf.

alert_actions.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attributes and values for configuring
global
# saved search actions in alert_actions.conf.  Saved searches are
configured
# in savedsearches.conf.
#
# There is an alert_actions.conf in $SPLUNK_HOME/etc/system/default/.
# To set custom configurations, place an alert_actions.conf in
# $SPLUNK_HOME/etc/system/local/.  For examples, see
# alert_actions.conf.example. You must restart Splunk to enable
# configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#  * You can also define global settings outside of any stanza, at the
top
#    of the file.
#  * Each conf file should have at most one default stanza. If there
are
#    multiple default stanzas, attributes are combined. In the case of
#    multiple definitions of the same attribute, the last definition in
the
#    file wins.
#  * If an attribute is defined at both the global level and in a
specific
#    stanza, the value in the specific stanza takes precedence.
```

```
maxresults = <integer>
* Set the global maximum number of search results sent via alerts.
* Defaults to 100.

hostname = [protocol]<host>[:<port>]
* Sets the hostname used in the web link (url) sent in alerts.
* This value accepts two forms.
  * hostname
       examples: splunkserver, splunkserver.example.com
  * protocol://hostname:port
       examples: http://splunkserver:8000,
https://splunkserver.example.com:443
* When this value is a simple hostname, the protocol and port which
  are configured within splunk are used to construct the base of
  the url.
* When this value begins with 'http://', it is used verbatim.
  NOTE: This means the correct port must be specified if it is not
  the default port for http or https.
* This is useful in cases when the Splunk server is not aware of
  how to construct an externally referenceable url, such as SSO
  environments, other proxies, or when the Splunk server hostname
  is not generally resolvable.
* Defaults to current hostname provided by the operating system,
  or if that fails, "localhost".
* When set to empty, default behavior is used.

ttl     = <integer>[p]
* Optional argument specifying the minimum time to live (in seconds)
  of the search artifacts, if this action is triggered.
* If p follows integer, then integer is the number of scheduled periods.
* If no actions are triggered, the artifacts will have their ttl
determined
  by the "dispatch.ttl" attribute in savedsearches.conf.
* Defaults to 10p
* Defaults to 86400 (24 hours)   for: email, rss
* Defaults to   600 (10 minutes) for: script
* Defaults to   120 (2 minutes)  for: summary_index, populate_lookup

maxtime = <integer>[m|s|h|d]
* The maximum amount of time that the execution of an action is allowed
to
  take before the action is aborted.
* Use the d, h, m and s suffixes to define the period of time:
  d = day, h = hour, m = minute and s = second.
  For example: 5d means 5 days.
* Defaults to 5m for everything except rss.
* Defaults to 1m for rss.

track_alert = [1|0]
* Indicates whether the execution of this action signifies a trackable
alert.
```

```
* Defaults to 0 (false).

command = <string>
* The search command (or pipeline) which is responsible for executing
  the action.
* Generally the command is a template search pipeline which is realized
  with values from the saved search - to reference saved search
  field values wrap them in dollar signs ($).
* For example, to reference the savedsearch name use $name$. To
  reference the search, use $search$

is_custom = [1|0]
* Specifies whether the alert action is based on the custom alert
  actions framework and is supposed to be listed in the search UI.

payload_format = [xml|json]
* Configure the format the alert script receives the configuration via
  STDIN.
* Defaults to "xml"

label = <string>
* For custom alert actions: Define the label shown in the UI. If not
  specified, the stanza name will be used instead.

description = <string>
* For custom alert actions: Define the description shown in the UI.

icon_path = <string>
* For custom alert actions: Define the icon shown in the UI for the
alert
  action. The path refers to appserver/static within the app where the
  alert action is defined in.

alert.execute.cmd = <string>
* For custom alert actions: Explicitly specify the command to be
executed
  when the alert action is triggered. This refers to a binary or script
  in the bin folder of the app the alert action is defined in, or to a
  path pointer file, also located in the bin folder.
* If a path pointer file (*.path) is specified, the contents of the file
  is read and the result is used as the command to be executed.
  Environment variables in the path pointer file are substituted.
* If a python (*.py) script is specified it will be prefixed with the
  bundled python interpreter.

alert.execute.cmd.arg.<n> = <string>
* Provide additional arguments to the alert action execution command.
  Environment variables are substituted.

############################################################################
# EMAIL: these settings are prefaced by the [email] stanza name
############################################################################
```

[email]

```
[email]
* Set email notification options under this stanza name.
* Follow this stanza name with any number of the following
  attribute/value pairs.
* If you do not specify an entry for each attribute, Splunk will
  use the default value.

from = <string>
* Email address from which the alert originates.
* Defaults to splunk@$LOCALHOST.

to      = <string>
* The To email address receiving the alert.

cc      = <string>
* Any cc email addresses receiving the alert.

bcc     = <string>
* Any bcc email addresses receiving the alert.

message.report = <string>
* Specify a custom email message for scheduled reports.
* Includes the ability to reference attributes from the result,
  saved search, or job

message.alert = <string>
* Specify a custom email message for alerts.
* Includes the ability to reference attributes from result,
  saved search, or job

subject = <string>
* Specify an alternate email subject if useNSSubject is false.
* Defaults to SplunkAlert-<savedsearchname>.

subject.alert = <string>
* Specify an alternate email subject for an alert.
* Defaults to SplunkAlert-<savedsearchname>.

subject.report = <string>
* Specify an alternate email subject for a scheduled report.
* Defaults to SplunkReport-<savedsearchname>.

useNSSubject = [1|0]
* Specify whether to use the namespaced subject (i.e subject.report) or
  subject.

footer.text = <string>
* Specify an alternate email footer.
```

```
* Defaults to "If you believe you've received this email in error,
please see your Splunk administrator.\r\n\r\nsplunk > the engine for
machine data."

format = [table|raw|csv]
* Specify the format of inline results in the email.
* Accepted values:  table, raw, and csv.
* Previously accepted values plain and html are no longer respected
  and equate to table.
* To make emails plain or html use the content_type attribute.

include.results_link = [1|0]
* Specify whether to include a link to the results.

include.search = [1|0]
* Specify whether to include the search that caused an email to be sent.

include.trigger = [1|0]
* Specify whether to show the trigger condition that caused the alert to
  fire.

include.trigger_time = [1|0]
* Specify whether to show the time that the alert was fired.

include.view_link = [1|0]
* Specify whether to show the title and a link to enable the user to
edit
  the saved search.

content_type = [html|plain]
* Specify the content type of the email.
  * plain sends email as plain text
  * html sends email as a multipart email that include both text and
html.

sendresults = [1|0]
* Specify whether the search results are included in the email. The
  results can be attached or inline, see inline (action.email.inline)
* Defaults to 0 (false).

inline = [1|0]
* Specify whether the search results are contained in the body of the
alert
  email.
* If the events are not sent inline, they are attached as a csv text.
* Defaults to 0 (false).

priority = [1|2|3|4|5]
* Set the priority of the email as it appears in the email client.
* Value mapping: 1 highest, 2 high, 3 normal, 4 low, 5 lowest.
* Defaults to 3.
```

```
mailserver = <host>[:<port>]
* You must have a Simple Mail Transfer Protocol (SMTP) server available
  to send email. This is not included with Splunk.
* Specifies the SMTP mail server to use when sending emails.
* <host> can be either the hostname or the IP address.
* Optionally, specify the SMTP <port> that Splunk should connect to.
* When the "use_ssl" attribute (see below) is set to 1 (true), you
  must specify both <host> and <port>.
  (Example: "example.com:465")
* Defaults to $LOCALHOST:25.

use_ssl    = [1|0]
* Whether to use SSL when communicating with the SMTP server.
* When set to 1 (true), you must also specify both the server name or
  IP address and the TCP port in the "mailserver" attribute.
* Defaults to 0 (false).

use_tls    = [1|0]
* Specify whether to use TLS (transport layer security) when
  communicating with the SMTP server (starttls)
* Defaults to 0 (false).

auth_username   = <string>
* The username to use when authenticating with the SMTP server. If this
is
  not defined or is set to an empty string, no authentication is
attempted.
  NOTE: your SMTP server might reject unauthenticated emails.
* Defaults to empty string.

auth_password   = <password>
* The password to use when authenticating with the SMTP server.
  Normally this value will be set when editing the email settings,
however
  you can set a clear text password here and it will be encrypted on
the
  next Splunk restart.
* Defaults to empty string.

sendpdf = [1|0]
* Specify whether to create and send the results as a PDF.
* Defaults to 0 (false).

sendcsv = [1|0]
* Specify whether to create and send the results as a csv file.
* Defaults to 0 (false).

pdfview = <string>
* Name of view to send as a PDF

reportPaperSize = [letter|legal|ledger|a2|a3|a4|a5]
* Default paper size for PDFs
```

```
* Accepted values: letter, legal, ledger, a2, a3, a4, a5
* Defaults to "letter".

reportPaperOrientation = [portrait|landscape]
* Paper orientation: portrait or landscape
* Defaults to "portrait".

reportIncludeSplunkLogo = [1|0]
* Specify whether to include a Splunk logo in Integrated PDF Rendering
* Defaults to 1 (true)

reportCIDFontList = <string>
* Specify the set (and load order) of CID fonts for handling
  Simplified Chinese(gb), Traditional Chinese(cns),
  Japanese(jp), and Korean(kor) in Integrated PDF Rendering.
* Specify in a space-separated list
* If multiple fonts provide a glyph for a given character code, the
glyph
  from the first font specified in the list will be used
* To skip loading any CID fonts, specify the empty string
* Defaults to "gb cns jp kor"

reportFileName = <string>
    * Specify the name of attached pdf or csv
    * Defaults to "$name$-$time:%Y-%m-%d$"

width_sort_columns = <bool>
* Whether columns should be sorted from least wide to most wide left to
right.
* Valid only if format=text
* Defaults to true

preprocess_results = <search-string>
* Supply a search string to Splunk to preprocess results before emailing
  them. Usually the preprocessing consists of filtering out unwanted
  internal fields.
* Defaults to empty string (no preprocessing)

pdf.footer_enabled = [1 or 0]
  * Set whether or not to display footer on PDF.
  * Defaults to 1.

pdf.header_enabled = [1 or 0]
  * Set whether or not to display header on PDF.
  * Defaults to 1.

pdf.logo_path = <string>
  * Define pdf logo by syntax <app>:<path-to-image>
  * If set, PDF will be rendered with this logo instead of Splunk one.
  * If not set, Splunk logo will be used by default
  * Logo will be read from
$SPLUNK_HOME/etc/apps/<app>/appserver/static/<path-to-image> if <app>
```

```
is provided.
  * Current app will be used if <app> is not provided.

pdf.header_left = [logo|title|description|timestamp|pagination|none]
  * Set which element will be displayed on the left side of header.
  * Nothing will be display if this option is not been set or set to
none
  * Defaults to None, nothing will be displayed on this position.

pdf.header_center = [logo|title|description|timestamp|pagination|none]
  * Set which element will be displayed on the center of header.
  * Nothing will be display if this option is not been set or set to
none
  * Defaults to description

pdf.header_right = [logo|title|description|timestamp|pagination|none]
  * Set which element will be displayed on the right side of header.
  * Nothing will be display if this option is not been set or set to
none
  * Defaults to None, nothing will be displayed on this position.

pdf.footer_left = [logo|title|description|timestamp|pagination|none]
  * Set which element will be displayed on the left side of footer.
  * Nothing will be display if this option is not been set or set to
none
  * Defaults to logo

pdf.footer_center = [logo|title|description|timestamp|pagination|none]
  * Set which element will be displayed on the center of footer.
  * Nothing will be display if this option is not been set or set to
none
  * Defaults to title

pdf.footer_right = [logo|title|description|timestamp|pagination|none]
  * Set which element will be displayed on the right side of footer.
  * Nothing will be display if this option is not been set or set to
none
  * Defaults to timestamp,pagination

pdf.html_image_rendering = <bool>
  * Whether images in HTML should be rendered.
  * If enabling rendering images in HTML breaks the pdf for whatever
reason,
  * it could be disabled by setting this flag to False,
  * so the old HTML rendering will be used.
  * Defaults to True.

sslVersions = <versions_list>
* Comma-separated list of SSL versions to support.
* The versions available are "ssl3", "tls1.0", "tls1.1", and "tls1.2".
* The special version "*" selects all supported versions.  The version
"tls"
```

```
  selects all versions tls1.0 or newer.
* If a version is prefixed with "-" it is removed from the list.
* SSLv2 is always disabled; "-ssl2" is accepted in the version list but
does nothing.
* When configured in FIPS mode, ssl3 is always disabled regardless
  of this configuration.
* Defaults to "*,-ssl2" (anything newer than SSLv2).

sslVerifyServerCert = true|false
* If this is set to true, you should make sure that the server that is
  being connected to is a valid one (authenticated).  Both the common
  name and the alternate name of the server are then checked for a
  match if they are specified in this configuration file.  A
  certificiate is considered verified if either is matched.
* If this is set to true, make sure
'server.conf/[sslConfig]/sslRootCAPath'
  has been set correctly.
* Default is false.

sslCommonNameToCheck = <commonName1>, <commonName2>, ...
* Optional. Defaults to no common name checking.
* Check the common name of the server's certificate against this list of
names.
* 'sslVerifyServerCert' must be set to true for this setting to work.

sslAltNameToCheck =  <alternateName1>, <alternateName2>, ...
* Optional. Defaults to no alternate name checking.
* Check the alternate name of the server's certificate against this list
of names.
* If there is no match, assume that Splunk is not authenticated against
this
  server.
* 'sslVerifyServerCert' must be set to true for this setting to work.

cipherSuite = <cipher suite string>
* If set, Splunk uses the specified cipher string for the communication
with
  with the SMTP server.
* If not set, Splunk uses the default cipher string provided by
OpenSSL.
* This is used to ensure that the client does not make connections
using
  weak encryption protocols.
* Default is 'TLSv1+HIGH:TLSv1.2+HIGH:@STRENGTH'.

############################################################################
# RSS: these settings are prefaced by the [rss] stanza
############################################################################
```

[rss]

```
[rss]
* Set RSS notification options under this stanza name.
* Follow this stanza name with any number of the following
  attribute/value pairs.
* If you do not specify an entry for each attribute, Splunk will
  use the default value.

items_count = <number>
* Number of saved RSS feeds.
* Cannot be more than maxresults (in the global settings).
* Defaults to 30.

############################################################################
# script: Used to configure any scripts that the alert triggers.
############################################################################
```

[script]

```
[script]
filename = <string>
* The filename, with no path, of the script to trigger.
* The script should be located in: $SPLUNK_HOME/bin/scripts/
* For system shell scripts on Unix, or .bat or .cmd on windows, there
  are no further requirements.
* For other types of scripts, the first line should begin with a #!
  marker, followed by a path to the interpreter that will run the
script.
  * Example: #!C:\Python27\python.exe
* Defaults to empty string.

############################################################################
# summary_index: these settings are prefaced by the [summary_index]
stanza
############################################################################
```

[summary_index]

```
[summary_index]
inline = [1|0]
* Specifies whether the summary index search command will run as part of
the
  scheduled search or as a follow-on action. This is useful when the
results
  of the scheduled search are expected to be large.
* Defaults to 1 (true).
```

```
_name = <string>
* The name of the summary index where Splunk will write the events.
* Defaults to "summary".

############################################################################
# populate_lookup: these settings are prefaced by the [populate_lookup]
stanza
############################################################################
```

[populate_lookup]

```
[populate_lookup]
dest = <string>
* Name of the lookup table to populate (stanza name in transforms.conf)
or
  the lookup file path to where you want the data written. If a path is
  specified it MUST be relative to $SPLUNK_HOME and a valid lookups
  directory.
  For example: "etc/system/lookups/<file-name>" or
  "etc/apps/<app>/lookups/<file-name>"
* The user executing this action MUST have write permissions to the app
for
  this action to work properly.
```

alert_actions.conf.example

```
#   Version 6.5.0
#
# This is an example alert_actions.conf.  Use this file to configure
alert
# actions for saved searches.
#
# To use one or more of these configurations, copy the configuration
block into
# alert_actions.conf in $SPLUNK_HOME/etc/system/local/.  You must
restart
# Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

[email]
# keep the search artifacts around for 24 hours
```

```
ttl = 86400

# if no @ is found in the address the hostname of the current machine is
appended
from = splunk

format = table

inline = false

sendresults = true

hostname = CanAccessFromTheWorld.com

command = sendemail "to=$action.email.to$"
"server=$action.email.mailserver{default=localhost}$"
"from=$action.email.from{default=splunk@localhost}$"
"subject=$action.email.subject{recurse=yes}$"
"format=$action.email.format{default=csv}$" "sssummary=Saved Search
[$name$]: $counttype$($results.count$)" "sslink=$results.url$"
"ssquery=$search$" "ssname=$name$"
"inline=$action.email.inline{default=False}$"
"sendresults=$action.email.sendresults{default=False}$"
"sendpdf=$action.email.sendpdf{default=False}$"
"pdfview=$action.email.pdfview$" "searchid=$search_id$"
"graceful=$graceful{default=True}$"
maxinputs="$maxinputs{default=1000}$"
maxtime="$action.email.maxtime{default=5m}$"
_validate-1 = action.email.sendresults, validate(
is_bool('action.email.sendresults'), "Value of argument
'action.email.sendresults' must be a boolean")

use_tls = 1
sslVersions = tls1.2
sslVerifyServerCert = true
sslCommonNameToCheck = host1, host2

[rss]
# at most 30 items in the feed
items_count=30

# keep the search artifacts around for 24 hours
ttl = 86400

command = createrss "path=$name$.xml" "name=$name$"
"link=$results.url$" "descr=Alert trigger: $name$,
results.count=$results.count$ " "count=30"
"graceful=$graceful{default=1}$"
maxtime="$action.rss.maxtime{default=1m}$"

[summary_index]
# don't need the artifacts anytime after they're in the summary index
```

```
ttl = 120

# make sure the following keys are not added to marker (command, ttl,
maxresults, _*)
command = summaryindex addtime=true
index="$action.summary_index._name{required=yes}$"
file="$name$_$#random$.stash" name="$name$"
marker="$action.summary_index*{format=$KEY=\\\"$VAL\\\",
key_regex="action.summary_index.(?!(?:command|maxresults|ttl|(?:_.*))$)(.*)"}$"

[custom_action]
# flag the action as custom alert action
is_custom = 1

# configure appearance in the UI
label = Custom Alert Action
description = Triggers a custom alert action
icon_path = custom_alert.png

# override default script execution
# java.path is a path pointer file in <app>/bin pointing to the actual
java executable
alert.execute.cmd = java.path
alert.execute.cmd.arg.1 = -jar
alert.execute.cmd.arg.2 = $SPLUNK_HOME/etc/apps/myapp/bin/custom.jar
alert.execute.cmd.arg.3 = --execute
```

app.conf

The following are the spec and example files for app.conf.

app.conf.spec

```
#   Version 6.5.0
#
# This file maintains the state of a given app in Splunk Enterprise. It
may also be used
# to customize certain aspects of an app.
#
# There is no global, default app.conf. Instead, an app.conf may exist
in each
# app in Splunk Enterprise.
#
# You must restart Splunk Enterprise to reload manual changes to
app.conf.
#
```

```
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

#
# Settings for how an app appears in Launcher (and online on Splunkbase)
#
```

[launcher]

```
[launcher]
# global setting

remote_tab = <bool>
* Set whether the Launcher interface will connect to apps.splunk.com.
* This setting only applies to the Launcher app and should be not set in
any
  other app
* Defaults to true.

# per-application settings

version = <version string>
* Version numbers are a number followed by a sequence of dots and
numbers.
* Version numbers for releases should use three digits.
* Pre-release versions can append a single-word suffix like "beta" or
"preview."
* Pre-release designations should use lower case and no spaces.
* Examples:
  * 1.2.0
  * 3.2.1
  * 11.0.34
  * 2.0beta
  * 1.3beta2
  * 1.0preview

description = <string>
* Short explanatory string displayed underneath the app's title in
Launcher.
* Descriptions should be 200 characters or less because most users won't
read
  long descriptions!

author = <name>
* For apps you intend to post to Splunkbase, enter the username of your
  splunk.com account.
* For internal-use-only apps, include your full name and/or contact
```

```
info
  (e.g. email).

# Your app can include an icon which will show up next to your app in
Launcher
# and on Splunkbase. You can also include a screenshot, which will show
up on
# Splunkbase when the user views info about your app before downloading
it.
# Icons are recommended, although not required.
# Screenshots are optional.
#
# There is no setting in app.conf for these images. Instead, icon and
# screenshot images should be placed in the appserver/static dir of
# your app. They will automatically be detected by Launcher and
Splunkbase.
#
# For example:
#
#     <app_directory>/appserver/static/appIcon.png    (the capital "I"
is required!)
#     <app_directory>/appserver/static/screenshot.png
#
# An icon image must be a 36px by 36px PNG file.
# An app screenshot must be 623px by 350px PNG file.

#
# [package] defines upgrade-related metadata, and will be
# used in future versions of Splunk Enterprise to streamline app
upgrades.
#
```

[package]

```
[package]
id = <appid>
* id should be omitted for internal-use-only apps which are not
intended to be
  uploaded to Splunkbase
* id is required for all new apps uploaded to Splunkbase. Future
versions of
  Splunk Enterprise will use appid to correlate locally-installed apps
and the
  same app on Splunkbase (e.g. to notify users about app updates)
* id must be the same as the folder name in which your app lives in
  $SPLUNK_HOME/etc/apps
* id must adhere to cross-platform folder-name restrictions:
  * must contain only letters, numbers, "." (dot), and "_" (underscore)
characters
  * must not end with a dot character
```

```
  * must not be any of the following names: CON, PRN, AUX, NUL,
      COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9,
      LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, LPT9

check_for_updates = <bool>
* Set whether Splunk Enterprise should check Splunkbase for updates to
this app.
* Defaults to true.


#
# Set install settings for this app
#
```

[install]

```
[install]
state = disabled | enabled
* Set whether app is disabled or enabled.
* If an app is disabled, its configs are ignored.
* Defaults to enabled.

state_change_requires_restart = true | false
* Set whether changing an app's state ALWAYS requires a restart of
Splunk Enterprise.
* State changes include enabling or disabling an app.
* When set to true, changing an app's state always requires a restart.
* When set to false, modifying an app's state may or may not require a
restart
  depending on what the app contains. This setting cannot be used to
avoid all
  restart requirements!
* Defaults to false.

is_configured = true | false
* Stores indication of whether the application's custom setup has been
performed
* Defaults to false

build = <integer>
* Required.
* Must be a positive integer.
* Increment this whenever you change files in appserver/static.
* Every release must change both "version" and "build" settings.
* Ensures browsers don't use cached copies of old static files
  in new versions of your app.
* Build is a single integer, unlike version which can be a complex
string
  like 1.5.18.
```

```
allows_disable = true | false
* Set whether an app allows itself to be disabled.
* Defaults to true.

install_source_checksum = <string>
* Records a checksum of the tarball from which a given app was
installed.
* Splunk Enterprise will automatically populate this value upon install.
* You should *not* set this value explicitly within your app!

#
# Handle reloading of custom .conf files (4.2+ versions only)
#
```

[triggers]

```
[triggers]
reload.<conf_file_name> = [ simple | rest_endpoints | access_endpoints
<handler_url> | http_get <handler_url> | http_post <handler_url> ]
* Splunk Enterprise will reload app configuration after every app-state
change:
  install, update, enable, and disable.
* If your app does not use a custom config file (e.g. myconffile.conf)
  then it won't need a [triggers] stanza, because
  $SPLUNK_HOME/etc/system/default/app.conf already includes a
[triggers]
  stanza which automatically reloads config files normally used by
Splunk Enterprise.
* If your app uses a custom config file (e.g. myconffile.conf) and you
want to
  avoid unnecessary Splunk Enterprise restarts, you'll need to add a
reload value in
  the [triggers] stanza.
* If you don't include [triggers] settings and your app uses a custom
  config file, a Splunk Enterprise restart will be required after every
state change.
* Specifying "simple" implies that Splunk Enterprise will take no
special action to
  reload your custom conf file.
* Specify "access_endpoints" and a URL to a REST endpoint, and Splunk
Enterprise will
  call its _reload() method at every app state change.
* Specify "http_get" and a URL to a REST endpoint, and Splunk Enterprise
will simulate
  an HTTP GET request against this URL at every app state change.
* Specify "http_post" and a URL to a REST endpoint, and Splunk
Enterprise will simulate
  an HTTP POST request against this URL at every app state change.
* "rest_endpoints" is reserved for Splunk Enterprise internal use for
reloading
```

```
  restmap.conf.

* Examples:
```

[triggers]

```
[triggers]
        * Do not force a restart of Splunk Enterprise for state changes
of MyApp
        * Do not run special code to tell MyApp to reload
myconffile.conf
        * Apps with custom config files will usually pick this option
        reload.myconffile = simple

        * Do not force a restart of Splunk Enterprise for state changes
of MyApp.
        * Splunk Enterprise calls the /admin/myendpoint/_reload method
in my custom EAI handler.
        * Use this advanced option only if MyApp requires custom code to
reload its configuration when its state changes
        reload.myotherconffile = access_endpoints /admin/myendpoint

#
# Set UI-specific settings for this app
#
```

[ui]

```
[ui]
is_visible = true | false
* Indicates if this app should be visible/navigable as a UI app
* Apps require at least 1 view to be available from the UI

show_in_nav = true | false
* Indicates if this app should be shown in glabal app dropdown

is_manageable = true | false
* Support for this setting has been removed. It no longer has any
effect.

label = <string>
* Defines the name of the app shown in the Splunk Enterprise GUI and
Launcher
* Recommended length between 5 and 80 characters.
* Must not include "Splunk For" prefix.
* Label is required.
* Examples of good labels:
    IMAP Monitor
```

```
    SQL Server Integration Services
    FISMA Compliance

docs_section_override = <string>
* Defines override for auto-generated app-specific documentation links
* If not specified, app-specific documentation link will
  include [<app-name>:<app-version>]
* If specified, app-specific documentation link will
  include [<docs_section_override>]
* This only applies to apps with documentation on the Splunk
documentation site

attribution_link = <string>
* URL that users can visit to find third-party software credits and
attributions for assets the app uses.
* External links must start with http:// or https://.
* Values that do not start with http:// or https:// will be interpreted
as Quickdraw "location" strings
* and translated to internal documentation references.

setup_view = <string>
* Optional setting
* Defines custom setup view found within /data/ui/views REST endpoint
* If not specified, default to setup.xml

#
# Credential-verification scripting (4.2+ versions only)
# Credential entries are superseded by passwords.conf from 6.3 onwards.
# While the entries here are still honored post-6.3, updates to these
will occur in passwords.conf which will shadow any values present here.
#
```

[credentials_settings]

```
[credentials_settings]
verify_script = <string>
* Optional setting.
* Command line to invoke to verify credentials used for this app.
* For scripts, the command line should include both the interpreter and
the
  script for it to run.
    * Example: "$SPLUNK_HOME/bin/python"
"$SPLUNK_HOME/etc/apps/<myapp>/bin/$MY_SCRIPT"
* The invoked program is communicated with over standard in / standard
out via
  the same protocol as splunk scripted auth.
* Paths incorporating variable expansion or explicit spaces must be
quoted.
  * For example, a path including $SPLUNK_HOME should be quoted, as
likely
```

```
    will expand to C:\Program Files\Splunk
```

[credential:<realm>:<username>]

```
[credential:<realm>:<username>]
password = <password>
* Password that corresponds to the given username for the given realm.
  Note that realm is optional
* The password can be in clear text, however when saved from splunkd the
  password will always be encrypted

# diag app extensions, 6.4+ only
```

[diag]

```
[diag]
extension_script = <filename>
* Setting this variable declares that this app will put additional
information
  into the troubleshooting & support oriented output of the 'splunk
diag'
  command.
* Must be a python script.
* Must be a simple filename, with no directory separators.
* The script must exist in the 'bin' sub-directory in the app.
* Full discussion of the interface is located on the Developer portal.
  See http://dev.splunk.com/view/SP-CAAAE8H
* Defaults to unset, no app-specific data collection will occur.

data_limit = <positive integer>[b|kb|MB|GB]
* Defines a soft-ceiling for the amount of uncompressed data that should
be
  added to the diag by the app extension.
* Large diags damage the main functionality of the tool by creating
data blobs
  too large to copy around or upload.
* Use this setting to ensure that your extension script does not
accidentally
  produce far too much data.
* Once data produced by this app extension reaches the limit, diag will
not add
  any further files on behalf of the extension.
* After diag has finished adding a file which goes over this limit, all
further files
  will not be added.
* Must be a positive number followed by a size suffix.
  * Valid suffixes: b: bytes, kb: kilobytes, mb: megabytes, gb:
gigabytes
```

```
  * Suffixes are case insensitive.
* Defaults to 100MB.


# Other diag settings

default_gather_lookups = <filename> [, <filename> ...]
* Setting this variable declares that the app contains lookups which
should
  always be gathered by diag (by default).
* Essentially, if there are lookups which are useful for troubleshooting
an
  app, and will never contain sensitive (user) data, they can be added
to this
  list, and they will appear in generated diags for use when
troubleshooting
  the app from customer diags.
* Any files in lookup dirs which are not listed here are not gathered
by
  default; this can be overridden with the diag flag --include-lookups
* This setting is new in Splunk Enterprise/Light version 6.5.  Older
versions
  gather all lookups by default.
* This does not override the size-ceiling on files in etc.  Large
lookups will
  still be excluded, unless the etc-filesize-limit is raised/disabled.
* This controls only files in the same app directory as this conf file.
 For
  example, if you have an app directory in etc/slave-apps (index
clustering),
  this setting must appear in etc/slave-apps/appname/default/app.conf
or
  local/app.conf
* Additional lists can be created with default_gather_lookups-classname
= ...
* Defaults to unset.
```

app.conf.example

```
#   Version 6.5.0
#
# The following are example app.conf configurations. Configure
properties for
# your custom application.
#
# There is NO DEFAULT app.conf.
#
# To use one or more of these configurations, copy the configuration
block into
```

```
# app.conf in $SPLUNK_HOME/etc/system/local/. You must restart Splunk
to
# enable configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

[launcher]
author=<author of app>
description=<textual description of app>
version=<version of app>
```

audit.conf

The following are the spec and example files for audit.conf.

audit.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attributes and values you can use to
configure
# auditing and event signing in audit.conf.
#
# There is NO DEFAULT audit.conf. To set custom configurations, place an
# audit.conf in $SPLUNK_HOME/etc/system/local/. For examples, see
# audit.conf.example.  You must restart Splunk to enable
configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#  * You can also define global settings outside of any stanza, at the
```

```
top of the file.
#  * Each conf file should have at most one default stanza. If there
are
#    multiple default stanzas, attributes are combined. In the case of
multiple
#    definitions of the same attribute, the last definition in the file
wins.
#  * If an attribute is defined at both the global level and in a
specific
#    stanza, the value in the specific stanza takes precedence.

############################################################################################
# KEYS: specify your public and private keys for encryption.
############################################################################################
```

[auditTrail]

```
[auditTrail]
* This stanza turns on cryptographic signing for audit trail events
(set in inputs.conf).
* You must have a private key to encrypt the signatures and a public
key to
  decrypt them.

privateKey= <path>
* The path to the file containing the private key.
* Generate your own keys using openssl in $SPLUNK_HOME/bin/.
* If not present, a default key will be generated one time and placed at
  $SPLUNK_HOME/etc/auth/audit/private.pem

publicKey= <path>
* The path to the file containing the public key.
* Generate your own keys using openssl in $SPLUNK_HOME/bin/.
* If not present, a default key will be generated one time and placed at
  $SPLUNK_HOME/etc/auth/audit/public.pem

queueing=[true|false]
* Turn off sending audit events to the indexQueue -- tail the audit
events
  instead.
* If this is set to 'false', you MUST add an inputs.conf stanza to tail
the
  audit log in order to have the events reach your index.
  * Defaults to true.
```

audit.conf.example

```
#   Version 6.5.0
#
# This is an example audit.conf.  Use this file to configure auditing.
#
# There is NO DEFAULT audit.conf.
#
# To use one or more of these configurations, copy the configuration
block into
# audit.conf in $SPLUNK_HOME/etc/system/local/.  You must restart
Splunk to
# enable configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

[auditTrail]
privateKey=/some/path/to/your/private/key/private_key.pem
publicKey=/some/path/to/your/public/key/public_key.pem

# If this stanza exists, audit trail events will be cryptographically
signed.
# You must have a private key to encrypt the signatures and a public
key to decrypt them.
# Generate your own keys using openssl in $SPLUNK_HOME/bin/.
```

authentication.conf

The following are the spec and example files for authentication.conf.

authentication.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attributes and values for configuring
# authentication via authentication.conf.
#
# There is an authentication.conf in $SPLUNK_HOME/etc/system/default/.
 To
# set custom configurations, place an authentication.conf in
# $SPLUNK_HOME/etc/system/local/. For examples, see
```

```
# authentication.conf.example.  You must restart Splunk to enable
# configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
top
#     of the file.
#   * Each conf file should have at most one default stanza. If there
are
#     multiple default stanzas, attributes are combined. In the case of
#     multiple definitions of the same attribute, the last definition in
the
#     file wins.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.

[authentication]
* Follow this stanza name with any number of the following
attribute/value
  pairs.

authType = [Splunk|LDAP|Scripted|SAML|ProxySSO]
* Specify which authentication system to use.
* Supported values: Splunk, LDAP, Scripted, SAML, ProxySSO.
* Defaults to Splunk.

authSettings = <authSettings-key>,<authSettings-key>,...
* Key to look up the specific configurations of chosen authentication
  system.
* <authSettings-key> is the name of a stanza header that specifies
  attributes for scripted authentication, SAML, ProxySSO and for an
LDAP
  strategy. Those stanzas are defined below.
* For LDAP, specify the LDAP strategy name(s) here. If you want Splunk
to
  query multiple LDAP servers, enter a comma-separated list of all
  strategies. Each strategy must be defined in its own stanza. The order
in
  which you specify the strategy names will be the order Splunk uses to
```

```
  query their servers when looking for a user.
* For scripted authentication, <authSettings-key> should be a single
  stanza name.

passwordHashAlgorithm =
[SHA512-crypt|SHA256-crypt|SHA512-crypt-<num_rounds>|SHA256-crypt-<num_rounds>|MD5-cryp
* For the default "Splunk" authType, this controls how hashed passwords
are
  stored in the $SPLUNK_HOME/etc/passwd file.
* "MD5-crypt" is an algorithm originally developed for FreeBSD in the
early
  1990's which became a widely used standard among UNIX machines.  It
was
  also used by Splunk up through the 5.0.x releases.  MD5-crypt runs the
  salted password through a sequence of 1000 MD5 operations.
* "SHA256-crypt" and "SHA512-crypt" are newer versions that use 5000
rounds
  of the SHA256 or SHA512 hash functions.  This is slower than
MD5-crypt and
  therefore more resistant to dictionary attacks.  SHA512-crypt is used
for
  system passwords on many versions of Linux.
* These SHA-based algorithm can optionally be followed by a number of
rounds
  to use.  For example, "SHA512-crypt-10000" will use twice as many
rounds
  of hashing as the default implementation.  The number of rounds must
be at
  least 1000.
  If you specify a very large number of rounds (i.e. more than 20x the
  default value of 5000), splunkd may become unresponsive and
connections to
  splunkd (from splunkweb or CLI) will time out.
* This setting only affects new password settings (either when a user is
  added or a user's password is changed)  Existing passwords will
continue
  to work but retain their previous hashing algorithm.
* The default is "SHA512-crypt".

externalTwoFactorAuthVendor = <string>
* OPTIONAL.
* A valid Multifactor vendor string will enable Multifactor
authentication
  and loads support for the corresponding vendor if supported by
Splunk.
* Empty string will disable Multifactor authentication in splunk.
* Currently splunk supports duo as a Multifactor authentication vendor.

externalTwoFactorAuthSettings = <externalTwoFactorAuthSettings-key>
* OPTIONAL.
* Key to look up the specific configuration of chosen Multifactor
  authentication vendor.
```

LDAP settings

```
####################
# LDAP settings
####################LDAP settings

[<authSettings-key>]
* Follow this stanza name with the attribute/value pairs listed below.
* For multiple strategies, you will need to specify multiple instances
of
  this stanza, each with its own stanza name and a separate set of
  attributes.
* The <authSettings-key> must be one of the values listed in the
  authSettings attribute, specified above in the [authentication]
stanza.

host = <string>
* REQUIRED
* This is the hostname of LDAP server.
* Be sure that your Splunk server can resolve the host name.

SSLEnabled = [0|1]
* OPTIONAL
* Defaults to disabled (0)
* See the file $SPLUNK_HOME/etc/openldap/openldap.conf for SSL LDAP
settings

port = <integer>
* OPTIONAL
* This is the port that Splunk should use to connect to your LDAP
server.
* Defaults to port 389 for non-SSL and port 636 for SSL

bindDN = <string>
* OPTIONAL, leave this blank to retrieve your LDAP entries using
  anonymous bind (must be supported by the LDAP server)
* Distinguished name of the user that will be retrieving the LDAP
entries
* This user must have read access to all LDAP users and groups you wish
to
  use in Splunk.

bindDNpassword = <password>
* OPTIONAL, leave this blank if anonymous bind is sufficient
* Password for the bindDN user.

userBaseDN = <string>
* REQUIRED
```

```
* This is the distinguished names of LDAP entries whose subtrees
contain the users
* Enter a ';' delimited list to search multiple trees.

userBaseFilter = <string>
* OPTIONAL
* This is the LDAP search filter you wish to use when searching for
users.
* Highly recommended, especially when there are many entries in your
LDAP
  user subtrees
* When used properly, search filters can significantly speed up LDAP
queries
* Example that matches users in the IT or HR department:
    * userBaseFilter = (|(department=IT)(department=HR))
    * See RFC 2254 for more detailed information on search filter syntax
* This defaults to no filtering.

userNameAttribute = <string>
* REQUIRED
* This is the user entry attribute whose value is the username.
* NOTE: This attribute should use case insensitive matching for its
values,
  and the values should not contain whitespace
    * Usernames are case insensitive in Splunk
* In Active Directory, this is 'sAMAccountName'
* A typical attribute for this is 'uid'

realNameAttribute = <string>
* REQUIRED
* This is the user entry attribute whose value is their real name
  (human readable).
* A typical attribute for this is 'cn'

emailAttribute = <string>
* OPTIONAL
* This is the user entry attribute whose value is their email address.
* Defaults to 'mail'

groupMappingAttribute  = <string>
* OPTIONAL
* This is the user entry attribute whose value is used by group entries
to
  declare membership.
* Groups are often mapped with user DN, so this defaults to 'dn'
* Set this if groups are mapped using a different attribute
  * Usually only needed for OpenLDAP servers.
  * A typical attribute used to map users to groups is 'uid'
    * For example, assume a group declares that one of its members is
      'splunkuser'
    * This implies that every user with 'uid' value 'splunkuser' will
be
```

```
      mapped to that group

groupBaseDN = [<string>;<string>;...]
* REQUIRED
* This is the distinguished names of LDAP entries whose subtrees
contain
  the groups.
* Enter a ';' delimited list to search multiple trees.
* If your LDAP environment does not have group entries, there is a
  configuration that can treat each user as its own group
  * Set groupBaseDN to the same as userBaseDN, which means you will
search
    for groups in the same place as users
  * Next, set the groupMemberAttribute and groupMappingAttribute to the
same
    attribute as userNameAttribute
    * This means the entry, when treated as a group, will use the
username
      value as its only member
  * For clarity, you should probably also set groupNameAttribute to the
same
    value as userNameAttribute as well

groupBaseFilter = <string>
* OPTIONAL
* The LDAP search filter Splunk uses when searching for static groups
* Like userBaseFilter, this is highly recommended to speed up LDAP
queries
* See RFC 2254 for more information
* This defaults to no filtering

dynamicGroupFilter = <string>
* OPTIONAL
* The LDAP search filter Splunk uses when searching for dynamic groups
* Only configure this if you intend to retrieve dynamic groups on your
LDAP server
* Example: '(objectclass=groupOfURLs)'

dynamicMemberAttribute = <string>
* OPTIONAL
* Only configure this if you intend to retrieve dynamic groups on your
  LDAP server
* This is REQUIRED if you want to retrieve dynamic groups
* This attribute contains the LDAP URL needed to retrieve members
dynamically
* Example: 'memberURL'

groupNameAttribute = <string>
* REQUIRED
* This is the group entry attribute whose value stores the group name.
* A typical attribute for this is 'cn' (common name)
* Recall that if you are configuring LDAP to treat user entries as
```

```
their own
  group, user entries must have this attribute

groupMemberAttribute = <string>
* REQUIRED
* This is the group entry attribute whose values are the groups members
* Typical attributes for this are 'member' and 'memberUid'
* For example, consider the groupMappingAttribute example above using
  groupMemberAttribute 'member'
  * To declare 'splunkuser' as a group member, its attribute 'member'
must
    have the value 'splunkuser'

nestedGroups = <bool>
* OPTIONAL
* Controls whether Splunk will expand nested groups using the
  'memberof' extension.
* Set to 1 if you have nested groups you want to expand and the
'memberof'
* extension on your LDAP server.

charset = <string>
* OPTIONAL
* ONLY set this for an LDAP setup that returns non-UTF-8 encoded data.
LDAP
  is supposed to always return UTF-8 encoded data (See RFC 2251), but
some
  tools incorrectly return other encodings.
* Follows the same format as CHARSET in props.conf (see props.conf.spec)
* An example value would be "latin-1"

anonymous_referrals = <bool>
* OPTIONAL
* Set this to 0 to turn off referral chasing
* Set this to 1 to turn on anonymous referral chasing
* IMPORTANT: We only chase referrals using anonymous bind. We do NOT
support
  rebinding using credentials.
* If you do not need referral support, we recommend setting this to 0
* If you wish to make referrals work, set this to 1 and ensure your
server
  allows anonymous searching
* Defaults to 1

sizelimit = <integer>
* OPTIONAL
* Limits the amount of entries we request in LDAP search
* IMPORTANT: The max entries returned is still subject to the maximum
  imposed by your LDAP server
  * Example: If you set this to 5000 and the server limits it to 1000,
             you'll still only get 1000 entries back
* Defaults to 1000
```

```
timelimit = <integer>
* OPTIONAL
* Limits the amount of time in seconds we will wait for an LDAP search
  request to complete
* If your searches finish quickly, you should lower this value from the
  default
* Defaults to 15

network_timeout = <integer>
* OPTIONAL
* Limits the amount of time a socket will poll a connection without
activity
* This is useful for determining if your LDAP server cannot be reached
* IMPORTANT: As a connection could be waiting for search results, this
value
             must be higher than 'timelimit'
* Like 'timelimit', if you have a fast connection to your LDAP server,
we
  recommend lowering this value
* Defaults to 20
```

Map roles

```
#####################
# Map roles
#####################Map roles

[roleMap_<authSettings-key>]
* The mapping of Splunk roles to LDAP groups for the LDAP strategy
specified
  by <authSettings-key>
* IMPORTANT: this role mapping ONLY applies to the specified strategy.
* Follow this stanza name with several Role-to-Group(s) mappings as
defined
  below.
* Note: Importing groups for the same user from different strategies is
not
  supported.

<Splunk RoleName> = <LDAP group string>
* Maps a Splunk role (from authorize.conf) to LDAP groups
* This LDAP group list is semicolon delimited (no spaces).
* List several of these attribute value pairs to map several Splunk
roles to
  LDAP Groups
```

Scripted authentication

```
#####################
# Scripted authentication
#####################Scripted authentication

[<authSettings-key>]
* Follow this stanza name with the following attribute/value pairs:

scriptPath = <string>
* REQUIRED
* This is the full path to the script, including the path to the program
  that runs it (python)
* For example: "$SPLUNK_HOME/bin/python"
"$SPLUNK_HOME/etc/system/bin/$MY_SCRIPT"
* Note: If a path contains spaces, it must be quoted. The example above
  handles the case where SPLUNK_HOME contains a space

scriptSearchFilters = [1|0]
* OPTIONAL - Only set this to 1 to call the script to add search
filters.
* 0 disables (default)

[cacheTiming]
* Use these settings to adjust how long Splunk will use the answers
returned
  from script functions before calling them again.

userLoginTTL = <time range string>
* Timeout for the userLogin script function.
* These return values are cached on a per-user basis.
* The default is '0' (no caching)

getUserInfoTTL = <time range string>
* Timeout for the getUserInfo script function.
* These return values are cached on a per-user basis.
* The default is '10s'

getUsersTTL = <time range string>
* Timeout for the getUsers script function.
* There is only one global getUsers cache (it is not tied to a
  specific user).
* The default is '10s'

* All timeouts can be expressed in seconds or as a search-like time
range
* Examples include '30' (30 seconds), '2mins' (2 minutes), '24h' (24
hours), etc.
* You can opt to use no caching for a particular function by setting the
  value to '0'
```

```
  * Be aware that this can severely hinder performance as a result of
heavy
    script invocation
* Choosing the correct values for cache timing involves a tradeoff
between
  new information latency and general performance
  * High values yield better performance from calling the script less,
but
    introduces a latency in picking up changes
  * Low values will pick up changes in your external auth system more
    quickly, but may slow down performance due to increased script
    invocations
```

Settings for Splunk Authentication mode

```
#####################
# Settings for Splunk Authentication mode
#####################Settings for Splunk Authentication mode

[splunk_auth]
* Settings for Splunk's internal authentication system.

minPasswordLength = <positive integer>
* Specifies the minimum permitted password length in characters when
  passwords are set or modified.
* This setting is optional.
* If 0, there is no required minimum.  In other words there is no
constraint.
* Password modification attempts which do not meet this requirement will
be
* explicitly rejected.  Defaults to 0 (disabled).
```

SAML settings

```
#####################
# SAML settings
#####################SAML settings

[<saml-authSettings-key>]
* Follow this stanza name with the attribute/value pairs listed below.
* The <authSettings-key> must be one of the values listed in the
* authSettings attribute, specified above in the [authentication]
stanza.

fqdn = <string>
```

```
* OPTIONAL
* The fully qualified domain name where this splunk instance is
running.
* If this value is not specified, Splunk will default to the value
specified
  in server.conf.
* If this value is specified and 'http://' or 'https://' prefix is not
  present, splunk will use the ssl setting for splunkweb.
* Splunk will use this information to populate the
'assertionConsumerServiceUrl'.

redirectPort = <port number>
* OPTIONAL
* The port where SAML responses will be sent. Typically, this is the
  web port.
* If internal port redirection is needed, set this port and the
  'assertionconsumerServiceUrl' in the AuthNRequest will contain this
port
  instead of the splunkweb port.
* To prevent any port information to be appended in the
  'assertionConsumerServiceUrl' attribute, set this to 0.

idpSSOUrl = <url>
* REQUIRED
* The protocol endpoint on the IDP (Identity Provider) where the
  AuthNRequests should be sent.
* SAML requests will fail if this information is missing.

idpAttributeQueryUrl = <url>
* OPTIONAL
* The protocol endpoint on the IDP (Identity Provider) where the
attribute
  query requests should be sent.
* Attribute queries can be used to get the latest 'role' information,
  if there is support for Attribute queries on the IDP.
* When this setting is absent, Splunk will cache the role information
from the saml
  assertion and use it to run saved searches.

idpCertPath = <Pathname>
* OPTIONAL
* This setting is required if 'signedAssertion' is set to true.
* This value is relative to $SPLUNK_HOME/etc/auth/idpCerts.
* The value for this setting can be the name of the certificate file or
a directory.
* If it is empty, Splunk will automatically verify with certificates in
all subdirectories
  present in $SPLUNK_HOME/etc/auth/idpCerts.
* If the saml response is to be verified with a IDP (Identity Provider)
certificate that
  is self signed, then this setting holds the filename of the
certificate.
```

```
* If the saml response is to be verified with a certificate that is a
part of a
  certificate chain(root, intermediate(s), leaf), create a subdirectory
and place the
  certificate chain as files in the subdirectory.
* If there are multiple end certificates, create a subdirectory such
that, one subdirectory
  holds one certificate chain.
* If multiple such certificate chains are present, the assertion is
considered verified,
  if validation succeeds with any certifcate chain.
* The file names within a certificate chain should be such that root
certificate is alphabetically
  before the intermediate which is alphabetically before of the end
cert.
  ex. cert_1.pem has the root, cert_2.pem has the first intermediate
cert, cert_3.pem has the second
      intermediate certificate and cert_4.pem has the end certificate.

idpSLOUrl = = <url>
* OPTIONAL
* The protocol endpoint on the IDP (Identity Provider) where a SP
  (Service Provider) initiated Single logout request should be sent.

errorUrl = <url>
* OPTIONAL
* The url to be displayed for a SAML error. Errors may be due to
  erroneous or incomplete configuration in either the IDP or Splunk.
  This url can be absolute or relative. Absolute url should follow
pattern
  <protocol>:[//]<host> e.g. https://www.external-site.com.
  Relative urls should start with '/'. A relative url will show up as an
  internal link of the splunk instance, e.g.
https://splunkhost:port/relativeUrlWithSlash

errorUrlLabel = <string>
* OPTIONAL
* Label or title of the content pointed to by errorUrl.

entityId = <string>
* REQUIRED
* The entity id for SP connection as configured on the IDP.

signAuthnRequest = [ true | false ]
* OPTIONAL
* This tells Splunk whether to sign AuthNRequests.
* Defaults to true.

signedAssertion = [true|false]
* OPTIONAL
* This tells Splunk if the SAML assertion has been signed by the IDP
* If set to false, Splunk will not verify the signature of the
```

```
assertion
  using the certificate of the IDP.
* Currently, we accept only signed assertions.
* Defaults to true.

attributeQuerySoapPassword = <password>
* OPTIONAL
* This setting is required if 'attributeQueryUrl' is specified.
* Attribute query requests are made using SOAP using basic
authentication
* The password to be used when making an attribute query request.
* This string will obfuscated upon splunkd startup.

attributeQuerySoapUsername = <string>
* OPTIONAL
* This setting is required if 'attributeQueryUrl' is specified.
* Attribute Query requests are made using SOAP using basic
authentication
* The username to be used when making an attribute query request.

attributeQueryRequestSigned = [ true | false ]
* OPTIONAL
* Specifies whether to sign attribute query requests.
* Defaults to true

attributeQueryResponseSigned = [ true | false ]
* OPTIONAL
* Specifies whether attribute query responses are signed.
* If set to false, Splunk will not verify the signature in the response
  using the certificate of the IDP.
* Defaults to true.

redirectAfterLogoutToUrl = <url>
* OPTIONAL
* The user will be redirected to this url after logging out of Splunk.
* If this is not specified and a idpSLO is also missing, the user will
be
  redirected to splunk.com after logout.

defaultRoleIfMissing = <splunk role>
* OPTIONAL
* If the IDP does not return any AD groups or splunk roles as a part of
the
  assertion, we will use this value if provided.

skipAttributeQueryRequestForUsers = <comma separated list of users>
* OPTIONAL
* To skip attribute query requests being sent to the IDP for certain
users,
  add them here.
* By default, attribute query requests will be skipped for local users.
* For non-local users, use this in conjunction with
```

```
'defaultRoleIfMissing'.

maxAttributeQueryThreads = <int>
* OPTIONAL
* Defaults to 2, max is 10
* Number of threads to use to make attribute query requests.
* Changes to this will require a restart to take effect.

maxAttributeQueryQueueSize = <int>
* OPTIONAL
* Defaults to 50
* The number of attribute query requests to queue, set to 0 for infinite
  size.
* Changes to this will require a restart to take effect.

attributeQueryTTL = <ttl in seconds>
* OPTIONAL
* Determines the time for which Splunk will cache the user and role
  information.
* Once the ttl expires, Splunk will make an attribute query request to
  retrieve the role information.
* Default ttl if not specified, is 3600 seconds.

allowSslCompression = [ true | false ]
* OPTIONAL
* If set to true, the server will allow clients to negotiate SSL-layer
  data compression.
* If not set, defaults to the setting in server.conf.

cipherSuite = <cipher suite string>
* OPTIONAL
* If set, Splunk uses the specified cipher string for the HTTP server.
* If not set, defaults to the setting in server.conf.
* Attribute query requests might fail if the IDP requires a relaxed
  ciphersuite.
* Use "openssl s_client -cipher 'TLSv1+HIGH:@STRENGTH' -host <IDP host>
-port 443"
  to determine if splunk can connect to the IDP

sslVersions = <versions_list>
* OPTIONAL
* Comma-separated list of SSL versions to support.
* The versions available are "ssl3", "tls1.0", "tls1.1", and "tls1.2"
* If not set, defaults to the setting in server.conf.

sslCommonNameToCheck = <commonName>
* OPTIONAL
* If this value is set, and 'sslVerifyServerCert' is set to true,
  splunkd will limit most outbound HTTPS connections to hosts which use
  a cert with this common name.
* If not set, Splunk uses the setting specified in server.conf.
```

```
sslAltNameToCheck = <alternateName1>, <alternateName2>, ...
* OPTIONAL
* If this value is set, and 'sslVerifyServerCert' is set to true,
  splunkd will also be willing to verify certificates which have a
so-called
  "Subject Alternate Name" that matches any of the alternate names in
this
  list.
* If not set, Splunk uses the setting specified in server.conf.

ecdhCurveName = <string>
* DEPRECATED; use 'ecdhCurves' instead.
* ECDH curve to use for ECDH key negotiation.
* If not set, Splunk uses the setting specified in server.conf.

ecdhCurves = <comma separated list of ec curves>
* ECDH curves to use for ECDH key negotiation.
* The curves should be specified in the order of preference.
* The client sends these curves as a part of Client Hello.
* The server supports only the curves specified in the list.
* We only support named curves specified by their SHORT names.
  (see struct ASN1_OBJECT in asn1.h)
* The list of valid named curves by their short/long names can be
obtained
  by executing this command:
  $SPLUNK_HOME/bin/splunk cmd openssl ecparam -list_curves
* Default is empty string.
* e.g. ecdhCurves = prime256v1,secp384r1,secp521r1
* If not set, Splunk uses the setting specified in server.conf.

clientCert = <path>
* Full path to the client certificate PEM format file.
* Certificates are auto-generated upon first starting Splunk.
* You may replace the auto-generated certificate with your own.
* Default is $SPLUNK_HOME/etc/auth/server.pem.
* If not set, Splunk uses the setting specified in
  server.conf/[sslConfig]/serverCert.

sslKeysfile = <filename>
* DEPRECATED; use 'clientCert' instead.
* File is in the directory specified by 'caPath' (see below).
* Default is server.pem.

sslPassword = <password>
* Optional server certificate password.
* If unset, Splunk uses the setting specified in server.conf.
* Default is password.

sslKeysfilePassword = <password>
* DEPRECATED; use 'sslPassword' instead.

caCertFile = <filename>
```

```
* OPTIONAL
* Public key of the signing authority.
* Default is cacert.pem.
* If not set, Splunk uses the setting specified in server.conf.

caPath = <path>
* DEPRECATED; use absolute paths for all certificate files.
* If certificate files given by other settings in this stanza are not
absolute
  paths, then they will be relative to this path.
* Default is $SPLUNK_HOME/etc/auth.

sslVerifyServerCert = <bool>
* OPTIONAL
* Used by distributed search: when making a search request to another
  server in the search cluster.
* If not set, Splunk uses the setting specified in server.conf.

blacklistedAutoMappedRoles = <comma separated list of roles>
* OPTIONAL
* Comma separated list of splunk roles that should be blacklisted
  from being auto-mapped by splunk from the IDP Response.

blacklistedUsers = <comma separated list of user names>
* OPTIONAL
* Comma separated list of user names from the IDP response to be
  blacklisted by splunk platform.

nameIdFormat = <string>
* OPTIONAL
* If supported by IDP, while making SAML Authentication request this
value can
  be used to specify the format of the Subject returned in SAML
Assertion.

ssoBinding = <string>
* OPTIONAL
* This is the binding that will be used when making a SP-initiated saml
request.
* Acceptable options are 'HTTPPost' and 'HTTPRedirect'
* Defaults to 'HTTPPost'
* This binding must match the one configured on the IDP.

sloBinding = <string>
* OPTIONAL
* This is the binding that will be used when making a logout request or
sending a logout
* response to complete the logout workflow.
* Acceptable options are 'HTTPPost' and 'HTTPRedirect'
* Defaults to 'HTTPPost'
* This binding must match the one configured on the IDP.
```

```
signatureAlgorithm = RSA-SHA1 | RSA-SHA256
* OPTIONAL
* Defaults to RSA-SHA1.
* This setting is applicable only for redirect binding.
* RSA-SHA1 corresponds to 'http://www.w3.org/2000/09/xmldsig#rsa-sha1'.
* RSA-SHA256 corresponds to
'http://www.w3.org/2001/04/xmldsig-more#rsa-sha256'.
* Specifies the signature algorithm that will be used for a SP-initiated
saml request,
* when 'signedAuthnRequest' is set to true.
* This will be sent as a part of 'sigAlg'.
```

Map roles

```
#####################
# Map roles
#####################Map roles

[roleMap_<saml-authSettings-key>]
* The mapping of Splunk roles to SAML groups for the SAML stanza
specified
  by <authSettings-key>
* If a SAML group is not explicitly mapped to a Splunk role, but has
  same name as a valid Splunk role then for ease of configuration, it
is
  auto-mapped to that Splunk role.
* Follow this stanza name with several Role-to-Group(s) mappings as
defined
  below.

<Splunk RoleName> = <SAML group string>
* Maps a Splunk role (from authorize.conf) to SAML groups
* This SAML group list is semicolon delimited (no spaces).
* List several of these attribute value pairs to map several Splunk
roles to
  SAML Groups.
* If role mapping is not specified, Splunk expects Splunk roles in the
  assertion and attribute query response returned from the IDP.
```

SAML User Roles Map

```
#####################
# SAML User Roles Map
#####################SAML User Roles Map
```

```
[userToRoleMap_<saml-authSettings-key>]
* The mapping of SAML user to Splunk roles for the SAML stanza specified
  by <authSettings-key>
* Follow this stanza name with several User-to-Role(s) mappings as
defined
  below.
* The stanza is used only when the IDP does not support Attribute Query
Request

<SAML User> = <Splunk Roles string>
* Maps a SAML user to Splunk role (from authorize.conf)
* This Splunk Role list is semicolon delimited (no spaces).
```

Authentication Response Attribute Map

```
#####################
# Authentication Response Attribute Map
####################Authentication Response Attribute Map

[authenticationResponseAttrMap_SAML]
* Splunk expects email, real name and roles to be returned as SAML
  Attributes in SAML assertion. This stanza can be used to map attribute
names
  to what Splunk expects. These are optional settings and are only
needed for
  certain IDPs.

role = <string>
* OPTIONAL
* Attribute name to be used as role in SAML Assertion.
* Default is "role"

realName = <string>
* OPTIONAL
* Attribute name to be used as realName in SAML Assertion.
* Default is "realName"

mail = <string>
* OPTIONAL
* Attribute name to be used as email in SAML Assertion.
* Default is "mail"
```

Settings for Proxy SSO mode

```
#####################
# Settings for Proxy SSO mode
```

```
####################Settings for Proxy SSO mode

[roleMap_proxySSO]

* The mapping of Splunk roles to groups passed in headers from proxy
server.
* If a group is not explicitly mapped to a Splunk role, but has
  same name as a valid Splunk role then for ease of configuration, it
is
  auto-mapped to that Splunk role.
* Follow this stanza name with several Role-to-Group(s) mappings as
defined
  below.

<Splunk RoleName> = <Group string>
* Maps a Splunk role (from authorize.conf) to groups
* This group list is semicolon delimited (no spaces).
* List several of these attribute value pairs to map several Splunk
roles to
  Groups
* If role mapping is not specified, user is logged in with default User
role.

[userToRoleMap_proxySSO]
* The mapping of ProxySSO user to Splunk roles
* Follow this stanza name with several User-to-Role(s) mappings as
defined
  below.

<ProxySSO User> = <Splunk Roles string>
* Maps a ProxySSO user to Splunk role (from authorize.conf)
* This Splunk Role list is semicolon delimited (no spaces).

[proxysso-authsettings-key]
* Follow this stanza name with the attribute/value pairs listed below.

defaultRoleIfMissing = <splunk role>
* OPTIONAL
* If splunk roles cannot be determined based on role mapping, use
default configured
* splunk role.

blacklistedAutoMappedRoles = <comma separated list of roles>
* OPTIONAL
* Comma separated list of splunk roles that should be blacklisted
  from being auto-mapped by splunk from the proxy server headers.

blacklistedUsers = <comma separated list of user names>
* OPTIONAL
* Comma separated list of user names from the proxy server headers to be
  blacklisted by splunk platform.
```

Secret Storage

```
#####################
# Secret Storage
#####################Secret Storage

[secrets]

disabled = <bool>
* Toggles integration with platform-provided secret storage facilities.
* Defaults to false if Common Criteria mode is enabled.
* Defaults to true if Common Criteria mode is disabled.
* NOTE: Splunk plans to submit Splunk Enterprise for Common Criteria
  evaluation. Splunk does not support using the product in Common
  Criteria mode until it has been certified by NIAP. See the "Securing
  Splunk Enterprise" manual for information on the status of Common
  Criteria certification.

filename = <filename>
* Designates a Python script that integrates with platform-provided
  secret storage facilities, like the GNOME keyring.
* <filename> should be the name of a Python script located in one of the
  following directories:
    $SPLUNK_HOME/etc/apps/*/bin
    $SPLUNK_HOME/etc/system/bin
    $SPLUNK_HOME/etc/searchscripts
* <filename> should be a pure basename; it should contain no path
separators.
* <filename> should end with a .py file extension.

namespace = <string>
* Use an instance-specific string as a namespace within secret storage.
* When using the GNOME keyring, this namespace is used as a keyring
name.
* If multiple Splunk instances must store separate sets of secrets
within the
  same storage backend, this value should be customized to be unique for
each
  Splunk instance.
* Defaults to "splunk".
```

Duo MFA vendor settings

```
#####################
# Duo MFA vendor settings
#####################Duo MFA vendor settings
[<duo-externalTwoFactorAuthSettings-key>]
* <duo-externalTwoFactorAuthSettings-key> must be the value listed in
```

```
the
  externalTwoFactorAuthSettings attribute, specified above in the
[authentication]
  stanza.
* This stanza contains Duo specific Multifactor authentication settings
and will be
  activated only when externalTwoFactorAuthVendor is Duo.
* All the below attributes except appSecretKey would be provided by Duo.

apiHostname = <string>
* REQUIRED
* Duo's API endpoint which performs the actual Multifactor
authentication.
* e.g. apiHostname = api-xyz.duosecurity.com

integrationKey = <string>
* REQUIRED
* Duo's integration key for splunk. Must be of size = 20.
* Integration key will be obfuscated before being saved here for
security.

secretKey = <string>
* REQUIRED
* Duo's secret key for splunk. Must be of size = 40.
* Secret key will be obfuscated before being saved here for security.

appSecretKey = <string>
* REQUIRED
* Splunk application specific secret key which should be random and
locally generated.
* Must be atleast of size = 40 or longer.
* This secret key would not be shared with Duo.
* Application secret key will be obfuscated before being saved here for
security.

failOpen = <bool>
* OPTIONAL
* Defaults to false if not set.
* If set to true, Splunk will bypass Duo Multifactor Authentication when
the service is
  unavailable.

timeout = <int>
* OPTIONAL
* It determines the connection timeout in seconds for the outbound duo
HTTPS connection.
* If not set, Splunk will use its default HTTPS connection timeout
which is 12 seconds.

sslVersions = <versions_list>
* OPTIONAL
* Comma-separated list of SSL versions to support for incoming
```

```
connections.
* The versions available are "ssl3", "tls1.0", "tls1.1", and "tls1.2".
* If not set, Splunk uses the sslVersions provided in server.conf

cipherSuite = <cipher suite string>
* OPTIONAL
* If set, Splunk uses the specified cipher string for the HTTP server.
* If not set, Splunk uses the cipher string provided in server.conf

ecdhCurves = <comma separated list of ec curves>
* OPTIONAL
* ECDH curves to use for ECDH key negotiation.
* If not set, Splunk uses the ecdh curve names provided in server.conf

sslVerifyServerCert = <bool>
* OPTIONAL
* Defaults to false if not set.
* If this is set to true, you should make sure that the server that is
  being connected to is a valid one (authenticated). Both the common
  name and the alternate name of the server are then checked for a
  match if they are specified in this configuration file.  A
  certificiate is considered verified if either is matched.

sslCommonNameToCheck = <commonName1>, <commonName2>, ...
* OPTIONAL
* Not set by default.
* If this value is set, Splunk will limit outbound duo HTTPS connections
  to host which use a cert with one of the listed common names.
* sslVerifyServerCert must be set to true for this setting to work.

sslAltNameToCheck =  <alternateName1>, <alternateName2>, ...
* OPTIONAL
* Not set by default.
* If this value is set, Splunk will limit outbound duo HTTPS connections
  to host which use a cert with one of the listed alternate names.
* sslVerifyServerCert must be set to true for this setting to work.

sslRootCAPath = <path>
* OPTIONAL
* Not set by default.
* The <path> must refer to full path of a PEM format file containing
one or more
  root CA certificates concatenated together.
* This Root CA must match the CA in the certificate chain of the SSL
certificate
  returned by duo server.

useClientSSLCompression = <bool>
* OPTIONAL
* If set to true on client side, compression is enabled between the
server and client
  as long as the server also supports it.
```

```
* If not set, Splunk uses the client SSL compression setting provided in
server.conf
```

authentication.conf.example

```
#   Version 6.5.0
#
# This is an example authentication.conf. authentication.conf is used
to
# configure LDAP, Scripted, SAML and Proxy SSO authentication in
addition
# to Splunk's native authentication.
#
# To use one of these configurations, copy the configuration block into
# authentication.conf in $SPLUNK_HOME/etc/system/local/.  You must
reload
# auth in manager or restart Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

##### Use just Splunk's built-in authentication (default):
[authentication]
authType = Splunk


##### LDAP examples

#### Basic LDAP configuration example
[authentication]
authType = LDAP
authSettings = ldaphost

[ldaphost]
host = ldaphost.domain.com
port = 389
SSLEnabled = 0
bindDN = cn=Directory Manager
bindDNpassword = password
userBaseDN = ou=People,dc=splunk,dc=com
userBaseFilter = (objectclass=splunkusers)
groupBaseDN = ou=Groups,dc=splunk,dc=com
groupBaseFilter = (objectclass=splunkgroups)
userNameAttribute = uid
realNameAttribute = givenName
```

```
groupMappingAttribute = dn
groupMemberAttribute = uniqueMember
groupNameAttribute = cn
timelimit = 10
network_timeout = 15

# This stanza maps roles you have created in authorize.conf to LDAP
Groups
[roleMap_ldaphost]
admin = SplunkAdmins

#### Example using the same server as 'ldaphost', but treating each user
as
#### their own group
[authentication]
authType = LDAP
authSettings = ldaphost_usergroups

[ldaphost_usergroups]
host = ldaphost.domain.com
port = 389
SSLEnabled = 0
bindDN = cn=Directory Manager
bindDNpassword = password
userBaseDN = ou=People,dc=splunk,dc=com
userBaseFilter = (objectclass=splunkusers)
groupBaseDN = ou=People,dc=splunk,dc=com
groupBaseFilter = (objectclass=splunkusers)
userNameAttribute = uid
realNameAttribute = givenName
groupMappingAttribute = uid
groupMemberAttribute = uid
groupNameAttribute = uid
timelimit = 10
network_timeout = 15

[roleMap_ldaphost_usergroups]
admin = admin_user1;admin_user2;admin_user3;admin_user4
power = power_user1;power_user2
user = user1;user2;user3

#### Sample Configuration for Active Directory (AD)
[authentication]
authSettings = AD
authType = LDAP

[AD]
SSLEnabled = 1
bindDN = ldap_bind@splunksupport.kom
bindDNpassword = ldap_bind_user_password
groupBaseDN = CN=Groups,DC=splunksupport,DC=kom
groupBaseFilter =
```

```
groupMappingAttribute = dn
groupMemberAttribute = member
groupNameAttribute = cn
host = ADbogus.splunksupport.kom
port = 636
realNameAttribute = cn
userBaseDN = CN=Users,DC=splunksupport,DC=kom
userBaseFilter =
userNameAttribute = sAMAccountName
timelimit = 15
network_timeout = 20
anonymous_referrals = 0

[roleMap_AD]
admin = SplunkAdmins
power = SplunkPowerUsers
user = SplunkUsers

#### Sample Configuration for Sun LDAP Server
[authentication]
authSettings = SunLDAP
authType = LDAP

[SunLDAP]
SSLEnabled = 0
bindDN = cn=Directory Manager
bindDNpassword = Directory_Manager_Password
groupBaseDN = ou=Groups,dc=splunksupport,dc=com
groupBaseFilter =
groupMappingAttribute = dn
groupMemberAttribute = uniqueMember
groupNameAttribute = cn
host = ldapbogus.splunksupport.com
port = 389
realNameAttribute = givenName
userBaseDN = ou=People,dc=splunksupport,dc=com
userBaseFilter =
userNameAttribute = uid
timelimit = 5
network_timeout = 8

[roleMap_SunLDAP]
admin = SplunkAdmins
power = SplunkPowerUsers
user = SplunkUsers

#### Sample Configuration for OpenLDAP
[authentication]
authSettings = OpenLDAP
authType = LDAP

[OpenLDAP]
```

```
bindDN = uid=directory_bind,cn=users,dc=osx,dc=company,dc=com
bindDNpassword = directory_bind_account_password
groupBaseFilter =
groupNameAttribute = cn
SSLEnabled = 0
port = 389
userBaseDN = cn=users,dc=osx,dc=company,dc=com
host = hostname_OR_IP
userBaseFilter =
userNameAttribute = uid
groupMappingAttribute = uid
groupBaseDN = dc=osx,dc=company,dc=com
groupMemberAttribute = memberUid
realNameAttribute = cn
timelimit = 5
network_timeout = 8
dynamicGroupFilter = (objectclass=groupOfURLs)
dynamicMemberAttribute = memberURL
nestedGroups = 1

[roleMap_OpenLDAP]
admin = SplunkAdmins
power = SplunkPowerUsers
user = SplunkUsers


##### Scripted Auth examples

#### The following example is for RADIUS authentication:
[authentication]
authType = Scripted
authSettings = script

[script]
scriptPath = "$SPLUNK_HOME/bin/python"
"$SPLUNK_HOME/share/splunk/authScriptSamples/radiusScripted.py"

# Cache results for 1 second per call
[cacheTiming]
userLoginTTL    = 1
getUserInfoTTL  = 1
getUsersTTL     = 1


#### The following example works with PAM authentication:
[authentication]
authType = Scripted
authSettings = script

[script]
scriptPath = "$SPLUNK_HOME/bin/python"
"$SPLUNK_HOME/share/splunk/authScriptSamples/pamScripted.py"
```

```
# Cache results for different times per function
[cacheTiming]
userLoginTTL    = 30s
getUserInfoTTL  = 1min
getUsersTTL     = 5mins


##### SAML auth example

[authentication]
authSettings = samlv2
authType = SAML

[samlv2]
attributeQuerySoapPassword = changeme
attributeQuerySoapUsername = test
entityId = test-splunk
idpAttributeQueryUrl = https://exsso/idp/attrsvc.ssaml2
idpCertPath = /home/splunk/etc/auth/idp.crt
idpSSOUrl = https://exsso/idp/SSO.saml2
idpSLOUrl = https://exsso/idp/SLO.saml2
signAuthnRequest = true
signedAssertion = true
attributeQueryRequestSigned = true
attributeQueryResponseSigned = true
redirectPort = 9332
cipherSuite = TLSv1 MEDIUM:@STRENGTH
nameIdFormat = urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

[roleMap_SAML]
admin = SplunkAdmins
power = SplunkPowerUsers
user = all

[userToRoleMap_SAML]
samluser = user

[authenticationResponseAttrMap_SAML]
role = "http://schemas.microsoft.com/ws/2008/06/identity/claims/groups"
mail =
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"
realName = "http://schemas.microsoft.com/identity/claims/displayname"

# Multifactor authentication example
[authentication]
externalTwoFactorAuthVendor = duo
externalTwoFactorAuthSettings = duo-mfa

# Duo specific authentication setting example
[duo-mfa]
apiHostname = api-xyz.duosecurity.com
```

```
appSecretKey = mustBeARandomStringOfSize40OrLonger
integrationKey = mustBeADuoProvidedStringOfSize20
secretKey = mustBeADuoProvidedStringOfSize40

##### Proxy SSO auth example

[authentication]
authSettings = my_proxy
authType = ProxySSO

[my_proxy]
blacklistedUsers = user1,user2
blacklistedAutoMappedRoles = admin
defaultRoleIfMissing = user

[roleMap_proxySSO]
admin = group1;group2
user = group1;group3

[userToRoleMap_proxySSO]
proxy_user1 = user
proxy_user2 = power;can_delete
```

authorize.conf

The following are the spec and example files for authorize.conf.

authorize.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attribute/value pairs for creating roles
in
# authorize.conf.  You can configure roles and granular access controls
by
# creating your own authorize.conf.

# There is an authorize.conf in $SPLUNK_HOME/etc/system/default/.  To
set
# custom configurations, place an authorize.conf in
# $SPLUNK_HOME/etc/system/local/. For examples, see
authorize.conf.example.
# You must restart Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please
see
```

```
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
top
#     of the file.
#   * Each conf file should have at most one default stanza. If there
are
#     multiple default stanzas, attributes are combined. In the case of
#     multiple definitions of the same attribute, the last definition in
#     the file wins.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.
```

[default]

```
[default]
srchFilterSelecting = <boolean>
* Determine's whether roles' search filters will be used for selecting
or
  eliminating during role inheritance.
* Selecting will join the search filters with an OR when combining the
  filters.
* Eliminating will join the search filters with an AND when combining
the
  filters.
    * All roles will default to true (in other words, selecting).
* Example:
  * role1 srchFilter = sourcetype!=ex1 with selecting=true
  * role2 srchFilter = sourcetype=ex2 with selecting = false
  * role3 srchFilter = sourcetype!=ex3 AND index=main with selecting =
true
  * role3 inherits from role2 and role 2 inherits from role1
  * Resulting srchFilter = ((sourcetype!=ex1) OR (sourcetype!=ex3 AND
index=main)) AND ((sourcetype=ex2))
```

[capability::<capability>]

```
[capability::<capability>]
* DO NOT edit, remove, or add capability stanzas. The existing
```

```
capabilities
  are the full set of Splunk system capabilities.
* Splunk adds all of its capabilities this way
* For the default list of capabilities and assignments, see
authorize.conf
  under the 'default' directory
* Descriptions of specific capabilities are listed below.
```

[role_<roleName>]

```
[role_<roleName>]
<capability> = <enabled>
* A capability that is enabled for this role.
* You can list many of these.
* Note that 'enabled' is the only accepted value here, as capabilities
are
  disabled by default.
* Roles inherit all capabilities from imported roles, and inherited
  capabilities cannot be disabled.
* Role names cannot have uppercase characters. User names, however, are
  case-insensitive.

importRoles = <string>
* Semicolon delimited list of other roles and their associated
capabilities
  that should be imported.
* Importing other roles also imports the other aspects of that role,
such as
  allowed indexes to search.
* By default a role imports no other roles.

grantableRoles = <string>
* Semicolon delimited list of roles that can be granted when edit_user
  capability is present.
* By default, a role with edit_user capability can create/edit a user
and
  assign any role to them. But when grantableRoles is present, the roles
  that can be assigned will be restricted to the ones provided.
* For a role that has no edit_user capability, grantableRoles has no
effect.
* Defaults to not present.
* Example: grantableRoles = role1;role2;role3

srchFilter = <string>
* Semicolon delimited list of search filters for this Role.
* By default we perform no search filtering.
* To override any search filters from imported roles, set this to '*',
as
  the 'admin' role does.
```

```
srchTimeWin = <number>
* Maximum time span of a search, in seconds.
    * This time window limit is applied backwards from the latest time
      specified in a search.
* By default, searches are not limited to any specific time window.
* To override any search time windows from imported roles, set this to
'0'
  (infinite), as the 'admin' role does.
* -1 is a special value that implies no search window has been set for
this role
    * This is equivalent to not setting srchTimeWin at all, which means
it
      can be easily overridden by an imported role

srchDiskQuota = <number>
* Maximum amount of disk space (MB) that can be used by search jobs of a
  user that belongs to this role
* Defaults to '100', for 100 MB.

srchJobsQuota = <number>
* Maximum number of concurrently running historical searches a member of
  this role can have.
* This excludes real-time searches, see rtSrchJobsQuota.
* Defaults to 3.

rtSrchJobsQuota = <number>
* Maximum number of concurrently running real-time searches a member of
this
  role can have.
* Defaults to 6.

srchMaxTime = <number><unit>
* Maximum amount of time that searches of users from this role will be
  allowed to run.
* Once the search has been ran for this amount of time it will be auto
  finalized, If the role
* Inherits from other roles, the maximum srchMaxTime value specified in
the
  included roles.
* This maximum does not apply to real-time searches.
* Examples: 1h, 10m, 2hours, 2h, 2hrs, 100s
* Defaults to 100days

srchIndexesDefault = <string>
* Semicolon delimited list of indexes to search when no index is
specified
* These indexes can be wildcarded, with the exception that '*' does not
  match internal indexes
* To match internal indexes, start with '_'. All internal indexes are
  represented by '_*'
* Defaults to none, but the UI will automatically populate this with
```

```
'main'
  in manager

srchIndexesAllowed = <string>
* Semicolon delimited list of indexes this role is allowed to search
* Follows the same wildcarding semantics as srchIndexesDefault
* Defaults to none, but the UI will automatically populate this with '*'
in
  manager

deleteIndexesAllowed = <string>
* Semicolon delimited list of indexes this role is allowed to delete
* This setting must be used in conjunction with the delete_by_keyword
  capability
* Follows the same wildcarding semantics as srchIndexesDefault
* Defaults to none

cumulativeSrchJobsQuota = <number>
* Maximum number of concurrently running historical searches in total
  across all members of this role
* Requires enable_cumulative_quota = true in limits.conf to take
effect.
* If a user belongs to multiple roles, the user's searches count
against the role with
  the largest cumulative search quota. Once the quota for that role is
consumed, the
  user's searches count against the role with the next largest quota,
and so on.
* In search head clustering environments, this setting takes effect on
a per-member basis.
  There is no cluster-wide accounting.

cumulativeRTSrchJobsQuota = <number>
* Maximum number of concurrently running real-time searches in total
  across all members of this role
* Requires enable_cumulative_quota = true in limits.conf to take
effect.
* If a user belongs to multiple roles, the user's searches count
against the role with
  the largest cumulative search quota. Once the quota for that role is
consumed, the
  user's searches count against the role with the next largest quota,
and so on.
* In search head clustering environments, this setting takes effect on
a per-member basis.
  There is no cluster-wide accounting.

### Descriptions of Splunk system capabilities
```

[capability::accelerate_datamodel]

```
[capability::accelerate_datamodel]
* Required to accelerate a datamodel.
```

[capability::admin_all_objects]

```
[capability::admin_all_objects]
* A role with this capability has access to objects in the system (user
  objects, search jobs, etc.)
* This bypasses any ACL restrictions (similar to root access in a *nix
  environment)
* We check this capability when accessing manager pages and objects
```

[capability::change_authentication]

```
[capability::change_authentication]
* Required to change authentication settings through the various
  authentication endpoints.
* Also controls whether authentication can be reloaded
```

[capability::change_own_password]

```
[capability::change_own_password]
* Self explanatory. Some auth systems prefer to have passwords be
immutable
  for some users.
```

[capability::list_storage_passwords]

```
[capability::list_storage_passwords]
* Controls access to the /storage/passwords endpoint. Users with this
capability
  can perform GETs. Note that the admin_all_objects capability is
required to
  perform POSTs to the /storage/passwords endpoint.
```

[capability::delete_by_keyword]

```
[capability::delete_by_keyword]
* Required to use the 'delete' search operator. Note that this does not
  actually delete the raw data on disk.
```

```
* Delete merely masks the data (via the index) from showing up in
search
  results.
```

[capability::edit_deployment_client]

```
[capability::edit_deployment_client]
* Self explanatory. The deployment client admin endpoint requires this
cap
  for edit.
```

[capability::list_deployment_client]

```
[capability::list_deployment_client]
* Self explanatory.
```

[capability::edit_deployment_server]

```
[capability::edit_deployment_server]
* Self explanatory. The deployment server admin endpoint requires this
cap
  for edit.
* Required to change/create remote inputs that get pushed to the
forwarders.
```

[capability::list_deployment_server]

```
[capability::list_deployment_server]
* Self explanatory.
```

[capability::edit_dist_peer]

```
[capability::edit_dist_peer]
* Required to add and edit peers for distributed search.
```

[capability::edit_forwarders]

```
[capability::edit_forwarders]
* Required to edit settings for forwarding data.
* Used by TCP and Syslog output admin handlers
* Includes settings for SSL, backoff schemes, etc.
```

[capability::edit_httpauths]

```
[capability::edit_httpauths]
* Required to edit and end user sessions through the httpauth-tokens
endpoint
```

[capability::edit_indexer_cluster]

```
[capability::edit_indexer_cluster]
* Required to edit or manage indexer cluster.
```

[capability::edit_input_defaults]

```
[capability::edit_input_defaults]
* Required to change the default hostname for input data in the server
  settings endpoint.
```

[capability::edit_monitor]

```
[capability::edit_monitor]
* Required to add inputs and edit settings for monitoring files.
* Used by the standard inputs endpoint as well as the one-shot input
  endpoint.
```

[capability::edit_modinput_winhostmon]

```
[capability::edit_modinput_winhostmon]
* Required to add and edit inputs for monitoring Windows host data.
```

[capability::edit_modinput_winnetmon]

```
[capability::edit_modinput_winnetmon]
* Required to add and edit inputs for monitoring Windows network data.
```

[capability::edit_modinput_winprintmon]

```
[capability::edit_modinput_winprintmon]
* Required to add and edit inputs for monitoring Windows printer data.
```

[capability::edit_modinput_perfmon]

```
[capability::edit_modinput_perfmon]
* Required to add and edit inputs for monitoring Windows performance.
```

[capability::edit_modinput_admon]

```
[capability::edit_modinput_admon]
* Required to add and edit inputs for monitoring Splunk's Active
Directory.
```

[capability::edit_roles]

```
[capability::edit_roles]
* Required to edit roles as well as change the mappings from users to
roles.
* Used by both the users and roles endpoint.
```

[capability::edit_roles_grantable]

```
[capability::edit_roles_grantable]
* Restrictive version of the edit_roles capability. Only allows creation
of
    roles with subset of the capabilities that the current user has as
part of
    its grantable_roles. only works in conjunction with edit_user and
grantableRoles
```

[capability::edit_scripted]

```
[capability::edit_scripted]
* Required to create and edit scripted inputs.
```

[capability::edit_search_server]

```
[capability::edit_search_server]
* Required to edit general distributed search settings like timeouts,
  heartbeats, and blacklists
```

[capability::list_introspection]

```
[capability::list_introspection]
* Required to read introspection settings and statistics for indexers,
search,
  processors, queues, etc.
* Does not permit editing introspection settings.
```

[capability::list_settings]

```
[capability::list_settings]
* Required to list general server and introspection settings such as the
server
  name, log levels, etc.
```

[capability::edit_server]

```
[capability::edit_server]
* Required to edit general server and introspection settings such as the
server
  name, log levels, etc.
* Inherits ability to read general server and introspection settings.
```

[capability::edit_search_head_clustering]

```
[capability::edit_search_head_clustering]
* Required to edit and manage search head clustering.
```

[capability::edit_search_scheduler]

```
[capability::edit_search_scheduler]
* Required to disable/enable the search scheduler.
```

[capability::edit_search_schedule_priority]

```
[capability::edit_search_schedule_priority]
* Required to give a search a higher-than-normal schedule priority.
```

[capability::edit_search_schedule_window]

```
[capability::edit_search_schedule_window]
* Required to give a search a non-automatic (or no) schedule window.
```

[capability::list_search_scheduler]

```
[capability::list_search_scheduler]
* Required to display search scheduler settings.
```

[capability::edit_sourcetypes]

```
[capability::edit_sourcetypes]
    * Required to create and edit sourcetypes.
```

[capability::edit_splunktcp]

```
[capability::edit_splunktcp]
* Required to change settings for receiving TCP input from another
Splunk
  instance.
```

[capability::edit_splunktcp_ssl]

```
[capability::edit_splunktcp_ssl]
* Required to list or edit any SSL specific settings for Splunk TCP
input.
```

[capability::edit_splunktcp_token]

```
[capability::edit_splunktcp_token]
* Required to list or edit splunktcptokens which can be used on a
receiving
  system to only accept data from forwarders that have been configured
with
  same token.
```

[capability::edit_tcp]

```
[capability::edit_tcp]
* Required to change settings for receiving general TCP inputs.
```

[capability::edit_udp]

```
[capability::edit_udp]
* Required to change settings for UDP inputs.
```

[capability::edit_telemetry_settings]

```
[capability::edit_telemetry_settings]
* Required to change settings to opt-in and send telemetry data.
```

[capability::edit_token_http]

```
[capability::edit_token_http]
* Required to create, edit, display and remove settings for HTTP token
input.
```

[capability::edit_user]

```
[capability::edit_user]
* Required to create, edit, or remove users.
* Note that Splunk users may edit certain aspects of their information
  without this capability.
* Also required to manage certificates for distributed search.
```

[capability::edit_view_html]

```
[capability::edit_view_html]
* Required to create, edit, or otherwise modify HTML-based views.
```

[capability::edit_web_settings]

```
[capability::edit_web_settings]
* Required to change the settings for web.conf through the system
settings
  endpoint.
```

[capability::get_diag]

```
[capability::get_diag]
* Required to use the /streams/diag endpoint to get remote diag from an
  instance
```

[capability::get_metadata]

```
[capability::get_metadata]
* Required to use the 'metadata' search processor.
```

[capability::get_typeahead]

```
[capability::get_typeahead]
* Required for typeahead. This includes the typeahead endpoint and the
  'typeahead' search processor.
```

[capability::input_file]

```
[capability::input_file]
* Required for inputcsv (except for dispatch=t mode) and inputlookup
```

[capability::indexes_edit]

```
[capability::indexes_edit]
* Required to change any index settings like file size and memory
limits.
```

[capability::license_tab]

```
[capability::license_tab]
* Required to access and change the license.(Deprecated)
```

[capability::license_edit]

```
[capability::license_edit]
* Required to access and change the license.
```

[capability::license_view_warnings]

```
[capability::license_view_warnings]
* Required to view license warnings on the system banner
```

[capability::list_forwarders]

```
[capability::list_forwarders]
* Required to show settings for forwarding data.
* Used by TCP and Syslog output admin handlers.
```

[capability::list_httpauths]

```
[capability::list_httpauths]
* Required to list user sessions through the httpauth-tokens endpoint.
```

[capability::list_indexer_cluster]

```
[capability::list_indexer_cluster]
* Required to list indexer cluster objects like buckets, peers etc.
```

[capability::list_inputs]

```
[capability::list_inputs]
* Required to view the list of various inputs.
* This includes input from files, TCP, UDP, Scripts, etc.
```

[capability::list_search_head_clustering]

```
[capability::list_search_head_clustering]
* Required to list search head clustering objects like artifacts,
delegated
  jobs, members, captain, etc.
```

[capability::output_file]

```
[capability::output_file]
* Required for outputcsv (except for dispatch=t mode) and outputlookup
```

[capability::request_remote_tok]

```
[capability::request_remote_tok]
* Required to get a remote authentication token.
* Used for distributing search to old 4.0.x Splunk instances.
* Also used for some distributed peer management and bundle replication.
```

[capability::rest_apps_management]

```
[capability::rest_apps_management]
* Required to edit settings for entries and categories in the python
remote
  apps handler.
* See restmap.conf for more information
```

[capability::rest_apps_view]

```
[capability::rest_apps_view]
* Required to list various properties in the python remote apps handler.
* See restmap.conf for more info
```

[capability::rest_properties_get]

```
[capability::rest_properties_get]
* Required to get information from the services/properties endpoint.
```

[capability::rest_properties_set]

```
[capability::rest_properties_set]
* Required to edit the services/properties endpoint.
```

[capability::restart_splunkd]

```
[capability::restart_splunkd]
* Required to restart Splunk through the server control handler.
```

[capability::rtsearch]

```
[capability::rtsearch]
* Required to run a realtime search.
```

[capability::run_debug_commands]

```
[capability::run_debug_commands]
* Required to run debugging commands like 'summarize'
```

[capability::schedule_search]

```
[capability::schedule_search]
* Required to schedule saved searches.
```

[capability::schedule_rtsearch]

```
[capability::schedule_rtsearch]
* Required to schedule real time saved searches. Note that
scheduled_search
  capability is also required to be enabled
```

[capability::search]

```
[capability::search]
* Self explanatory - required to run a search.
```

[capability::use_file_operator]

```
[capability::use_file_operator]
* Required to use the 'file' search operator.
```

[capability::accelerate_search]

```
[capability::accelerate_search]
* Required to save an accelerated search
* All users have this capability by default
```

[capability::web_debug]

```
[capability::web_debug]
* Required to access /_bump and /debug/** web debug endpoints
```

[capability::edit_server_crl]

```
[capability::edit_server_crl]
* Required to reload CRL information within Splunk
```

[capability::search_process_config_refresh]

```
[capability::search_process_config_refresh]
* Required to use the "refresh search-process-config" CLI command,
which
  manually flushes idle search processes.
```

[capability::extra_x509_validation]

```
[capability::extra_x509_validation]
* Required to perform additional X509 validation through
  the /server/security/extra-x509-validation.
```

authorize.conf.example

```
#   Version 6.5.0
#
# This is an example authorize.conf.  Use this file to configure roles
and
# capabilities.
#
# To use one or more of these configurations, copy the configuration
block
# into authorize.conf in $SPLUNK_HOME/etc/system/local/.  You must
reload
# auth or restart Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

[role_ninja]
rtsearch = enabled
importRoles = user
srchFilter = host=foo
srchIndexesAllowed = *
srchIndexesDefault = mail;main
srchJobsQuota   = 8
rtSrchJobsQuota = 8
srchDiskQuota   = 500

# This creates the role 'ninja', which inherits capabilities from the
'user'
# role.  ninja has almost the same capabilities as power, except cannot
```

```
# schedule searches.
#
# The search filter limits ninja to searching on host=foo.
#
# ninja is allowed to search all public indexes (those that do not start
# with underscore), and will search the indexes mail and main if no
index is
# specified in the search.
#
# ninja is allowed to run 8 search jobs and 8 real time search jobs
# concurrently (these counts are independent).
#
# ninja is allowed to take up 500 megabytes total on disk for all their
jobs.
```

checklist.conf

The following are the spec and example files for checklist.conf.

checklist.conf.spec

```
# This file contains the set of attributes and values you can use to
# configure checklist.conf in Monitoring Console.
#
```

[<uniq-check-item-name>]

```
[<uniq-check-item-name>]
* A unique string for the name of this health check.

title = <ASCII string>
* (required) Displayed title for this health check.

category = <ASCII string>
* (required) Category for overarching goups of health check items.

tags = <ASCII string>
* (optional) Comma separated list of tags that apply to this health
check.
* If omitted user will not be able to run this health check as part of a
subset of health checks.

description = <ASCII string>
* (optional) A description of what this health check is checking.
* If omitted no description will be displayed.
```

```
failure_text = <ASCII string>
* (optional) If this health check did not pass, this tells what could
have gone wrong.
* If omitted nothing will be displayed to help the user identify why
this check is failing.

suggested_action = <ASCII string>
* (optional) Suggested actions for diagnosing and fixing your Splunk
installation
  so this health check is no longer failing.
* If omitted no suggested actions for fixing this health check will be
displayed.

doc_link = <ASCII string>
* (optional) Location string for help documentation for this health
check.
* If omitted no help link will be displayed to help the user fix this
health check.

applicable_to_groups = <ASCII string>
* (optional) Comma separated list of applicable groups that this check
should be run against.
* If omitted this check item can be applied to all groups.

environments_to_exclude = <ASCII string>
* (optional) Comma separated list of environments that the health check
should not run in.
*   Possible environments are 'standalone' and 'distributed'
* If omitted this check can be applied to all groups.

disabled = [0|1]
* Disable this check item by setting to 1.
* Defaults to 0.

search = <ASCII string>
* (required) Search string to be run to perform the health check.
* Please seperate lines by "\" if the search string has multiple lines.
*
* In single-instance mode, this search will be used to generate the
final result.
* In multi-instance mode, this search will generate one row per instance
in the result table.
*
* THE SEARCH RESULT NEEDS TO BE IN THE FOLLOWING FORMAT:
* |-------------------------------------------------------------
* |     instance    |            metric         | severity_level |
* |-------------------------------------------------------------
* | <instance name> | <metric number or string> | <level number> |
* |-------------------------------------------------------------
* |       ...       |              ...          |      ...       |
* |-------------------------------------------------------------
```

```
*
* <instance name> (required, unique) is either the "host" field of
events or the
  "splunk_server" field of "| rest" search.
*   In order to generate this field, please do things like:
*     ... | rename host as instance
*   or
*     ... | rename splunk_server as instance
*
* <metric number or string> (optional) one ore more columns to "show
your work"
*   This should be the data that severity_level is determined from.
*   The user should be able to look at this field to get some idea of
what made the instance fail this check.
*
* <level number> (required) could be one of the following:
*   - -1 (N/A)                means: "Not Applicable"
*   - 0        (ok)                 means: "all good"
*         - 1 (info)                means: "just ignore it if you don't
understand"
*         - 2 (warning)         means: "well, you'd better take a look"
*         - 3 (error)                means: "FIRE!"
*
* Please also note that the search string must contain either of the
following
  token to properly scope to either a single instance or a group of
instances,
  depending on the settings of checklistsettings.conf.
*         $rest_scope$                 - used for "|rest" search
*         $hist_scope$                 - used for historical search

drilldown = <ASCII string>
* (optional) Link to a search or Monitoring Console dashboard for
additional information.
* Please note that the drilldown string must contain a $ delimited
string.
  * This string must match one of the fields output by the search.
  * Most dashboards will need the name of the instance, eg $instance$
```

checklist.conf.example

```
No example
```

collections.conf

The following are the spec and example files for collections.conf.

collections.conf.spec

```
#   Version 6.5.0
#
# This file configures the KV Store collections for a given app in
Splunk.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

[<collection-name>]

```
[<collection-name>]
enforceTypes = true|false
* Indicates whether to enforce data types when inserting data into the
  collection.
* When set to true, invalid insert operations fail.
* When set to false, invalid insert operations drop only the invalid
field.
* Defaults to false.

field.<name> = number|bool|string|time
* Field type for a field called <name>.
* If the data type is not provided, it is inferred from the provided
JSON
  data type.

accelerated_fields.<name> = <json>
* Acceleration definition for an acceleration called <name>.
* Must be a valid JSON document (invalid JSON is ignored).
* Example: 'acceleration.foo={"a":1, "b":-1}' is a compound
acceleration
  that first sorts 'a' in ascending order and then 'b' in descending
order.
* If multiple accelerations with the same definition are in the same
  collection, the duplicates are skipped.
* If the data within a field is too large for acceleration, you will
see a
  warning when you try to create an accelerated field and the
acceleration
  will not be created.
* An acceleration is always created on the _key.
* The order of accelerations is important. For example, an acceleration
of
  { "a":1, "b":1 } speeds queries on "a" and "a" + "b", but not on "b"
```

```
   lone.
* Multiple separate accelerations also speed up queries. For example,
  separate accelerations { "a": 1 } and { "b": 1 } will speed up queries
on
  "a" + "b", but not as well as a combined acceleration { "a":1, "b":1
}.
* Defaults to nothing (no acceleration).

profilingEnabled = true|false
* Indicates whether to enable logging of slow-running operations, as
defined
  in 'profilingThresholdMs'.
* Defaults to false.

profilingThresholdMs = <zero or positive integer>
* The threshold for logging a slow-running operation, in milliseconds.
* When set to 0, all operations are logged.
* This setting is only used when 'profilingEnabled' is true.
* This setting impacts the performance of the collection.
* Defaults to 1000.

replicate = true|false
* Indicates whether to replicate this collection on indexers. When
false,
  this collection is not replicated, and lookups that depend on this
  collection will not be available (although if you run a lookup command
  with 'local=true', local lookups will still be available). When true,
  this collection is replicated on indexers.
* Defaults to false.

replication_dump_strategy = one_file|auto
* Indicates how to store dump files. When set to one_file, dump files
are
  stored in a single file. When set to auto, dumps are stored in
multiple
  files when the size of the collection exceeds the value of
  'replication_dump_maximum_file_size'.
* Defaults to auto.

replication_dump_maximum_file_size = <unsigned integer>
* Specifies the maximum file size (in KB) for each dump file when
  'replication_dump_strategy=auto'.
* If this value is larger than 'concerningReplicatedFileSize', which is
set
  in distsearch.conf, the value of 'concerningReplicatedFileSize' will
be
  used instead.
* KV Store does not pre-calculate the size of the records that will be
written
  to disk, so the size of the resulting files can be affected by the
  'max_rows_in_memory_per_dump' setting from 'limits.conf'.
* Defaults to 10240KB.
```

```
type = internal_cache|undefined
* Indicates the type of data that this collection holds.
* When set to 'internal_cache', changing the configuration of the
current
  instance between search head cluster, search head pool, or standalone
  will erase the data in the collection.
* Defaults to 'undefined'.
* For internal use only.
```

collections.conf.example

```
#   Version 6.5.0
#
# The following is an example collections.conf configuration.
#
# To use one or more of these configurations, copy the configuration
block
# into collections.conf in $SPLUNK_HOME/etc/system/local/. You must
restart
# Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

[mycollection]

field.foo = number
field.bar = string
accelerated_fields.myacceleration = {"foo": 1, "bar": -1}
```

commands.conf

The following are the spec and example files for commands.conf.

commands.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attribute/value pairs for creating search
```

```
# commands for any custom search scripts created.  Add your custom
search
# script to $SPLUNK_HOME/etc/searchscripts/ or
# $SPLUNK_HOME/etc/apps/MY_APP/bin/.  For the latter, put a custom
# commands.conf in $SPLUNK_HOME/etc/apps/MY_APP.  For the former, put
your
# custom commands.conf in $SPLUNK_HOME/etc/system/local/.

# There is a commands.conf in $SPLUNK_HOME/etc/system/default/.  For
examples,
# see commands.conf.example.  You must restart Splunk to enable
configurations.

# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
top of
#     the file.
#   * Each conf file should have at most one default stanza. If there
are
#     multiple default stanzas, attributes are combined. In the case of
#     multiple definitions of the same attribute, the last definition in
the
#     file wins.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.
```

[<STANZA_NAME>]

```
[<STANZA_NAME>]
* Each stanza represents a search command; the command is the stanza
name.
* The stanza name invokes the command in the search language.
* Set the following attributes/values for the command.  Otherwise,
Splunk uses
  the defaults.
* If the filename attribute is not specified, Splunk searches for an
  external program by appending extensions (e.g. ".py", ".pl") to the
```

```
  stanza name.
* If chunked = true, in addition to ".py" and ".pl" as above, Splunk
  searches using the extensions ".exe", ".bat", ".cmd", ".sh", ".js",
  and no extension (to find extensionless binaries).
* See the filename attribute for more information about how Splunk
  searches for external programs.

type = <string>
* Type of script: python, perl
* Defaults to python.

filename = <string>
* Optionally specify the program to be executed when the search command
is used.
* Splunk looks for the given filename in the app's bin directory.
* The filename attribute can not reference any file outside of the app's
bin directory.
* If the filename ends in ".py", Splunk's python interpreter is used
  to invoke the external script.
* If chunked = true, Splunk looks for the given filename in
  $SPLUNK_HOME/etc/apps/MY_APP/<PLATFORM>/bin before searching
  $SPLUNK_HOME/etc/apps/MY_APP/bin, where <PLATFORM> is one of
  "linux_x86_64", "linux_x86", "windows_x86_64", "windows_x86",
  "darwin_x86_64" (depending on the platform on which Splunk is
  running on).
* If chunked = true and if a path pointer file (*.path) is specified,
  the contents of the file are read and the result is used as the
  command to be run. Environment variables in the path pointer
  file are substituted. Path pointer files can be used to reference
  system binaries (e.g. /usr/bin/python).

command.arg.<N> = <string>
* Additional command-line arguments to use when invoking this
  program. Environment variables will be substituted (e.g.
$SPLUNK_HOME).
* Only available if chunked = true.

local = [true|false]
* If true, specifies that the command should be run on the search head
only
* Defaults to false

perf_warn_limit = <integer>
* Issue a performance warning message if more than this many input
events are
  passed to this external command (0 = never)
* Defaults to 0 (disabled)

streaming = [true|false]
* Specify whether the command is streamable.
* Defaults to false.
```

```
maxinputs = <integer>
* Maximum number of events that can be passed to the command for each
  invocation.
* This limit cannot exceed the value of maxresultrows in limits.conf.
* 0 for no limit.
* Defaults to 50000.

passauth = [true|false]
* If set to true, splunkd passes several authentication-related facts
  at the start of input, as part of the header (see enableheader).
* The following headers are sent
  * authString: psuedo-xml string that resembles
      <auth><userId>username</userId><username>username</username><authToken>auth_token
    where the username is passed twice, and the authToken may be used
    to contact splunkd during the script run.
  * sessionKey: the session key again.
  * owner: the user portion of the search context
  * namespace: the app portion of the search context
* Requires enableheader = true; if enableheader = false, this flag will
  be treated as false as well.
* Defaults to false.
* If chunked = true, this attribute is ignored. An authentication
  token is always passed to commands using the chunked custom search
  command protocol.

run_in_preview = [true|false]
* Specify whether to run this command if generating results just for
preview
  rather than final output.
* Defaults to true

enableheader = [true|false]
* Indicate whether or not your script is expecting header information or
not.
* Currently, the only thing in the header information is an auth token.
* If set to true it will expect as input a head section + '\n' then the
csv input
* NOTE: Should be set to true if you use splunk.Intersplunk
* Defaults to true.

retainsevents = [true|false]
* Specify whether the command retains events (the way the
sort/dedup/cluster
  commands do) or whether it transforms them (the way the stats command
does).
* Defaults to false.

generating = [true|false]
* Specify whether your command generates new events. If no events are
passed to
  the command, will it generate events?
* Defaults to false.
```

```
generates_timeorder = [true|false]
* If generating = true, does command generate events in descending time
order
  (latest first)
* Defaults to false.

overrides_timeorder = [true|false]
* If generating = false and streaming=true, does command change the
order of
  events with respect to time?
* Defaults to false.

requires_preop = [true|false]
* Specify whether the command sequence specified by the
'streaming_preop' key
  is required for proper execution or is it an optimization only
* Default is false (streaming_preop not required)

streaming_preop = <string>
* A string that denotes the requested pre-streaming search string.

required_fields = <string>
* A comma separated list of fields that this command may use.
* Informs previous commands that they should retain/extract these fields
if
  possible.  No error is generated if a field specified is missing.
* Defaults to '*'

supports_multivalues = [true|false]
* Specify whether the command supports multivalues.
* If true, multivalues will be treated as python lists of strings,
instead of a
  flat string (when using Intersplunk to interpret stdin/stdout).
* If the list only contains one element, the value of that element will
be
  returned, rather than a list
  (for example, isinstance(val, basestring) == True).

supports_getinfo = [true|false]
* Specifies whether the command supports dynamic probing for settings
  (first argument invoked == __GETINFO__ or __EXECUTE__).

supports_rawargs = [true|false]
* Specifies whether the command supports raw arguments being passed to
it or if
  it prefers parsed arguments (where quotes are stripped).
* If unspecified, the default is false

undo_scheduler_escaping = [true|false]
* Specifies whether the commands raw arguments need to be unesacped.
* This is perticularly applies to the commands being invoked by the
```

```
scheduler.
* This applies only if the command supports raw
arguments(supports_rawargs).
* If unspecified, the default is false

requires_srinfo = [true|false]
* Specifies if the command requires information stored in
SearchResultsInfo.
* If true, requires that enableheader be set to true, and the full
  pathname of the info file (a csv file) will be emitted in the header
under
  the key 'infoPath'
* If unspecified, the default is false


needs_empty_results = [true|false]
* Specifies whether or not this search command needs to be called with
  intermediate empty search results
* If unspecified, the default is true

changes_colorder = [true|false]
* Specify whether the script output should be used to change the column
  ordering of the fields.
* Default is true

outputheader = <true/false>
* If set to true, output of script should be
  a header section + blank line + csv output
* If false, script output should be pure csv only
* Default is false

clear_required_fields = [true|false]
* If true, required_fields represents the *only* fields required.
* If false, required_fields are additive to any fields that may be
required by
  subsequent commands.
* In most cases, false is appropriate for streaming commands and true
for
  reporting commands
* Default is false

stderr_dest = [log|message|none]
* What do to with the stderr output from the script
* 'log' means to write the output to the job's search.log.
* 'message' means to write each line as an search info message.  The
message
  level can be set to adding that level (in ALL CAPS) to the start of
the
  line, e.g. "WARN my warning message."
* 'none' means to discard the stderr output
* Defaults to log
```

```
is_order_sensitive = [true|false]
* Specify whether the command requires ordered input.
* Defaults to false.

is_risky = [true|false]
* Searches using Splunk Web are flagged to warn users when they
  unknowingly run a search that contains commands that might be a
  security risk. This warning appears when users click a link or type
  a URL that loads a search that contains risky commands. This warning
  does not appear when users create ad hoc searches.
* This flag is used to determine whether the command is risky.
* Defaults to false.
* - Specific commands that ship with the product have their own defaults

chunked = [true|false]
* If true, this command supports the new "chunked" custom
  search command protocol.
* If true, the only other commands.conf attributes supported are
  is_risky, maxwait, maxchunksize, filename, and command.arg.<N>.
* If false, this command uses the legacy custom search command
  protocol supported by Intersplunk.py.
* Default is false

maxwait = <integer>
* Only available if chunked = true.
* Not supported in Windows.
* The value of maxwait is the maximum number of seconds the custom
  search command can pause before producing output.
* If set to 0, the command can pause forever.
* Default is 0

maxchunksize = <integer>
* Only available if chunked = true.
* The value of maxchunksize is maximum size chunk (size of metadata
  plus size of body) the external command may produce. If the command
  tries to produce a larger chunk, the command is terminated.
* If set to 0, the command may send any size chunk.
* Default is 0
```

commands.conf.example

```
#   Version 6.5.0
#
# Configuration for external search commands
#

###############
# defaults for all external commands, exceptions are below in individual
# stanzas
```

```
# type of script: 'python', 'perl'
TYPE = python
# default FILENAME would be <stanza-name>.py for python,
<stanza-name>.pl for
# perl and <stanza-name> otherwise

# is command streamable?
STREAMING = false

# maximum data that can be passed to command (0 = no limit)
MAXINPUTS = 50000

# end defaults
######################

[crawl]
FILENAME = crawl.py

[createrss]
FILENAME = createrss.py

[diff]
FILENAME = diff.py

[gentimes]
FILENAME = gentimes.py

[head]
FILENAME = head.py

[loglady]
FILENAME = loglady.py

[marklar]
FILENAME = marklar.py

[runshellscript]
FILENAME = runshellscript.py

[sendemail]
FILENAME = sendemail.py

[translate]
FILENAME = translate.py

[transpose]
FILENAME = transpose.py

[uniq]
FILENAME = uniq.py
```

```
[windbag]
filename = windbag.py
supports_multivalues = true

[xmlkv]
FILENAME = xmlkv.py

[xmlunescape]
FILENAME = xmlunescape.py
```

crawl.conf

The following are the spec and example files for crawl.conf.

crawl.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attribute/value pairs for configuring
crawl.
#
# There is a crawl.conf in $SPLUNK_HOME/etc/system/default/.  To set
custom
# configurations, place a crawl.conf in $SPLUNK_HOME/etc/system/local/.
For
# help, see crawl.conf.example. You must restart Splunk to enable
# configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
#
# Set of attribute-values used by crawl.
#
# If attribute, ends in _list, the form is:
#
#      attr = val, val, val, etc.
#
# The space after the comma is necessary, so that "," can be used, as
in
# BAD_FILE_PATTERNS's use of "*,v"
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
top of
#     the file.
#   * Each conf file should have at most one default stanza. If there
are
#     multiple default stanzas, attributes are combined. In the case of
#     multiple definitions of the same attribute, the last definition in
the
#     file wins.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.
```

[default]

```
[default]
```

[files]

```
[files]
* Sets file crawler-specific attributes under this stanza header.
* Follow this stanza name with any of the following attributes.

root = <semi-colon separate list of directories>
* Set a list of directories this crawler should search through.
* Defaults to /;/Library/Logs

bad_directories_list = <comma-separated list of bad directories>
* List any directories you don't want to crawl.
* Defaults to:
        bin, sbin, boot, mnt, proc, tmp, temp, dev, initrd, help,
driver, drivers, share, bak, old, lib, include, doc, docs, man, html,
images, tests, js, dtd, org, com, net, class, java, resource, locale,
static, testing, src, sys, icons, css, dist, cache, users, system,
resources, examples, gdm, manual, spool, lock, kerberos, .thumbnails,
libs, old, manuals, splunk, splunkpreview, mail, resources,
documentation, applications, library, network, automount, mount, cores,
lost\+found, fonts, extensions, components, printers, caches, findlogs,
music, volumes, libexec

bad_extensions_list = <comma-separated list of file extensions to skip>
* List any file extensions and crawl will skip files that end in those
extensions.
```

```
* Defaults to:
        0t, a, adb, ads, ali, am, asa, asm, asp, au, bak, bas, bat, bmp,
c, cache, cc, cg, cgi, class, clp, com, conf, config, cpp, cs, css,
csv, cxx, dat, doc, dot, dvi, dylib, ec, elc, eps, exe, f, f77, f90,
for, ftn, gif, h, hh, hlp, hpp, hqx, hs, htm, html, hxx, icns, ico,
ics, in, inc, jar, java, jin, jpeg, jpg, js, jsp, kml, la, lai, lhs,
lib, license, lo, m, m4, mcp, mid, mp3, mpg, msf, nib, nsmap, o, obj,
odt, ogg, old, ook, opt, os, os2, pal, pbm, pdf, pdf, pem, pgm, php,
php3, php4, pl, plex, plist, plo, plx, pm, png, po, pod, ppd, ppm, ppt,
prc, presets, ps, psd, psym, py, pyc, pyd, pyw, rast, rb, rc, rde, rdf,
rdr, res, rgb, ro, rsrc, s, sgml, sh, shtml, so, soap, sql, ss, stg,
strings, tcl, tdt, template, tif, tiff, tk, uue, v, vhd, wsdl, xbm, xlb,
xls, xlw, xml, xsd, xsl, xslt, jame, d, ac, properties, pid, del, lock,
md5, rpm, pp, deb, iso, vim, lng, list

bad_file_matches_list = <comma-separated list of regex>
* Crawl applies the specified regex and skips files that match the
patterns.
* There is an implied "$" (end of file name) after each pattern.
* Defaults to:
        *~, *#, *,v, *readme*, *install, (/|^).*, *passwd*, *example*,
*makefile, core.*

packed_extensions_list = <comma-separated list of extensions>
* Specify extensions of compressed files to exclude.
* Defaults to:
        bz, bz2, tbz, tbz2, Z, gz, tgz, tar, zip

collapse_threshold = <integer>
* Specify the minimum number of files a source must have to be
considered a
  directory.
* Defaults to 1000.

days_sizek_pairs_list = <comma-separated hyphenated pairs of integers>
* Specify a comma-separated list of age (days) and size (kb) pairs to
constrain
  what files are crawled.
* For example: days_sizek_pairs_list = 7-0, 30-1000 tells Splunk to
crawl only
  files last modified within 7 days and at least 0kb in size, or
modified
  within the last 30 days and at least 1000kb in size.
* Defaults to 30-0.

big_dir_filecount = <integer>
* Skip directories with files above <integer>
* Defaults to 10000.

index = <$INDEX>
* Specify index to add crawled files to.
* Defaults to main.
```

```
max_badfiles_per_dir = <integer>
* Specify how far to crawl into a directory for files.
* Crawl excludes a directory if it doesn't find valid files within the
  specified max_badfiles_per_dir.
* Defaults to 100.
```

[network]

```
[network]
* Sets network crawler-specific attributes under this stanza header.
* Follow this stanza name with any of the following attributes.

host = <host or ip>
* default host to use as a starting point for crawling a network
* Defaults to 'localhost'.

subnet = <int>
* default number of bits to use in the subnet mask. Given a host with IP
  123.123.123.123, a subnet value of 32, would scan only that host, and
a value
  or 24 would scan 123.123.123.*.
* Defaults to 32.
```

crawl.conf.example

```
#   Version 6.5.0
#
# The following are example crawl.conf configurations. Configure
properties for
# crawl.
#
# To use one or more of these configurations, copy the configuration
block into
# crawl.conf in $SPLUNK_HOME/etc/system/local/. You must restart Splunk
to
# enable configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

[files]
bad_directories_list= bin, sbin, boot, mnt, proc, tmp, temp, home,
mail, .thumbnails, cache, old
```

```
bad_extensions_list= mp3, mpg, jpeg, jpg,  m4, mcp, mid
bad_file_matches_list= *example*, *makefile, core.*
packed_extensions_list= gz, tgz, tar, zip
collapse_threshold= 10
days_sizek_pairs_list= 3-0,7-1000, 30-10000
big_dir_filecount= 100
index=main
max_badfiles_per_dir=100


[network]
host = myserver
subnet = 24
```

datamodels.conf

The following are the spec and example files for datamodels.conf.

datamodels.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attribute/value pairs for configuring
# data models.  To configure a datamodel for an app, put your custom
# datamodels.conf in $SPLUNK_HOME/etc/apps/MY_APP/local/

# For examples, see datamodels.conf.example.  You must restart Splunk
to
# enable configurations.

# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
top
#     of the file.
```

```
#   * Each conf file should have at most one default stanza. If there
are
#     multiple default stanzas, attributes are combined. In the case of
#     multiple definitions of the same attribute, the last definition in
the
#     file wins.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.
```

[<datamodel_name>]

```
[<datamodel_name>]
* Each stanza represents a data model. The data model name is the stanza
name.

acceleration = <bool>
* Set acceleration to true to enable automatic acceleration of this data
model.
* Automatic acceleration creates auxiliary column stores for the fields
  and values in the events for this datamodel on a per-bucket basis.
* These column stores take additional space on disk, so be sure you
have the
  proper amount of disk space. Additional space required depends on the
  number of events, fields, and distinct field values in the data.
* The Splunk software creates and maintains these column stores on a
schedule
  you can specify with 'acceleration.cron_schedule.' You can query
  them with the 'tstats' command.

acceleration.earliest_time = <relative-time-str>
* Specifies how far back in time the Splunk software should keep these
column
  stores (and create if acceleration.backfill_time is not set).
* Specified by a relative time string. For example, '-7d' means
'accelerate
  data within the last 7 days.'
* Defaults to an empty string, meaning 'keep these stores for all time.'

acceleration.backfill_time = <relative-time-str>
* ADVANCED: Specifies how far back in time the Splunk software should
create
  its column stores.
* ONLY set this parameter if you want to backfill less data than the
  retention period set by 'acceleration.earliest_time'. You may want to
use
  this parameter to limit your time window for column store creation in
a large
  environment where initial creation of a large set of column stores is
an
```

```
  expensive operation.
* WARNING: Do not set 'acceleration.backfill_time' to a
  narrow time window. If one of your indexers is down for a period
longer
  than this backfill time, you may miss accelerating a window of your
incoming
  data.
* MUST be set to a more recent time than 'acceleration.earliest_time'.
For
  example, if you set 'acceleration.earliest_time' to '-1y' to retain
your
  column stores for a one year window, you could set
'acceleration.backfill_time'
  to '-20d' to create column stores that only cover the last 20 days.
However,
  you cannot set 'acceleration.backfill_time' to '-2y', because that
goes
  farther back in time than the 'acceleration.earliest_time' setting of
'-1y'.
* Defaults to empty string (unset). When 'acceleration.backfill_time'
is unset,
  the Splunk software always backfills fully to
'acceleration.earliest_time.'

acceleration.max_time = <unsigned int>
* The maximum amount of time that the column store creation search is
  allowed to run (in seconds).
* Note that this is an approximate time, as the 'summarize' search only
  finishes on clean bucket boundaries to avoid wasted work.
* Defaults to: 3600
* An 'acceleration.max_time' setting of '0' indicates that there is no
time
  limit.

acceleration.cron_schedule = <cron-string>
* Cron schedule to be used to probe/generate the column stores for this
  data model.
* Defaults to: */5 * * * *

acceleration.manual_rebuilds = <bool>
* ADVANCED: When set to 'true,' this setting prevents outdated summaries
from
  being rebuilt by the 'summarize' command.
* Normally, during the creation phase, the 'summarize' command
automatically
  rebuilds summaries that are considered to be out-of-date, such as when
the
  configuration backing the data model changes.
* The Splunk software considers a summary to be outdated when:
        * The data model search stored in its metadata no longer matches
its current
          data model search.
```

```
        * The search stored in its metadata cannot be parsed.
    * A lookup table associated with the data model is altered.
* NOTE: If the Splunk software finds a partial summary be outdated, it
always
  rebuilds that summary so that a bucket summary only has results
corresponding to
  one datamodel search.
* Defaults to: false

acceleration.max_concurrent = <unsigned int>
* The maximum number of concurrent acceleration instances for this data
  model that the scheduler is allowed to run.
* Defaults to: 2

acceleration.schedule_priority = default | higher | highest
* Raises the scheduling priority of a search:
  + "default": No scheduling priority increase.
  + "higher": Scheduling priority is higher than other data model
searches.
  + "highest": Scheduling priority is higher than other searches
regardless of
    scheduling tier except real-time-scheduled searches with priority =
highest
    always have priority over all other searches.
  + Hence, the high-to-low order (where RTSS = real-time-scheduled
search, CSS
    = continuous-scheduled search, DMAS = data-model-accelerated
search, d =
    default, h = higher, H = highest) is:
      RTSS(H) > DMAS(H) > CSS(H)
      > RTSS(h) > RTSS(d) > CSS(h) > CSS(d)
      > DMAS(h) > DMAS(d)
* The scheduler honors a non-default priority only when the search owner
has
  the 'edit_search_schedule_priority' capability.
* Defaults to: default
* WARNING: Having too many searches with a non-default priority will
impede the
  ability of the scheduler to minimize search starvation.  Use this
setting
  only for mission-critical searches.

acceleration.hunk.compression_codec = <string>
* Applicable only to Hunk Data models. Specifies the compression codec
to
  be used for the accelerated orc/parquet files.

acceleration.hunk.dfs_block_size = <unsigned int>
* Applicable only to Hunk data models. Specifies the block size in bytes
for
  the compression files.
```

```
acceleration.hunk.file_format = <string>
* Applicable only to Hunk data models. Valid options are "orc" and
"parquet"


#******** Dataset Related Attributes ******
# These attributes affect your interactions with datasets in Splunk Web
and should
# not be changed under normal conditions. Do not modify them unless you
are sure you
# know what you are doing.

dataset.description = <string>
* User-entered description of the dataset entity.

dataset.type = [datamodel|table]
* The type of dataset:
  + "datamodel": An individual data model dataset.
  + "table": A special root data model dataset with a search where the
dataset is
    defined by the dataset.commands attribute.
* Default: datamodel

dataset.commands = [<object>(, <object>)*]
* When the dataset.type = "table" this stringified JSON payload is
created by the
  table editor and defines the dataset.

dataset.fields = [<string>(, <string>)*]
* Automatically generated JSON payload when dataset.type = "table" and
the root
  data model dataset's search is updated.

dataset.display.diversity = [latest|random|diverse|rare]
* The user-selected diversity for previewing events contained by the
dataset:
  + "latest": search a subset of the latest events
  + "random": search a random sampling of events
  + "diverse": search a diverse sampling of events
  + "rare": search a rare sampling of events based on clustering
* Default: latest

dataset.display.sample_ratio = <int>
* The integer value used to calculate the sample ratio for the dataset
diversity.
  The formula is 1 / <int>.
* The sample ratio specifies the likelihood of any event being included
in the
  sample.
* For example, if sample_ratio = 500 each event has a 1/500 chance of
being
  included in the sample result set.
```

```
* Default: 1

dataset.display.limiting = <int>
* The limit of events to search over when previewing the dataset.
* Default: 100000

dataset.display.currentCommand = <int>
* The currently selected command the user is on while editing the
dataset.

dataset.display.mode = [table|datasummary]
* The type of preview to use when editing the dataset:
  + "table": show individual events/results as rows.
  + "datasummary": show field values as columns.
* Default: table

dataset.display.datasummary.earliestTime = <time-str>
* The earliest time used for the search that powers the datasummary view
of
  the dataset.

dataset.display.datasummary.latestTime = <time-str>
* The latest time used for the search that powers the datasummary view
of
  the dataset.
```

datamodels.conf.example

```
#   Version 6.5.0
#
# Configuration for example datamodels
#

# An example of accelerating data for the 'mymodel' datamodel for the
# past five days, generating and checking the column stores every 10
minutes
[mymodel]
acceleration = true
acceleration.earliest_time = -5d
acceleration.cron_schedule = */10 * * * *
acceleration.hunk.compression_codec = snappy
acceleration.hunk.dfs_block_size = 134217728
acceleration.hunk.file_format = orc
```

datatypesbnf.conf

The following are the spec and example files for datatypesbnf.conf.

datatypesbnf.conf.spec

```
#   Version 6.5.0
#
# This file effects how the search assistant (typeahead) shows the
syntax for
# search commands
```

[<syntax-type>]

```
[<syntax-type>]
* The name of the syntax type you're configuring.
* Follow this field name with one syntax= definition.
* Syntax type can only contain a-z, and -, but cannot begin with -

syntax = <string>
* The syntax for you syntax type.
* Should correspond to a regular expression describing the term.
* Can also be a <field> or other similar value.
```

datatypesbnf.conf.example

```
No example
```

default.meta.conf

The following are the spec and example files for default.meta.conf.

default.meta.conf.spec

```
#   Version 6.5.0
#
#
# *.meta files contain ownership information, access controls, and
export
# settings for Splunk objects like saved searches, event types, and
```

```
views.
# Each app has its own default.meta file.

# Interaction of ACLs across app-level, category level, and specific
object
# configuration:
* To access/use an object, users must have read access to:
  * the app containing the object
  * the generic category within the app (eg [views])
  * the object itself
* If any layer does not permit read access, the object will not be
accessible.

* To update/modify an object, such as to edit a saved search, users
must have:
  * read and write access to the object
  * read access to the app, to locate the object
  * read access to the generic category within the app (eg.
[savedsearches])
* If object does not permit write access to the user, the object will
not be
  modifiable.
* If any layer does not permit read access to the user, the object will
not be
  accessible in order to modify

* In order to add or remove objects from an app, users must have:
  * write access to the app
* If users do not have write access to the app, an attempt to add or
remove an
  object will fail.


* Objects that are exported to other apps or to system context have no
change
  to their accessibility rules.  Users must still have read access to
the
  containing app, category, and object, despite the export.

# Set access controls on the app containing this metadata file.
[]
access = read : [ * ], write : [ admin, power ]
* Allow all users to read this app's contents. Unless overridden by
other
  metadata, allow only admin and power users to share objects into this
app.

# Set access controls on this app's views.
```

[views]

```
[views]
access = read : [ * ], write : [ admin ]
* Allow all users to read this app's views. Allow only admin users to
create,
  remove, share, or unshare views in this app.

# Set access controls on a specific view in this app.
```

[views/index_status]

```
[views/index_status]
access = read : [ admin ], write : [ admin ]
* Allow only admin users to read or modify this view.

# Make this view available in all apps.
export = system
* To make this view available only in this app, set 'export = none'
instead.
owner = admin
* Set admin as the owner of this view.
```

default.meta.conf.example

```
#   Version 6.5.0
#
# This file contains example patterns for the metadata files
default.meta and
# local.meta
#

# This example would make all of the objects in an app globally
accessible to
# all apps
[]
export=system
```

default-mode.conf

The following are the spec and example files for default-mode.conf.

default-mode.conf.spec

```
#   Version 6.5.0
#
# This file documents the syntax of default-mode.conf for comprehension
and
# troubleshooting purposes.

# default-mode.conf is a file that exists primarily for Splunk Support
and
# Services to configure splunk.

# CAVEATS:

# DO NOT make changes to default-mode.conf without coordinating with
Splunk
# Support or Services.  End-user changes to default-mode.conf are not
# supported.
#
# default-mode.conf *will* be removed in a future version of Splunk,
along
# with the entire configuration scheme that it affects. Any settings
present
# in default-mode.conf files will be completely ignored at this point.
#
# Any number of seemingly reasonable configurations in
default-mode.conf
# might fail to work, behave bizarrely, corrupt your data, iron your
cat,
# cause unexpected rashes, or order unwanted food delivery to your
house.
# Changes here alter the way that pieces of code will communicate which
are
# only intended to be used in a specific configuration.


# INFORMATION:

# The main value of this spec file is to assist in reading these files
for
# troubleshooting purposes.  default-mode.conf was originally intended
to
# provide a way to describe the alternate setups used by the Splunk
Light
# Forwarder and Splunk Universal Forwarder.

# The only reasonable action is to re-enable input pipelines that are
# disabled by default in those forwarder configurations.  However, keep
the
# prior caveats in mind.  Any future means of enabling inputs will have
```

```
a
# different form when this mechanism is removed.

# SYNTAX:
```

[pipeline:<string>]

```
[pipeline:<string>]
disabled = true | false
disabled_processors = <string>
```

[pipeline:<string>]

```
[pipeline:<string>]
* Refers to a particular Splunkd pipeline.
* The set of named pipelines is a splunk-internal design.  That does
not
  mean that the Splunk design is a secret, but it means it is not
external
  for the purposes of configuration.
* Useful information on the data processing system of splunk can be
found
  in the external documentation, for example
  http://docs.splunk.com/Documentation/Splunk/latest/Deploy/Datapipeline


disabled = true | false
* If set to true on a specific pipeline, the pipeline will not be
loaded in
  the system.

disabled_processors = <processor1>, <processor2>
* Processors which normally would be loaded in this pipeline are not
loaded
  if they appear in this list
* The set of named procoessors is again a splunk-internal design
component.
```

default-mode.conf.example

```
No example
```

deployment.conf

The following are the spec and example files for deployment.conf.

deployment.conf.spec

```
#   Version 6.5.0
#
# *** REMOVED; NO LONGER USED ***
#
#
# This configuration file  has been replaced by:
# 1.) deploymentclient.conf - for configuring Deployment Clients.
# 2.) serverclass.conf - for Deployment Server server class
configuration.
#
#
# Compatibility:
# Splunk 4.x Deployment Server is NOT compatible with Splunk 3.x
Deployment Clients.
#
```

deployment.conf.example

```
No example
```

deploymentclient.conf

The following are the spec and example files for deploymentclient.conf.

deploymentclient.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attributes and values for configuring a
# deployment client to receive content (apps and configurations) from a
# deployment server.
#
# To customize the way a deployment client behaves, place a
# deploymentclient.conf in $SPLUNK_HOME/etc/system/local/ on that
Splunk
```

```
# instance. Configure what apps or configuration content is deployed to
a
# given deployment client in serverclass.conf.  Refer to
# serverclass.conf.spec and serverclass.conf.example for more
information.
#
# You must restart Splunk for changes to this configuration file to
take
# effect.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

#*******************************************************************************
# Configure a Splunk deployment client.
#
# Note: At a minimum the [deployment-client] stanza is required in
# deploymentclient.conf for deployment client to be enabled.
#*******************************************************************************
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
top
#     of the file.
#   * Each conf file should have at most one default stanza. If there
are
#     multiple default stanzas, attributes are combined. In the case of
#     multiple definitions of the same attribute, the last definition in
the
#     file wins.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.
```

[deployment-client]

```
[deployment-client]
disabled = [false|true]
* Defaults to false
* Enable/Disable deployment client.
```

```
clientName = deploymentClient
* Defaults to deploymentClient.
* A name that the deployment server can filter on.
* Takes precedence over DNS names.

workingDir = $SPLUNK_HOME/var/run
* Temporary folder used by the deploymentClient to download apps and
  configuration content.

repositoryLocation = $SPLUNK_HOME/etc/apps
* The location into which content is installed after being downloaded
from a
  deployment server.
* Apps and configuration content must be installed into the default
location
  ($SPLUNK_HOME/etc/apps) or it will not be recognized by
  the Splunk instance on the deployment client.
    * Note: Apps and configuration content to be deployed may be
located in
      an alternate location on the deployment server. Set both
      repositoryLocation and serverRepositoryLocationPolicy explicitly
to
      ensure that the content is installed into the correct location
      ($SPLUNK_HOME/etc/apps) on the deployment clientr
    * The deployment client uses the 'serverRepositoryLocationPolicy'
      defined below to determine which value of repositoryLocation to
use.

serverRepositoryLocationPolicy =
[acceptSplunkHome|acceptAlways|rejectAlways]
* Defaults to acceptSplunkHome.
* acceptSplunkHome - accept the repositoryLocation supplied by the
                     deployment server, only if it is rooted by
                     $SPLUNK_HOME.
* acceptAlways - always accept the repositoryLocation supplied by the
                 deployment server.
* rejectAlways - reject the server supplied value and use the
                 repositoryLocation specified in the local
                 deploymentclient.conf.

endpoint=$deploymentServerUri$/services/streams/deployment?name=$serverClassName$:$appN
* The HTTP endpoint from which content should be downloaded.
* Note: The deployment server may specify a different endpoint from
which to
  download each set of content (individual apps, etc).
* The deployment client will use the serverEndpointPolicy defined below
to
  determine which value to use.
* $deploymentServerUri$ will resolve to targetUri defined in the
  [target-broker] stanza below.
* $serverClassName$ and $appName$ mean what they say.
```

```
serverEndpointPolicy = [acceptAlways|rejectAlways]
* defaults to acceptAlways
* acceptAlways - always accept the endpoint supplied by the server.
* rejectAlways - reject the endpoint supplied by the server. Always use
the
                 'endpoint' definition above.

phoneHomeIntervalInSecs = <number in seconds>
* Defaults to 60.
* Fractional seconds are allowed.
* This determines how frequently this deployment client should check for
new
  content.

handshakeRetryIntervalInSecs = <number in seconds>
* Defaults to one fifth of phoneHomeIntervalInSecs
* Fractional seconds are allowed.
* This sets the handshake retry frequency.
* Could be used to tune the initial connection rate on a new server

handshakeReplySubscriptionRetry = <integer>
* Defaults to 10
* If splunk is unable to complete the handshake, it will retry
subscribing to
  the handshake channel after this many handshake attempts

appEventsResyncIntervalInSecs = <number in seconds>
* Defaults to 10*phoneHomeIntervalInSecs
* Fractional seconds are allowed.
* This sets the interval at which the client reports back its app state
to the server.

# Advanced!
# You should use this property only when you have a hierarchical
deployment
# server installation, and have a Splunk instance that behaves as both a
# DeploymentClient and a DeploymentServer.

# NOTE: hierarchical deployment servers are not a currently recommended
# configuration.  Splunk has seen problems in the field that have not
yet
# been resolved with this type of configuration.

reloadDSOnAppInstall = [false|true]
* Defaults to false
* Setting this flag to true will cause the deploymentServer on this
Splunk
  instance to be reloaded whenever an app is installed by this
  deploymentClient.

sslVersions = <versions_list>
* Comma-separated list of SSL versions to connect to the specified
```

```
Deployment Server
* The versions available are "ssl3", "tls1.0", "tls1.1", and "tls1.2".
* The special version "*" selects all supported versions.  The version
"tls"
  selects all versions tls1.0 or newer.
* If a version is prefixed with "-" it is removed from the list.
* SSLv2 is always disabled; "-ssl2" is accepted in the version list but
does nothing.
* When configured in FIPS mode, ssl3 is always disabled regardless
  of this configuration.
* Defaults to sslVersions value in server.conf [sslConfig] stanza.

sslVerifyServerCert = <bool>
* If this is set to true, Splunk verifies that the Deployment Server
(specified in 'targetUri')
  being connected to is a valid one (authenticated).  Both the common
  name and the alternate name of the server are then checked for a
  match if they are specified in 'sslCommonNameToCheck' and
'sslAltNameToCheck'.
  A certificiate is considered verified if either is matched.
* Defaults to sslVerifyServerCert value in server.conf [sslConfig]
stanza.

caCertFile = <path>
* Full path to a CA (Certificate Authority) certificate(s) PEM format
file.
* The <path> must refer to a PEM format file containing one or more
root CA
  certificates concatenated together.
* Used for validating SSL certificate from Deployment Server
* Defaults to caCertFile value in server.conf [sslConfig] stanza.

sslCommonNameToCheck = <commonName1>, <commonName2>, ...
* If this value is set, and 'sslVerifyServerCert' is set to true,
  splunkd checks the common name(s) of the certificate presented by
  the Deployment Server (specified in 'targetUri') against this list of
common names.
* Defaults to sslCommonNameToCheck value in server.conf [sslConfig]
stanza.

sslAltNameToCheck =  <alternateName1>, <alternateName2>, ...
* If this value is set, and 'sslVerifyServerCert' is set to true,
  splunkd checks the alternate name(s) of the certificate presented by
  the Deployment Server (specified in 'targetUri') against this list of
subject alternate names.
* Defaults to sslAltNameToCheck value in server.conf [sslConfig]
stanza.

cipherSuite = <cipher suite string>
* If set, uses the specified cipher string for making outbound HTTPS
connection.
```

```
ecdhCurves = <comma separated list of ec curves>
* ECDH curves to use for ECDH key negotiation.
* The curves should be specified in the order of preference.
* The client sends these curves as a part of Client Hello.
* We only support named curves specified by their SHORT names.
  (see struct ASN1_OBJECT in asn1.h)
* The list of valid named curves by their short/long names can be
obtained
  by executing this command:
  $SPLUNK_HOME/bin/splunk cmd openssl ecparam -list_curves
* Default is empty string.
* e.g. ecdhCurves = prime256v1,secp384r1,secp521r1

# The following stanza specifies deployment server connection
information
```

[target-broker:deploymentServer]

```
[target-broker:deploymentServer]
targetUri= <deploymentServer>:<mgmtPort>
* URI of the deployment server.

phoneHomeIntervalInSecs = <nonnegative number>
* see phoneHomeIntervalInSecs above
```

deploymentclient.conf.example

```
#   Version 6.5.0
#
# Example 1
# Deployment client receives apps and places them into the same
# repositoryLocation (locally, relative to $SPLUNK_HOME) as it picked
them
# up from. This is typically $SPLUNK_HOME/etc/apps.  There
# is nothing in [deployment-client] because the deployment client is
not
# overriding the value set on the deployment server side.

[deployment-client]

[target-broker:deploymentServer]
targetUri= deploymentserver.splunk.mycompany.com:8089


# Example 2
# Deployment server keeps apps to be deployed in a non-standard location
on
```

```
# the server side (perhaps for organization purposes).
# Deployment client receives apps and places them in the standard
location.
# Note: Apps deployed to any location other than
# $SPLUNK_HOME/etc/apps on the deployment client side will
# not be recognized and run.
# This configuration rejects any location specified by the deployment
server
# and replaces it with the standard client-side location.

[deployment-client]
serverRepositoryLocationPolicy = rejectAlways
repositoryLocation = $SPLUNK_HOME/etc/apps

[target-broker:deploymentServer]
targetUri= deploymentserver.splunk.mycompany.com:8089


# Example 3
# Deployment client should get apps from an HTTP server that is
different
# from the one specified by the deployment server.

[deployment-client]
serverEndpointPolicy = rejectAlways
endpoint =
http://apache.mycompany.server:8080/$serverClassName$/$appName$.tar

[target-broker:deploymentServer]
targetUri= deploymentserver.splunk.mycompany.com:8089


# Example 4
# Deployment client should get apps from a location on the file system
and
# not from a location specified by the deployment server

[deployment-client]
serverEndpointPolicy = rejectAlways
endpoint = file:/<some_mount_point>/$serverClassName$/$appName$.tar

[target-broker:deploymentServer]
targetUri= deploymentserver.splunk.mycompany.com:8089
handshakeRetryIntervalInSecs=20

# Example 5
# Deployment client should phonehome server for app updates quicker
# Deployment client should only send back appEvents once a day

[deployment-client]
phoneHomeIntervalInSecs=30
appEventsResyncIntervalInSecs=86400
```

```
[target-broker:deploymentServer]
targetUri= deploymentserver.splunk.mycompany.com:8089
```

distsearch.conf

The following are the spec and example files for distsearch.conf.

distsearch.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attributes and values you can use to
configure
# distributed search.
#
# To set custom configurations, place a distsearch.conf in
# $SPLUNK_HOME/etc/system/local/.  For examples, see
distsearch.conf.example.
# You must restart Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
#
# These attributes are all configured on the search head, with the
exception of
# the optional attributes listed under the SEARCH HEAD BUNDLE MOUNTING
OPTIONS
# heading, which are configured on the search peers.
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
top of
#     the file.
#   * Each conf file should have at most one default stanza. If there
are
#     multiple default stanzas, attributes are combined. In the case of
#     multiple definitions of the same attribute, the last definition in
```

```
the
#     file wins.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.

[distributedSearch]
* Set distributed search configuration options under this stanza name.
* Follow this stanza name with any number of the following
attribute/value
  pairs.
* If you do not set any attribute, Splunk uses the default value (if
there
  is one listed).

disabled = [true|false]
* Toggle distributed search off (true) and on (false).
* Defaults to false (your distributed search stanza is enabled by
default).

heartbeatMcastAddr = <IP address>
* This setting is deprecated

heartbeatPort = <port>
* This setting is deprecated

ttl = <integer>
* This setting is deprecated

heartbeatFrequency = <int, in seconds>
* This setting is deprecated

statusTimeout = <int, in seconds>
* Set connection timeout when gathering a search peer's basic
  info (/services/server/info).
* Note: Read/write timeouts are automatically set to twice this value.
* Defaults to 10.

removedTimedOutServers = [true|false]
* This setting is no longer supported, and will be ignored.

checkTimedOutServersFrequency = <integer, in seconds>
* This setting is no longer supported, and will be ignored.

autoAddServers = [true|false]
* This setting is deprecated

bestEffortSearch = [true|false]
* Whether to remove a peer from search when it does not have any of our
  bundles.
* If set to true searches will never block on bundle replication, even
when a
```

```
  peer is first added - the peers that don't have any common bundles
will
  simply not be searched.
* Defaults to false

skipOurselves = [true|false]
* This setting is deprecated

servers = <comma separated list of servers>
* Initial list of servers.
* Each member of this list must be a valid uri in the format of
scheme://hostname:port

disabled_servers = <comma separated list of servers>
* A list of disabled search peers. Peers in this list are not monitored
or searched.
* Each member of this list must be a valid uri in the format of
scheme://hostname:port

quarantined_servers = <comma separated list of servers>
* A list of quarantined search peers.
* Each member of this list must be a valid uri in the format of
scheme://hostname:port
* The admin may quarantine peers that seem unhealthy and are degrading
search
  performancce of the whole deployment.
* Quarantined peers are monitored but not searched by default.
* A user may use the splunk_server arguments to target a search to
qurantined peers
  at the risk of slowing the search.
* When a peer is quarantined, running realtime searches will NOT be
restarted. Running
  realtime searches will continue to return results from the quarantined
peers. Any
  realtime searches started after the peer has been quarantined will not
contact the peer.
* Whenever a quarantined peer is excluded from search, appropriate
warnings will be displayed
  in the search.log and Job Inspector

shareBundles = [true|false]
* Indicates whether this server will use bundle replication to share
search
  time configuration with search peers.
* If set to false, the search head assumes that all the search peers
can access
  the correct bundles via share storage and have configured the options
listed
  under "SEARCH HEAD BUNDLE MOUNTING OPTIONS".
* Defaults to true.

useSHPBundleReplication = <bool>|always
```

```
* Relevant only in search head pooling environments. Whether the search
heads
  in the pool should compete with each other to decide which one should
handle
  the bundle replication (every time bundle replication needs to happen)
or
  whether each of them should individually replicate the bundles.
* When set to always and bundle mounting is being used then use the
search head
  pool guid rather than  each individual server name to identify
bundles (and
  search heads to the remote peers).
* Defaults to true

trySSLFirst = <bool>
* This setting is no longer supported, and will be ignored.

peerResolutionThreads = <int>
* This setting is no longer supported, and will be ignored.

defaultUriScheme = [http|https]
* When a new peer is added without specifying a scheme for the uri to
its management
  port we will use this scheme by default.
* Defaults to https

serverTimeout = <int, in seconds>
* REMOVED, this setting is now ignored and has been replaced by
  connectionTimeout, sendTimeout, receiveTimeout

connectionTimeout = <int, in seconds>
* Amount of time in seconds to use as a timeout during search peer
connection
  establishment.

sendTimeout = <int, in seconds>
* Amount of time in seconds to use as a timeout while trying to
write/send data
  to a search peer.

receiveTimeout = <int, in seconds>
* Amount of time in seconds to use as a timeout while trying to
read/receive
  data from a search peer.

authTokenConnectionTimeout = <number, in seconds>
* Maximum number of seconds to connect to a remote search peer, when
getting
  its auth token
* Fractional seconds are allowed
* Default is 5
```

```
authTokenSendTimeout = <number, in seconds>
* Maximum number of seconds to send a request to the remote peer, when
getting
  its auth token
* Fractional seconds are allowed
* Default is 10

authTokenReceiveTimeout = <number, in seconds>
* Maximum number of seconds to receive a response from a remote peer,
when
  getting its auth token
* Fractional seconds are allowed
* Default is 10

#*******************************************************************************
```

DISTRIBUTED SEARCH KEY PAIR GENERATION OPTIONS

```
# DISTRIBUTED SEARCH KEY PAIR GENERATION OPTIONS
#*******************************************************************************

[tokenExchKeys]

certDir = <directory>
* This directory contains the local Splunk instance's distributed search
key
  pair.
* This directory also contains the public keys of servers that
distribute
  searches to this Splunk instance.

publicKey = <filename>
* Name of public key file for this Splunk instance.

privateKey = <filename>
* Name of private key file for this Splunk instance.

genKeyScript = <command>
* Command used to generate the two files above.

#*******************************************************************************
```

REPLICATION SETTING OPTIONS

```
# REPLICATION SETTING OPTIONS
#*******************************************************************************

[replicationSettings]
```

```
connectionTimeout = <int, in seconds>
* The maximum number of seconds to wait before timing out on initial
connection
  to a peer.

sendRcvTimeout = <int, in seconds>
* The maximum number of seconds to wait for the sending of a full
replication
  to a peer.

replicationThreads = <int>
* The maximum number of threads to use when performing bundle
replication to peers.
* Must be a positive number
* Defaults to 5.

maxMemoryBundleSize = <int>
* The maximum size (in MB) of bundles to hold in memory. If the bundle
is
  larger than this the bundles will be read and encoded on the fly for
each
  peer the replication is taking place.
* Defaults to 10

maxBundleSize = <int>
* The maximum size (in MB) of the bundle for which replication can
occur. If
  the bundle is larger than this  bundle replication will not occur and
an
  error message will be logged.
* Defaults to: 1024 (1GB)

concerningReplicatedFileSize = <int>
* Any individual file within a bundle that is larger than this value (in
MB)
  will trigger a splunkd.log message.
* Where possible, avoid replicating such files, e.g. by customizing your
blacklists.
* Defaults to: 50

excludeReplicatedLookupSize = <int>
* Any lookup file larger than this value (in MB) will be excluded from
the knowledge bundle that the search head replicates to its search
peers.
* When this value is set to 0, this feature is disabled.
* Defaults to 0

allowStreamUpload = auto | true | false
* Whether to enable streaming bundle replication for peers.
* If set to auto, streaming bundle replication will be used when
connecting to
```

```
  peers with a complete implementation of this feature (Splunk 6.0 or
higher).
* If set to true, streaming bundle replication will be used when
connecting to
  peers with a complete or experimental implementation of this feature
(Splunk
  4.2.3 or higher).
* If set to false, streaming bundle replication will never be used.
  Whatever the value of this setting, streaming bundle replication will
not be
  used for peers that completely lack support for this feature.
* Defaults to: auto

allowSkipEncoding = <bool>
* Whether to avoid URL-encoding bundle data on upload.
* Defaults to: true

allowDeltaUpload = <bool>
* Whether to enable delta-based bundle replication.
* Defaults to: true

sanitizeMetaFiles = <bool>
* Whether to sanitize or filter *.meta files before replication.
* This feature can be used to avoid unnecessary replications triggered
by
  writes to *.meta files that have no real effect on search behavior.
* The types of stanzas that "survive" filtering are configured via the
  replicationSettings:refineConf stanza.
* The filtering process removes comments and cosmetic whitespace.
* Defaults to: true

[replicationSettings:refineConf]

replicate.<conf_file_name> = <bool>
* Controls whether Splunk replicates a particular type of *.conf file,
along
  with any associated permissions in *.meta files.
* These settings on their own do not cause files to be replicated. A
file must
  still be whitelisted (via replicationWhitelist) to be eligible for
inclusion
  via these settings.
* In a sense, these settings constitute another level of filtering that
applies
  specifically to *.conf files and stanzas with *.meta files.
* Defaults to: false

#*******************************************************************************
```

REPLICATION WHITELIST OPTIONS

```
# REPLICATION WHITELIST OPTIONS
#*******************************************************************************

[replicationWhitelist]

<name> = <whitelist_pattern>
* Controls Splunk's search-time conf replication from search heads to
search
  nodes.
* Only files that match a whitelist entry will be replicated.
* Conversely, files which are not matched by any whitelist will not be
  replicated.
* Only files located under $SPLUNK_HOME/etc will ever be replicated in
this
  way.
  * The regex will be matched against the filename, relative to
$SPLUNK_HOME/etc.
    Example: for a file
"$SPLUNK_HOME/etc/apps/fancy_app/default/inputs.conf"
             this whitelist should match
"apps/fancy_app/default/inputs.conf"
  * Similarly, the etc/system files are available as system/...
    user-specific files are available as users/username/appname/...
* The 'name' element is generally just descriptive, with one exception:
  if <name> begins with "refine.", files whitelisted by the given
pattern will
  also go through another level of filtering configured in the
  replicationSettings:refineConf stanza.
* The whitelist_pattern is the Splunk-style pattern matching, which is
  primarily regex-based with special local behavior for '...' and '*'.
  * ... matches anything, while * matches anything besides directory
separators.
    See props.conf.spec for more detail on these.
  * Note '.' will match a literal dot, not any character.
* Note that these lists are applied globally across all conf data, not
to any
  particular app, regardless of where they are defined.  Be careful to
pull in
  only your intended files.

#*******************************************************************************
```

REPLICATION BLACKLIST OPTIONS

```
# REPLICATION BLACKLIST OPTIONS
#*******************************************************************************
```

```
[replicationBlacklist]

<name> = <blacklist_pattern>
* All comments from the replication whitelist notes above also apply
here.
* Replication blacklist takes precedence over the whitelist, meaning
that a
  file that matches both the whitelist and the blacklist will NOT be
  replicated.
* This can be used to prevent unwanted bundle replication in two common
  scenarios:
    * Very large files, which part of an app may not want to be
replicated,
      especially if they are not needed on search nodes.
    * Frequently updated files (for example, some lookups) will trigger
      retransmission of all search head data.
* Note that these lists are applied globally across all conf data.
Especially
  for blacklisting, be careful to constrain your blacklist to match only
data
  your application will not need.

#*******************************************************************************
```

BUNDLE ENFORCER WHITELIST OPTIONS

```
# BUNDLE ENFORCER WHITELIST OPTIONS
#*******************************************************************************

[bundleEnforcerWhitelist]

<name> = <whitelist_pattern>
* Peers uses this to make sure knowledge bundle sent by search heads
and
  masters do not contain alien files.
* If this stanza is empty, the receiver accepts the bundle unless it
contains
  files matching the rules specified in [bundleEnforcerBlacklist].
Hence, if
  both [bundleEnforcerWhitelist] and [bundleEnforcerBlacklist] are
empty (which
  is the default), then the receiver accepts all bundles.
* If this stanza is not empty, the receiver accepts the bundle only if
it
  contains only files that match the rules specified here but not those
in
  [bundleEnforcerBlacklist].
* All rules are regexs.
* This stanza is empty by default.
```

```
#*******************************************************************************
```

BUNDLE ENFORCER BLACKLIST OPTIONS

```
# BUNDLE ENFORCER BLACKLIST OPTIONS
#*******************************************************************************

[bundleEnforcerBlacklist]

<name> = <blacklist_pattern>
* Peers uses this to make sure knowledge bundle sent by search heads
and
  masters do not contain alien files.
* This list overrides [bundleEnforceWhitelist] above. That means the
receiver
  rejects (i.e. removes) the bundle if it contains any file that matches
the
  rules specified here even if that file is allowed by
[bundleEnforcerWhitelist].
* If this stanza is empty, then only [bundleEnforcerWhitelist] matters.
* This stanza is empty by default.

#*******************************************************************************
```

SEARCH HEAD BUNDLE MOUNTING OPTIONS

```
# SEARCH HEAD BUNDLE MOUNTING OPTIONS
# You set these attributes on the search peers only, and only if you
also set
# shareBundles=false in [distributedSearch] on the search head. Use
them to
# achieve replication-less bundle access. The search peers use a shared
storage
# mountpoint to access the search head bundles ($SPLUNK_HOME/etc).
#*******************************************************************************

[searchhead:<searchhead-splunk-server-name>]
* <searchhead-splunk-server-name> is the name of the related searchhead
  installation.
* This setting is located in server.conf, serverName = <name>

mounted_bundles = [true|false]
* Determines whether the bundles belong to the search head specified in
the
  stanza name are mounted.
* You must set this to "true" to use mounted bundles.
* Default is "false".
```

```
bundles_location = <path_to_bundles>
* The path to where the search head's bundles are mounted. This must be
the
  mountpoint on the search peer,  not on the search head. This should
point to
  a directory that is equivalent to $SPLUNK_HOME/etc/. It must contain
at least
  the following subdirectories: system, apps, users.


#*******************************************************************************
```

DISTRIBUTED SEARCH GROUP DEFINITIONS

```
# DISTRIBUTED SEARCH GROUP DEFINITIONS
# These are the definitions of the distributed search groups. A search
group is
# a set of search peers as identified by thier host:management-port. A
search
# may be directed to a search group using the splunk_server_group
argument.The
# search will be dispatched to only the members of the group.
#*******************************************************************************

[distributedSearch:<splunk-server-group-name>]
* <splunk-server-group-name> is the name of the splunk-server-group
that is
  defined in this stanza

servers = <comma separated list of servers>
* List of search peers that are members of this group. Comma serparated
list
  of peer identifiers i.e. hostname:port

default = [true|false]
* Will set this as the default group of peers against which all
searches are
  run unless a server-group is not explicitly specified.
```

distsearch.conf.example

```
#   Version 6.5.0
#
# These are example configurations for distsearch.conf. Use this file
to
# configure distributed search.  For all available attribute/value
pairs, see
```

```
# distsearch.conf.spec.
#
# There is NO DEFAULT distsearch.conf.
#
# To use one or more of these configurations, copy the configuration
block into
# distsearch.conf in $SPLUNK_HOME/etc/system/local/.  You must restart
Splunk
# to enable configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

[distributedSearch]
servers = https://192.168.1.1:8059,https://192.168.1.2:8059

# This entry distributes searches to 192.168.1.1:8059,192.168.1.2:8059.
# These machines will be contacted on port 8059 using https
# Attributes not set here will use the defaults listed in
distsearch.conf.spec.

# this stanza controls the timing settings for connecting to a remote
peer and
# the send timeout
[replicationSettings]
connectionTimeout = 10
sendRcvTimeout = 60

# this stanza controls what files are replicated to the other peer each
is a
# regex
[replicationWhitelist]
allConf = *.conf

# Mounted bundles example.
# This example shows two distsearch.conf configurations, one for the
search
# head and another for each of the search head's search peers. It shows
only
# the attributes necessary to implement mounted bundles.

# On a search head whose Splunk server name is "searcher01":
[distributedSearch]
...
shareBundles = false

# On each search peer:
[searchhead:searcher01]
mounted_bundles = true
```

```
bundles_location = /opt/shared_bundles/searcher01
```

eventdiscoverer.conf

The following are the spec and example files for eventdiscoverer.conf.

eventdiscoverer.conf.spec

```
#   Version 6.5.0

# This file contains possible attributes and values you can use to
configure
# event discovery through the search command "typelearner."
#
# There is an eventdiscoverer.conf in $SPLUNK_HOME/etc/system/default/.
 To set
# custom configurations, place an eventdiscoverer.conf in
# $SPLUNK_HOME/etc/system/local/.  For examples, see
# eventdiscoverer.conf.example. You must restart Splunk to enable
# configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
top of
#     the file.
#   * Each conf file should have at most one default stanza. If there
are
#     multiple default stanzas, attributes are combined. In the case of
#     multiple definitions of the same attribute, the last definition in
the
#     file wins.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.
```

```
ignored_keywords = <comma-separated list of terms>
* If you find that event types have terms you do not want considered
(for
  example, "mylaptopname"), add that term to this list.
* Terms in this list are never considered for defining an event type.
* For more details, refer to
$SPLUNK_HOME/etc/system/default/eventdiscoverer.conf).
* Default = "sun, mon, tue,..."

ignored_fields = <comma-separated list of fields>
* Similar to ignored_keywords, except these are fields as defined in
Splunk
  instead of terms.
* Defaults include time-related fields that would not be useful for
defining an
  event type.

important_keywords = <comma-separated list of terms>
* When there are multiple possible phrases for generating an eventtype
search,
  those phrases with important_keyword terms are favored.  For example,
  "fatal error" would be preferred over "last message repeated", as
"fatal" is
  an important keyword.
* Default = "abort, abstract, accept,..."
* For the full default setting, see
$SPLUNK_HOME/etc/system/default/eventdiscoverer.conf.
```

eventdiscoverer.conf.example

```
#   Version 6.5.0
#
# This is an example eventdiscoverer.conf.  These settings are used to
control
# the discovery of common eventtypes used by the typelearner search
command.
#
# To use one or more of these configurations, copy the configuration
block into
# eventdiscoverer.conf in $SPLUNK_HOME/etc/system/local/.  You must
restart
# Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

```
# Terms in this list are never considered for defining an eventtype.
ignored_keywords = foo, bar, application, kate, charlie

# Fields in this list are never considered for defining an eventtype.
ignored_fields = pid, others, directory
```

event_renderers.conf

The following are the spec and example files for event_renderers.conf.

event_renderers.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attribute/value pairs for configuring
event rendering properties.
#
# Beginning with version 6.0, Splunk Enterprise does not support the
# customization of event displays using event renderers.
#
# There is an event_renderers.conf in $SPLUNK_HOME/etc/system/default/.
 To set custom configurations,
# place an event_renderers.conf in $SPLUNK_HOME/etc/system/local/, or
your own custom app directory.
#
# To learn more about configuration files (including precedence) please
see the documentation
# located at
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#     * You can also define global settings outside of any stanza, at
the top of the file.
#     * Each conf file should have at most one default stanza. If there
are multiple default
#       stanzas, attributes are combined. In the case of multiple
definitions of the same
#       attribute, the last definition in the file wins.
#     * If an attribute is defined at both the global level and in a
specific stanza, the
```

```
#       value in the specific stanza takes precedence.
```

[<name>]

```
[<name>]
* Stanza name. This name must be unique.

eventtype = <event type>
* Specify event type name from eventtypes.conf.

priority = <positive integer>
* Highest number wins!!

template = <valid Mako template>
* Any template from the $APP/appserver/event_renderers directory.

css_class = <css class name suffix to apply to the parent event element
class attribute>
* This can be any valid css class value.
* The value is appended to a standard suffix string of "splEvent-". A
css_class value of foo would
result in the parent element of the event having an html attribute
class with a value of splEvent-foo
(for example, class="splEvent-foo"). You can externalize your css style
rules for this in
$APP/appserver/static/application.css. For example, to make the text
red you would add to
application.css:.splEvent-foo { color:red; }
```

event_renderers.conf.example

```
#   Version 6.5.0
# DO NOT EDIT THIS FILE!
# Please make all changes to files in $SPLUNK_HOME/etc/system/local.
# To make changes, copy the section/stanza you want to change from
$SPLUNK_HOME/etc/system/default
# into ../local and edit there.
#
# This file contains mappings between Splunk eventtypes and event
renderers.
#
# Beginning with version 6.0, Splunk Enterprise does not support the
# customization of event displays using event renderers.
#

[event_renderer_1]
eventtype = hawaiian_type
```

```
priority = 1
css_class = EventRenderer1

[event_renderer_2]
eventtype = french_food_type
priority = 1
template = event_renderer2.html
css_class = EventRenderer2

[event_renderer_3]
eventtype = japan_type
priority = 1
css_class = EventRenderer3
```

eventtypes.conf

The following are the spec and example files for eventtypes.conf.

eventtypes.conf.spec

```
#   Version 6.5.0
#
# This file contains all possible attributes and value pairs for an
# eventtypes.conf file.  Use this file to configure event types and
their
# properties. You can also pipe any search to the "typelearner" command
to
# create event types.  Event types created this way will be written to
# $SPLUNK_HOME/etc/system/local/eventtypes.conf.
#
# There is an eventtypes.conf in $SPLUNK_HOME/etc/system/default/.  To
set
# custom configurations, place an eventtypes.conf in
# $SPLUNK_HOME/etc/system/local/. For examples, see
eventtypes.conf.example.
# You must restart Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#  * You can also define global settings outside of any stanza, at the
top
#    of the file.
#  * Each conf file should have at most one default stanza. If there
are
#    multiple default stanzas, attributes are combined. In the case of
#    multiple definitions of the same attribute, the last definition in
the
#    file wins.
#  * If an attribute is defined at both the global level and in a
specific
#    stanza, the value in the specific stanza takes precedence.
```

[<$EVENTTYPE>]

```
[<$EVENTTYPE>]
* Header for the event type
* $EVENTTYPE is the name of your event type.
* You can have any number of event types, each represented by a stanza
and
  any number of the following attribute/value pairs.
* NOTE: If the name of the event type includes field names surrounded by
the
  percent character (for example "%$FIELD%") then the value of $FIELD is
  substituted into the event type name for that event.  For example, an
  event type with the header [cisco-%code%] that has "code=432" becomes
  labeled "cisco-432".

disabled = [1|0]
* Toggle event type on or off.
* Set to 1 to disable.

search = <string>
* Search terms for this event type.
* For example: error OR warn.
* NOTE: You cannot base an event type on:
* A search that includes a pipe operator (a "|" character).
* A subsearch (a search pipeline enclosed in square brackets).
* A search referencing a report. This is a best practice. Any report
that is referenced by an
  event type can later be updated in a way that makes it invalid as an
event type. For example,
  a report that is updated to include transforming commands cannot be
used as the definition for
  an event type. You have more control over your event type if you
```

```
define it with the same search
  string as the report.

priority = <integer, 1 through 10>
* Value used to determine the order in which the matching eventtypes of
an
  event are displayed.
* 1 is the highest priority and 10 is the lowest priority.

description = <string>
* Optional human-readable description of this saved search.

tags = <string>
* DEPRECATED - see tags.conf.spec

color = <string>
* color for this event type.
* Supported colors: none, et_blue, et_green, et_magenta, et_orange,
  et_purple, et_red, et_sky, et_teal, et_yellow
```

eventtypes.conf.example

```
#   Version 6.5.0
#
# This file contains an example eventtypes.conf.  Use this file to
configure custom eventtypes.
#
# To use one or more of these configurations, copy the configuration
block into eventtypes.conf
# in $SPLUNK_HOME/etc/system/local/. You must restart Splunk to enable
configurations.
#
# To learn more about configuration files (including precedence) please
see the documentation
# located at
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
#

# The following example makes an eventtype called "error" based on the
search "error OR fatal."

[error]
search = error OR fatal



# The following example makes an eventtype template because it includes
a field name
```

```
# surrounded by the percent character (in this case "%code%").
# The value of "%code%" is substituted into the event type name for that
event.
# For example, if the following example event type is instantiated on an
event that has a
# "code=432," it becomes "cisco-432".

[cisco-%code%]
search = cisco
```

fields.conf

The following are the spec and example files for fields.conf.

fields.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attribute and value pairs for:
#  * Telling Splunk how to handle multi-value fields.
#  * Distinguishing indexed and extracted fields.
#  * Improving search performance by telling the search processor how to
#    handle field values.

# Use this file if you are creating a field at index time (not
advised).
#
# There is a fields.conf in $SPLUNK_HOME/etc/system/default/.  To set
custom
# configurations, place a fields.conf in
$SPLUNK_HOME/etc/system/local/.  For
# examples, see fields.conf.example.  You must restart Splunk to enable
# configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
```

```
#   * You can also define global settings outside of any stanza, at the
top of
#     the file.
#   * Each conf file should have at most one default stanza. If there
are
#     multiple default stanzas, attributes are combined. In the case of
#     multiple definitions of the same attribute, the last definition in
the
#     file wins.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.
```

[<field name>]

```
[<field name>]
* Name of the field you're configuring.
* Follow this stanza name with any number of the following
attribute/value
  pairs.
* Field names can only contain a-z, A-Z, 0-9, and  _, but cannot begin
with a
  number or _

# TOKENIZER indicates that your configured field's value is a smaller
part of a
# token.  For example, your field's value is "123" but it occurs as
"foo123" in
# your event.
TOKENIZER = <regular expression>
* Use this setting to configure multivalue fields (refer to the online
  documentation for multivalue fields).
* A regular expression that indicates how the field can take on multiple
values
  at the same time.
* If empty, the field can only take on a single value.
* Otherwise, the first group is taken from each match to form the set of
  values.
* This setting is used by the "search" and "where" commands, the summary
and
  XML outputs of the asynchronous search API, and by the top, timeline
and
  stats commands.
* Tokenization of indexed fields (INDEXED = true) is not supported so
this
  attribute is ignored for indexed fields.
* Default to empty.

INDEXED = [true|false]
* Indicate whether a field is indexed or not.
```

```
* Set to true if the field is indexed.
* Set to false for fields extracted at search time (the majority of
fields).
* Defaults to false.

INDEXED_VALUE = [true|false|<sed-cmd>|<simple-substitution-string>]
* Set this to true if the value is in the raw text of the event.
* Set this to false if the value is not in the raw text of the event.
* Setting this to true expands any search for key=value into a search of
  value AND key=value (since value is indexed).
* For advanced customization, this setting supports sed style
substitution.
  For example, 'INDEXED_VALUE=s/foo/bar/g' would take the value of the
field,
  replace all instances of 'foo' with 'bar,' and use that new value as
the
  value to search in the index.
* This setting also supports a simple substitution based on looking for
the
  literal string '<VALUE>' (including the '<' and '>' characters).
  For example, 'INDEXED_VALUE=source::*<VALUE>*' would take a search
for
  'myfield=myvalue' and search for 'source::*myvalue*' in the index as
a
  single term.
* For both substitution constructs, if the resulting string starts with
a '[',
  Splunk interprets the string as a Splunk LISPY expression.  For
example,
  'INDEXED_VALUE=[OR <VALUE> source::*<VALUE>]' would turn
'myfield=myvalue'
  into applying the LISPY expression '[OR myvalue source::*myvalue]'
(meaning
  it matches either 'myvalue' or 'source::*myvalue' terms).
* Defaults to true.
* NOTE: You only need to set indexed_value if indexed = false.
```

fields.conf.example

```
#   Version 6.5.0
#
# This file contains an example fields.conf.  Use this file to configure
# dynamic field extractions.
#
# To use one or more of these configurations, copy the configuration
block into
# fields.conf in $SPLUNK_HOME/etc/system/local/. You must restart
Splunk to
# enable configurations.
```

```
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
#
# These tokenizers result in the values of To, From and Cc treated as a
list,
# where each list element is an email address found in the raw string of
data.

[To]
TOKENIZER = (\w[\w\.\-]*@[\w\.\-]*\w)

[From]
TOKENIZER = (\w[\w\.\-]*@[\w\.\-]*\w)

[Cc]
TOKENIZER = (\w[\w\.\-]*@[\w\.\-]*\w)
```

indexes.conf

The following are the spec and example files for indexes.conf.

indexes.conf.spec

```
#   Version 6.5.0
#
# This file contains all possible options for an indexes.conf file.
 Use
# this file to configure Splunk's indexes and their properties.
#
# There is an indexes.conf in $SPLUNK_HOME/etc/system/default/.  To set
# custom configurations, place an indexes.conf in
# $SPLUNK_HOME/etc/system/local/. For examples, see
indexes.conf.example.
# You must restart Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
#
# CAUTION:  You can drastically affect your Splunk installation by
```

```
changing
# these settings.  Consult technical support
# (http://www.splunk.com/page/submit_issue) if you are not sure how to
# configure this file.
#
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
top
#     of the file.
#   * Each conf file should have at most one default stanza. If there
are
#     multiple default stanzas, attributes are combined. In the case of
#     multiple definitions of the same attribute, the last definition in
the
#     file wins.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.

sync = <nonnegative integer>
* The index processor syncs events every <integer> number of events.
* Set to 0 to disable.
* Highest legal value is 32767
* Defaults to 0.

defaultDatabase = <index name>
* If no index is specified during search, Splunk searches the default
index.
* The specified index displays as the default in Splunk Manager
settings.
* Defaults to "main".

queryLanguageDefinition = <path to file>
* DO NOT EDIT THIS SETTING. SERIOUSLY.
* The path to the search language definition file.
* Defaults to $SPLUNK_HOME/etc/searchLanguage.xml.

lastChanceIndex = <index name>
* Gives ability to define a last chance index for events destined for
  non-existent indexes.
* If an event arrives whose index destination key points to an index
that is
  not configured (such as when using index=<index name> in the input
stanza or
  by a setting in a transform), it will route that event to the index
specified
```

```
  by this setting.  The index destination key of that event will be
overwritten
  with the specified index name before routing.
* <index name> must name an existing enabled index.  Splunk will not
start if
  this is not the case.
* If this setting is not defined or is empty, it will drop such events.
* If set to "default", then the default index specified by the
  "defaultDatabase" will be used as a last chance index.
* Defaults to empty.

memPoolMB = <positive integer>|auto
* Determines how much memory is given to the indexer memory pool. This
  restricts the number of outstanding events in the indexer at any given
  time.
* Must be greater than 0; maximum value is 1048576 (which corresponds to
1 TB)
* Setting this too high can lead to splunkd memory usage going up
  substantially.
* Setting this too low can degrade splunkd indexing performance.
* Setting this to "auto" or an invalid value will cause Splunk to
autotune
  this parameter.
* Defaults to "auto".
  * The values derived when "auto" is seen are as follows:
    * System Memory Available less than ... | memPoolMB
                  1 GB                      |    64  MB
                  2 GB                      |    128 MB
                  8 GB                      |    128 MB
                  8 GB or higher            |    512 MB
* Only set this value if you are an expert user or have been advised to
by
  Splunk Support.
* CARELESSNESS IN SETTING THIS MAY LEAD TO PERMANENT BRAIN DAMAGE OR
  LOSS OF JOB.

indexThreads = <nonnegative integer>|auto
* Determines the number of threads to use for indexing.
* Must be at least 1 and no more than 16.
* This value should not be set higher than the number of processor
cores in
  the box.
* If splunkd is also doing parsing and aggregation, the number should be
set
  lower than the total number of processors minus two.
* Setting this to "auto" or an invalid value will cause Splunk to
autotune
  this parameter.
* Only set this value if you are an expert user or have been advised to
by
  Splunk Support.
* CARELESSNESS IN SETTING THIS MAY LEAD TO PERMANENT BRAIN DAMAGE OR
```

```
  LOSS OF JOB.
* Defaults to "auto".

rtRouterThreads = 0|1
* Set this to 1 if you expect to use non-indexed real time searches
regularly.  Index
  throughput drops rapidly if there are a handful of these running
concurrently on the system.
* If you are not sure what "indexed vs non-indexed" real time searches
are, see
  README of indexed_realtime* settings in limits.conf
* NOTE: This is not a boolean value, only 0 or 1 is accepted.  In the
future, we
  may allow more than a single thread, but current implementation
  only allows one to create a single thread per pipeline set

rtRouterQueueSize = <positive integer>
* Defaults to 10000
* This setting is only relevant if rtRouterThreads != 0
* This queue sits between the indexer pipeline set thread (producer) and
the rtRouterThread
* Changing the size of this queue may impact real time search
performance

assureUTF8 = true|false
* Verifies that all data retrieved from the index is proper by
validating
  all the byte strings.
  * This does not ensure all data will be emitted, but can be a
workaround
    if an index is corrupted in such a way that the text inside it is
no
    longer valid utf8.
* Will degrade indexing performance when enabled (set to true).
* Can only be set globally, by specifying in the [default] stanza.
* Defaults to false.

enableRealtimeSearch = true|false
* Enables real-time searches.
* Defaults to true.

suppressBannerList = <comma-separated list of strings>
* suppresses index missing warning banner messages for specified indexes
* Defaults to empty

maxRunningProcessGroups = <positive integer>
* splunkd fires off helper child processes like splunk-optimize,
  recover-metadata, etc.  This param limits how many child processes
can be
  running at any given time.
* This maximum applies to entire splunkd, not per index.  If you have N
  indexes, there will be at most maxRunningProcessGroups child
```

```
processes,
  not N*maxRunningProcessGroups
* Must maintain maxRunningProcessGroupsLowPriority <
maxRunningProcessGroups
* This is an advanced parameter; do NOT set unless instructed by Splunk
  Support
* Highest legal value is 4294967295
* Defaults to 8 (note: up until 5.0 it defaulted to 20)

maxRunningProcessGroupsLowPriority = <positive integer>
* Of the maxRunningProcessGroups (q.v.) helper child processes, at most
  maxRunningProcessGroupsLowPriority may be low-priority (e.g. fsck)
ones.
* This maximum applies to entire splunkd, not per index.  If you have N
  indexes, there will be at most maxRunningProcessGroupsLowPriority
  low-priority child processes, not
N*maxRunningProcessGroupsLowPriority
* Must maintain maxRunningProcessGroupsLowPriority <
maxRunningProcessGroups
* This is an advanced parameter; do NOT set unless instructed by Splunk
  Support
* Highest legal value is 4294967295
* Defaults to 1

bucketRebuildMemoryHint = <positive integer>[KB|MB|GB]|auto
* Suggestion for the bucket rebuild process for the size (bytes) of
tsidx
  file it will try to build.
* Larger files use more memory in rebuild, but rebuild will fail if
there is
  not enough.
* Smaller files make the rebuild take longer during the final optimize
step.
* Note: this value is not a hard limit on either rebuild memory usage
or
  tsidx size.
* This is an advanced parameter, do NOT set this unless instructed by
Splunk
  Support.
* Defaults to "auto", which varies by the amount of physical RAM on the
host
  *  less than 2GB RAM = 67108864 (64MB) tsidx
  *  2GB to 8GB RAM = 134217728 (128MB) tsidx
  *  more than 8GB RAM = 268435456 (256MB) tsidx
* If not "auto", then must be 16MB-1GB.
* Value may be specified using a size suffix: "16777216" or "16MB" are
  equivalent.
* Inappropriate use of this parameter will cause splunkd to not start
if
  rebuild is required.
* Highest legal value (in bytes) is 4294967295
```

```
inPlaceUpdates = true|false
* If true, metadata updates are written to the .data files directly
* If false, metadata updates are written to a temporary file and then
moved
  into place
* Intended for advanced debugging of metadata issues
* Setting this parameter to false (to use a temporary file) will impact
  indexing performance, particularly with large numbers of hosts,
sources,
  or sourcetypes (~1 million, across all indexes.)
* This is an advanced parameter; do NOT set unless instructed by Splunk
  Support
* Defaults to true

serviceOnlyAsNeeded = true|false
* Causes index service (housekeeping tasks) overhead to be incurred only
  after index activity.
* Indexer module problems may be easier to diagnose when this
optimization
  is disabled (set to false).
* Defaults to true.

serviceSubtaskTimingPeriod = <positive integer>
* Subtasks of indexer service task will be timed on every Nth execution,
  where N = value of this parameter, in seconds.
* Smaller values will give greater accuracy; larger values will lessen
timer
  overhead.
* Timer measurements will be found in metrics.log, marked
  "group=subtask_seconds, task=indexer_service"
* Highest legal value is 4294967295
* We strongly suggest value of this parameter divide evenly into value
of
  'rotatePeriodInSecs' parameter.
* Defaults to 30

processTrackerServiceInterval = <nonnegative integer>
* Controls how often, in seconds, indexer checks status of the child OS
  processes it had launched to see if it can launch new processes for
queued
  requests.
* If set to 0, indexer will check child process status every second.
* Highest legal value is 4294967295
* Defaults to 15

maxBucketSizeCacheEntries = <nonnegative integer>
* This value is not longer needed and its value is ignored.

tsidxStatsHomePath = <path on server>
* An absolute path that specifies where Splunk creates namespace data
with
  'tscollect' command
```

```
* If the directory does not exist, we attempt to create it.
* Optional. If this is unspecified, we default to the 'tsidxstats'
directory
  under $SPLUNK_DB

hotBucketTimeRefreshInterval = <positive integer>
* Controls how often each index refreshes the available hot bucket times
  used by the indexes REST endpoint.
* Refresh will occur every N times service is performed for each index.
  * For busy indexes, this is a multiple of seconds.
  * For idle indexes, this is a multiple of the second-long-periods in
    which data is received.
* This tunable is only intended to relax the frequency of these
refreshes in
* the unexpected case that it adversely affects performance in unusual
  production scenarios.
* This time is tracked on a per-index basis, and thus can be adjusted
  on a per-index basis if needed.
* If, for some reason, you want have the index information refreshed
with
  every service (and accept minor performance overhead), you can use the
  value 1.
* Defaults to 10 (services).

#*******************************************************************************
```

PER INDEX OPTIONS

```
# PER INDEX OPTIONS
# These options may be set under an [<index>] entry.
#
# Index names must consist of only numbers, letters, periods,
underscores,
# and hyphens.
#*******************************************************************************

disabled = true|false
* Toggles your index entry off and on.
* Set to true to disable an index.
* Defaults to false.

deleted = true
* If present, means that this index has been marked for deletion: if
splunkd
  is running, deletion is in progress; if splunkd is stopped, deletion
will
  re-commence on startup.
* Normally absent, hence no default.
* Do NOT manually set, clear, or modify value of this parameter.
* Seriously: LEAVE THIS PARAMETER ALONE.
```

```
homePath = <path on index server>
* An absolute path that contains the hotdb and warmdb for the index.
* Splunkd keeps a file handle open for warmdbs at all times.
* May contain a volume reference (see volume section below).
* CAUTION: Path MUST be writable.
* Required. Splunk will not start if an index lacks a valid homePath.
* Must restart splunkd after changing this parameter; index reload will
not
  suffice.

coldPath = <path on index server>
* An absolute path that contains the colddbs for the index.
* Cold databases are opened as needed when searching.
* May contain a volume reference (see volume section below).
* CAUTION: Path MUST be writable.
* Required. Splunk will not start if an index lacks a valid coldPath.
* Must restart splunkd after changing this parameter; index reload will
not
  suffice.

thawedPath = <path on index server>
* An absolute path that contains the thawed (resurrected) databases for
the
  index.
* May NOT contain a volume reference.
* Required. Splunk will not start if an index lacks a valid thawedPath.
* Must restart splunkd after changing this parameter; index reload will
not
  suffice.

bloomHomePath = <path on index server>
* Location where the bloomfilter files for the index are stored.
* If specified, MUST be defined in terms of a volume definition (see
volume
  section below)
* If bloomHomePath is not specified, bloomfilter files for index will be
  stored inline, inside bucket directories.
* CAUTION: Path must be writable.
* Must restart splunkd after changing this parameter; index reload will
not
  suffice.

createBloomfilter = true|false
* Controls whether to create bloomfilter files for the index.
* TRUE: bloomfilter files will be created. FALSE: not created.
* Defaults to true.

summaryHomePath = <path on index server>
* An absolute path where transparent summarization results for data in
this
  index should be stored. Must be different for each index and may be on
```

```
any
  disk drive.
* May contain a volume reference (see volume section below).
* Volume reference must be used if data retention based on data size is
  desired.
* If not specified, Splunk will use a directory 'summary' in the same
  location as homePath
  * For example, if homePath is "/opt/splunk/var/lib/splunk/index1/db",
    then summaryHomePath would be
"/opt/splunk/var/lib/splunk/index1/summary".
* CAUTION: Path must be writable.
* Must restart splunkd after changing this parameter; index reload will
not
  suffice.
* Defaults to unset.

tstatsHomePath = <path on index server>
* Required.
* Location where datamodel acceleration TSIDX data for this index should
be
  stored
* MUST be defined in terms of a volume definition (see volume section
below)
* Must restart splunkd after changing this parameter; index reload will
not
  suffice.
* CAUTION: Path must be writable.
* Defaults to volume:_splunk_summaries/$_index_name/datamodel_summary,
  where $_index_name is runtime-expanded to the name of the index

maxBloomBackfillBucketAge = <nonnegative integer>[smhd]|infinite
* If a (warm or cold) bloomfilter-less bucket is older than this,
Splunk
  will not create a bloomfilter for that bucket.
* When set to 0, bloomfilters are never backfilled
* When set to "infinite", bloomfilters are always backfilled
* NB that if createBloomfilter=false, bloomfilters are never backfilled
  regardless of the value of this parameter
* Highest legal value in computed seconds is 2 billion, or 2000000000,
which
  is approximately 68 years.
* Defaults to 30d.

enableOnlineBucketRepair = true|false
* Controls asynchronous "online fsck" bucket repair, which runs
concurrently
  with Splunk
* When enabled, you do not have to wait until buckets are repaired, to
start
  Splunk
* When enabled, you might observe a slight performance degradation
* Defaults to true.
```

```
enableDataIntegrityControl = true|false
* If set to true, hashes are computed on the rawdata slices and stored
for
  future data integrity checks
* If set to false, no hashes are computed on the rawdata slices
* It has a global default value of false

# The following options can be set either per index or globally (as
defaults
# for all indexes).  Defaults set globally are overridden if set on a
# per-index basis.

maxWarmDBCount = <nonnegative integer>
* The maximum number of warm buckets.
* Warm buckets are located in the <homePath> for the index.
* If set to zero, Splunk will not retain any warm buckets
  (will roll them to cold as soon as it can)
* Highest legal value is 4294967295
* Defaults to 300.

maxTotalDataSizeMB = <nonnegative integer>
* The maximum size of an index (in MB).
* If an index grows larger than the maximum size, the oldest data is
frozen.
* This parameter only applies to hot, warm, and cold buckets.  It does
not
  apply to thawed buckets.
* Highest legal value is 4294967295
* Defaults to 500000.

rotatePeriodInSecs = <positive integer>
* Controls the service period (in seconds): how often splunkd performs
  certain housekeeping tasks.  Among these tasks are:
  * Check if a new hotdb needs to be created.
  * Check if there are any cold DBs that should be frozen.
  * Check whether buckets need to be moved out of hot and cold DBs, due
to
    respective size constraints (i.e., homePath.maxDataSizeMB and
    coldPath.maxDataSizeMB)
* This value becomes the default value of the rotatePeriodInSecs
attribute
  for all volumes (see rotatePeriodInSecs in the Volumes section)
* Highest legal value is 4294967295
* Defaults to 60.

frozenTimePeriodInSecs = <nonnegative integer>
* Number of seconds after which indexed data rolls to frozen.
* If you do not specify a coldToFrozenScript, data is deleted when
rolled to
  frozen.
* IMPORTANT: Every event in the DB must be older than
```

```
frozenTimePeriodInSecs
  before it will roll. Then, the DB will be frozen the next time splunkd
  checks (based on rotatePeriodInSecs attribute).
* Highest legal value is 4294967295
* Defaults to 188697600 (6 years).

warmToColdScript = <script path>
* Specifies a script to run when moving data from warm to cold.
* This attribute is supported for backwards compatibility with versions
  older than 4.0.  Migrating data across filesystems is now handled
natively
  by splunkd.
* If you specify a script here, the script becomes responsible for
moving
  the event data, and Splunk-native data migration will not be used.
* The script must accept two arguments:
  * First: the warm directory (bucket) to be rolled to cold.
  * Second: the destination in the cold path.
* Searches and other activities are paused while the script is running.
* Contact Splunk Support (http://www.splunk.com/page/submit_issue) if
you
  need help configuring this setting.
* The script must be in $SPLUNK_HOME/bin or a subdirectory thereof.
* Defaults to empty.

coldToFrozenScript = [path to script interpreter] <path to script>
* Specifies a script to run when data will leave the splunk index
system.
  * Essentially, this implements any archival tasks before the data is
    deleted out of its default location.
* Add "$DIR" (quotes included) to this setting on Windows (see below
  for details).
* Script Requirements:
  * The script must accept one argument:
    * An absolute path to the bucket directory to archive.
  * Your script should work reliably.
    * If your script returns success (0), Splunk will complete deleting
      the directory from the managed index location.
    * If your script return failure (non-zero), Splunk will leave the
bucket
      in the index, and try calling your script again several minutes
later.
    * If your script continues to return failure, this will eventually
cause
      the index to grow to maximum configured size, or fill the disk.
  * Your script should complete in a reasonable amount of time.
    * If the script stalls indefinitely, it will occupy slots.
    * This script should not run for long as it would occupy
      resources which will affect indexing.
* If the string $DIR is present in this setting, it will be expanded to
the
  absolute path to the directory.
```

```
* If $DIR is not present, the directory will be added to the end of the
  invocation line of the script.
  * This is important for Windows.
    * For historical reasons, the entire string is broken up by
      shell-pattern expansion rules.
    * Since windows paths frequently include spaces, and the windows
shell
      breaks on space, the quotes are needed for the script to
understand
      the directory.
* If your script can be run directly on your platform, you can specify
just
  the script.
  * Examples of this are:
    * .bat and .cmd files on Windows
    * scripts set executable on UNIX with a #! shebang line pointing to
a
      valid interpreter.
* You can also specify an explicit path to an interpreter and the
script.
    * Example:  /path/to/my/installation/of/python.exe
path/to/my/script.py

* Splunk ships with an example archiving script in that you SHOULD NOT
USE
  $SPLUNK_HOME/bin called coldToFrozenExample.py
  * DO NOT USE the example for production use, because:
    * 1 - It will be overwritten on upgrade.
    * 2 - You should be implementing whatever requirements you need in a
          script of your creation.  If you have no such requirements,
use
          coldToFrozenDir
* Example configuration:
  * If you create a script in bin/ called our_archival_script.py, you
could use:
    UNIX:
        coldToFrozenScript = "$SPLUNK_HOME/bin/python"
"$SPLUNK_HOME/bin/our_archival_script.py"
    Windows:
        coldToFrozenScript = "$SPLUNK_HOME/bin/python"
"$SPLUNK_HOME/bin/our_archival_script.py" "$DIR"
* The example script handles data created by different versions of
splunk
  differently. Specifically data from before 4.2 and after are handled
  differently. See "Freezing and Thawing" below:
* The script must be in $SPLUNK_HOME/bin or a subdirectory thereof.

coldToFrozenDir = <path to frozen archive>
* An alternative to a coldToFrozen script - simply specify a destination
  path for the frozen archive
* Splunk will automatically put frozen buckets in this directory
* For information on how buckets created by different versions are
```

```
  handled, see "Freezing and Thawing" below.
* If both coldToFrozenDir and coldToFrozenScript are specified,
  coldToFrozenDir will take precedence
* Must restart splunkd after changing this parameter; index reload will
not
  suffice.
* May NOT contain a volume reference.

# Freezing and Thawing (this should move to web docs
4.2 and later data:
  * To archive: remove files except for the rawdata directory, since
rawdata
    contains all the facts in the bucket.
  * To restore: run splunk rebuild <bucket_dir> on the archived bucket,
then
    atomically move the bucket to thawed for that index
4.1 and earlier data:
  * To archive: gzip the .tsidx files, as they are highly compressible
but
    cannot be recreated
  * To restore: unpack the tsidx files within the bucket, then
atomically
    move the bucket to thawed for that index

compressRawdata = true|false
* This parameter is ignored. The splunkd process always compresses raw
data.

maxConcurrentOptimizes = <nonnegative integer>
* The number of concurrent optimize processes that can run against the
hot
  DB.
* This number should be increased if:
  * There are always many small tsidx files in the hot DB.
  * After rolling, there are many tsidx files in warm or cold DB.
* Must restart splunkd after changing this parameter; index reload will
not
  suffice.
* Highest legal value is 4294967295
* Defaults to 6

maxDataSize = <positive integer>|auto|auto_high_volume
* The maximum size in MB for a hot DB to reach before a roll to warm is
  triggered.
* Specifying "auto" or "auto_high_volume" will cause Splunk to autotune
this
  parameter (recommended).
* You should use "auto_high_volume" for high-volume indexes (such as
the
  main index); otherwise, use "auto".  A "high volume index" would
typically
  be considered one that gets over 10GB of data per day.
```

```
* Defaults to "auto", which sets the size to 750MB.
* "auto_high_volume" sets the size to 10GB on 64-bit, and 1GB on 32-bit
  systems.
* Although the maximum value you can set this is 1048576 MB, which
  corresponds to 1 TB, a reasonable number ranges anywhere from 100 to
  50000.  Before proceeding with any higher value, please seek approval
of
  Splunk Support.
* If you specify an invalid number or string, maxDataSize will be auto
  tuned.
* NOTE: The maximum size of your warm buckets may slightly exceed
  'maxDataSize', due to post-processing and timing issues with the
rolling
  policy.

rawFileSizeBytes = <positive integer>
* Deprecated in version 4.2 and later. We will ignore this value.
* Rawdata chunks are no longer stored in individual files.
* If you really need to optimize the new rawdata chunks (highly
unlikely),
  edit rawChunkSizeBytes

rawChunkSizeBytes = <positive integer>
* Target uncompressed size in bytes for individual raw slice in the
rawdata
  journal of the index.
* If 0 is specified, rawChunkSizeBytes will be set to the default
value.
* NOTE: rawChunkSizeBytes only specifies a target chunk size. The
actual
  chunk size may be slightly larger by an amount proportional to an
  individual event size.
* WARNING: This is an advanced parameter. Only change it if you are
  instructed to do so by Splunk Support.
* Must restart splunkd after changing this parameter; index reload will
not
  suffice.
* Highest legal value is 18446744073709551615
* Defaults to 131072 (128KB).

minRawFileSyncSecs = <nonnegative decimal>|disable
* How frequently we force a filesystem sync while compressing journal
  slices.  During this interval, uncompressed slices are left on disk
even
  after they are compressed.  Then we force a filesystem sync of the
  compressed journal and remove the accumulated uncompressed files.
* If 0 is specified, we force a filesystem sync after every slice
completes
  compressing.
* Specifying "disable" disables syncing entirely: uncompressed slices
are
  removed as soon as compression is complete
```

```
* Some filesystems are very inefficient at performing sync operations,
so
  only enable this if you are sure it is needed
* Must restart splunkd after changing this parameter; index reload will
not
  suffice.
* No exponent may follow the decimal.
* Highest legal value is 18446744073709551615
* Defaults to "disable".

maxMemMB = <nonnegative integer>
* The amount of memory to allocate for indexing.
* This amount of memory will be allocated PER INDEX THREAD, or, if
  indexThreads is set to 0, once per index.
* IMPORTANT:  Calculate this number carefully. splunkd will crash if you
set
  this number higher than the amount of memory available.
* The default is recommended for all environments.
* Highest legal value is 4294967295
* Defaults to 5.

maxHotSpanSecs = <positive integer>
* Upper bound of timespan of hot/warm buckets in seconds.
* NOTE: If you set this too small, you can get an explosion of hot/warm
  buckets in the filesystem.
* If you set this parameter to less than 3600, it will be automatically
  reset to 3600, which will then activate snapping behavior (see below).
* This is an advanced parameter that should be set
  with care and understanding of the characteristics of your data.
* If set to 3600 (1 hour), or 86400 (1 day), becomes also the lower
bound
  of hot bucket timespans.  Further, snapping behavior (i.e. ohSnap)
  is activated, whereby hot bucket boundaries will be set at exactly the
  hour or day mark, relative to local midnight.
* Highest legal value is 4294967295
* Defaults to 7776000 seconds (90 days).
* Note that this limit will be applied per ingestion pipeline. For more
  information about multiple ingestion pipelines see
parallelIngestionPipelines
  in server.conf.spec file.
* With N parallel ingestion pipelines, each ingestion pipeline will
write to
  and manage its own set of hot buckets, without taking into account the
state
  of hot buckets managed by other ingestion pipelines.  Each ingestion
pipeline
  will independently apply this setting only to its own set of hot
buckets.

maxHotIdleSecs = <nonnegative integer>
* Provides a ceiling for buckets to stay in hot status without receiving
any
```

```
  data.
* If a hot bucket receives no data for more than maxHotIdleSecs
seconds,
  Splunk rolls it to warm.
* This setting operates independently of maxHotBuckets, which can also
cause
  hot buckets to roll.
* A value of 0 turns off the idle check (equivalent to infinite idle
time).
* Highest legal value is 4294967295
* Defaults to 0.

maxHotBuckets = <positive integer>
* Maximum hot buckets that can exist per index.
* When maxHotBuckets is exceeded, Splunk rolls the least recently used
(LRU)
  hot bucket to warm.
* Both normal hot buckets and quarantined hot buckets count towards
this
  total.
* This setting operates independently of maxHotIdleSecs, which can also
  cause hot buckets to roll.
* Highest legal value is 4294967295
* Defaults to 3.
* Note that this limit will be applied per ingestion pipeline. For more
  information about multiple ingestion pipelines see
parallelIngestionPipelines
  in server.conf.spec file.
* With N parallel ingestion pipelines the maximum number of hot buckets
across
  all of the ingestion pipelines will be N * maxHotBuckets but
maxHotBuckets
  for each ingestion pipeline.  Each ingestion pipeline will
independently
  write to and manage up to maxHotBuckets number of hot buckets.  As a
  consequence of this, when multiple ingestion pipelines are used, there
may
  be multiple (dependent on number of ingestion pipelines configured)
hot
  buckets with events with overlapping time ranges.

minHotIdleSecsBeforeForceRoll = <nonnegative integer>|auto
* When there are no existing hot buckets that can fit new events
because of
  their timestamps and the constraints on the index (refer to
maxHotBuckets,
  maxHotSpanSecs and quarantinePastSecs), if any hot bucket has been
idle
  (i.e. not receiving any data) for minHotIdleSecsBeforeForceRoll
number of
  seconds, a new bucket will be created to receive these new events and
the
```

idle bucket will be rolled to warm.
* If no hot bucket has been idle for minHotIdleSecsBeforeForceRoll number of seconds,
or if minHotIdleSecsBeforeForceRoll has been set to zero, then a best fit bucket
will be chosen for these new events from the existing set of hot buckets.
* This setting operates independently of maxHotIdleSecs, which causes hot buckets
to roll after they have been idle for maxHotIdleSecs number of seconds,
regardless of whether new events can fit into the existing hot buckets or not
due to an event timestamp. minHotIdleSecsBeforeForceRoll, on the other hand,
controls a hot bucket roll *only* under the circumstances when the timestamp
of a new event cannot fit into the existing hot buckets given the other
parameter constraints on the system (parameters such as maxHotBuckets,
maxHotSpanSecs and quarantinePastSecs).
* auto: Specifying "auto" will cause Splunk to autotune this parameter
(recommended). The value begins at 600 seconds but automatically adjusts upwards for
optimal performance. Specifically, the value will increase when a hot bucket rolls
due to idle time with a significantly smaller size than maxDataSize. As a consequence,
the outcome may be fewer buckets, though these buckets may span wider earliest-latest
time ranges of events.
* 0: A value of 0 turns off the idle check (equivalent to infinite idle time).
Setting this to zero means that we will never roll a hot bucket for the
reason that an event cannot fit into an existing hot bucket due to the
constraints of other parameters. Instead, we will find a best fitting
bucket to accommodate that event.
* Highest legal value is 4294967295.
* NOTE: If you set this configuration, there is a chance that this could lead to
frequent hot bucket rolls depending on the value. If your index contains a
large number of buckets whose size-on-disk falls considerably short of the
size specified in maxDataSize, and if the reason for the roll of these buckets
is due to "caller=lru", then setting the parameter value to a larger value or
to zero may reduce the frequency of hot bucket rolls (see AUTO above). You may check
splunkd.log for a similar message below for rolls due to this setting.

```
    INFO  HotBucketRoller - finished moving hot to warm
bid=_internal~0~97597E05-7156-43E5-85B1-B0751462D16B idx=_internal
from=hot_v1_0 to=db_1462477093_1462477093_0 size=40960 caller=lru
maxHotBuckets=3, count=4 hot buckets,evicting_count=1 LRU hots
* Defaults to "auto".

quarantinePastSecs = <positive integer>
* Events with timestamp of quarantinePastSecs older than "now" will be
  dropped into quarantine bucket.
* This is a mechanism to prevent the main hot buckets from being
polluted
  with fringe events.
* Highest legal value is 4294967295
* Defaults to 77760000 (900 days).

quarantineFutureSecs = <positive integer>
* Events with timestamp of quarantineFutureSecs newer than "now" will
be
  dropped into quarantine bucket.
* This is a mechanism to prevent main hot buckets from being polluted
with
  fringe events.
* Highest legal value is 4294967295
* Defaults to 2592000 (30 days).

maxMetaEntries = <nonnegative integer>
* Sets the maximum number of unique lines in .data files in a bucket,
which
  may help to reduce memory consumption
* If exceeded, a hot bucket is rolled to prevent further increase
* If your buckets are rolling due to Strings.data hitting this limit,
the
  culprit may be the 'punct' field in your data.  If you do not use
punct,
  it may be best to simply disable this (see props.conf.spec)
  * NOTE: since at least 5.0.x, large strings.data from punct will be
rare.
* There is a delta between when maximum is exceeded and bucket is
rolled.
* This means a bucket may end up with epsilon more lines than specified,
but
  this is not a major concern unless excess is significant
* If set to 0, this setting is ignored (it is treated as infinite)
* Highest legal value is 4294967295

syncMeta = true|false
* When "true", a sync operation is called before file descriptor is
closed
  on metadata file updates.
* This functionality was introduced to improve integrity of metadata
files,
  especially in regards to operating system crashes/machine failures.
```

```
* NOTE: Do not change this parameter without the input of a Splunk
support
  professional.
* Must restart splunkd after changing this parameter; index reload will
not
  suffice.
* Defaults to true.

serviceMetaPeriod = <positive integer>
* Defines how frequently metadata is synced to disk, in seconds.
* Defaults to 25 (seconds).
* You may want to set this to a higher value if the sum of your
metadata
  file sizes is larger than many tens of megabytes, to avoid the hit on
I/O
  in the indexing fast path.
* Highest legal value is 4294967295

partialServiceMetaPeriod = <positive integer>
* Related to serviceMetaPeriod.  If set, it enables metadata sync every
  <integer> seconds, but only for records where the sync can be done
  efficiently in-place, without requiring a full re-write of the
metadata
  file.  Records that require full re-write will be synced at
  serviceMetaPeriod.
* <integer> specifies how frequently it should sync.  Zero means that
this
  feature is turned off and serviceMetaPeriod is the only time when
metadata
  sync happens.
* If the value of partialServiceMetaPeriod is greater than
  serviceMetaPeriod, this setting will have no effect.
* By default it is turned off (zero).
* This parameter is ignored if serviceOnlyAsNeeded = true (the
default).
* Highest legal value is 4294967295

throttleCheckPeriod = <positive integer>
* Defines how frequently Splunk checks for index throttling condition,
  in seconds.
* NOTE: Do not change this parameter without the input of a Splunk
Support
  professional.
* Highest legal value is 4294967295
* Defaults to 15

maxTimeUnreplicatedWithAcks = <nonnegative decimal>
* Important if you have enabled indexer acknowledgements (ack) on
forwarders
  and have replication enabled (via Index Clustering)
* This parameter puts an upper limit on how long events can sit
unacknowledged
```

```
  in a raw slice
* To disable this, you can set to 0, but this is NOT recommended!!!
* NOTE: This is an advanced parameter; make sure you understand the
settings
        on all your forwarders before changing this.  This number should
not
        exceed ack timeout configured on any forwarders, and should
indeed
        be set to at most half of the minimum value of that timeout.
 You
        can find this setting in outputs.conf readTimeout setting, under
        the tcpout stanza.
* Highest legal value is 2147483647
* Defaults to 60 (seconds)

maxTimeUnreplicatedNoAcks = <nonnegative decimal>
* Important only if replication is enabled for this index, otherwise
ignored
* This parameter puts an upper limit on how long an event can sit in raw
  slice.
* If there are any ack''d events sharing this raw slice, this parameter
will
  not apply (maxTimeUnreplicatedWithAcks will be used instead)
* Highest legal value is 2147483647
* To disable this, you can set to 0; please be careful and understand
the
  consequences before changing this parameter
* Defaults to 60 (seconds)

isReadOnly = true|false
* Set to true to make an index read-only.
* If true, no new events can be added to the index, but the index is
still
  searchable.
* Must restart splunkd after changing this parameter; index reload will
not
  suffice.
* Defaults to false.

homePath.maxDataSizeMB = <nonnegative integer>
* Specifies the maximum size of homePath (which contains hot and warm
  buckets).
* If this size is exceeded, Splunk will move buckets with the oldest
value
  of latest time (for a given bucket) into the cold DB until homePath
is
  below the maximum size.
* If this attribute is missing or set to 0, Splunk will not constrain
the
  size of homePath.
* Highest legal value is 4294967295
* Defaults to 0.
```

```
coldPath.maxDataSizeMB = <nonnegative integer>
* Specifies the maximum size of coldPath (which contains cold buckets).
* If this size is exceeded, Splunk will freeze buckets with the oldest
value
  of latest time (for a given bucket) until coldPath is below the
maximum
  size.
* If this attribute is missing or set to 0, Splunk will not constrain
size
  of coldPath
* If we freeze buckets due to enforcement of this policy parameter, and
  coldToFrozenScript and/or coldToFrozenDir archiving parameters are
also
  set on the index, these parameters *will* take into effect
* Highest legal value is 4294967295
* Defaults to 0.

disableGlobalMetadata = true|false
* NOTE: This option was introduced in 4.3.3, but as of 5.0 it is
obsolete
  and ignored if set.
* It used to disable writing to the global metadata.  In 5.0 global
metadata
  was removed.

repFactor = <nonnegative integer>|auto
* Only relevant if this instance is a clustering slave (but see note
about
  "auto" below).
* See server.conf spec for details on clustering configuration.
* Value of 0 turns off replication for this index.
* If set to "auto", slave will use whatever value the master has.
* Highest legal value is 4294967295
* Defaults to 0.

minStreamGroupQueueSize = <nonnegative integer>
* Minimum size of the queue that stores events in memory before
committing
  them to a tsidx file.  As Splunk operates, it continually adjusts this
  size internally.  Splunk could decide to use a small queue size and
thus
  generate tiny tsidx files under certain unusual circumstances, such as
  file system errors.  The danger of a very low minimum is that it can
  generate very tiny tsidx files with one or very few events, making it
  impossible for splunk-optimize to catch up and optimize the tsidx
files
  into reasonably sized files.
* Defaults to 2000.
* Only set this value if you have been advised to by Splunk Support.
* Highest legal value is 4294967295
```

```
streamingTargetTsidxSyncPeriodMsec = <nonnegative integer>
* Period we force sync tsidx files on streaming targets. This setting is
  needed for multi-site clustering where streaming targets may be
primary.
* if set to 0, we never sync (equivalent to infinity)

journalCompression = gzip|lz4
* Select compression algorithm for rawdata journal file
* Defaults to gzip

enableTsidxReduction = true|false
* By enabling this setting, you turn on the tsidx reduction capability.
This causes the
  indexer to reduce the tsidx files of buckets, when the buckets reach
the age specified
  by timePeriodInSecBeforeTsidxReduction.
* Defaults to false.

suspendHotRollByDeleteQuery = true|false
* When the "delete" search command is run, all buckets containing data
to be deleted are
  marked for updating of their metadata files. The indexer normally
first rolls any hot buckets,
  as rolling must precede the metadata file updates.
* When suspendHotRollByDeleteQuery is set to true, the rolling of hot
buckets for the "delete"
  command is suspended. The hot buckets, although marked, do not roll
immediately, but instead
  wait to roll in response to the same circumstances operative for any
other hot buckets; for
  example, due to reaching a limit set by maxHotBuckets, maxDataSize,
etc. When these hot buckets
  finally roll, their metadata files are then updated.
* Defaults to false

tsidxReductionCheckPeriodInSec = <positive integer>
* Time period between service runs to reduce the tsidx files for any
buckets that have
  reached the age specified by timePeriodInSecBeforeTsidxReduction.
* Defaults to 600 (seconds).

timePeriodInSecBeforeTsidxReduction = <positive integer>
* Age at which buckets become eligible for tsidx reduction.
  The bucket age is the difference between the current time
  and the timestamp of the bucket's latest event.
* Defaults to 604800 (seconds).

#*******************************************************************************
```

PER PROVIDER FAMILY OPTIONS

```
# PER PROVIDER FAMILY OPTIONS
# A provider family is a way of collecting properties that are common to
# multiple providers. There are no properties that can only be used in a
# provider family, and not in a provider. If the same property is
specified
# in a family, and in a provider belonging to that family, then the
latter
# value "wins".
#
# All family stanzas begin with "provider-family:". For example:
# [provider-family:family_name]
# vix.mode=stream
# vix.command = java
# vix.command.arg.1 = -Xmx512m
# ....
#*******************************************************************************

#*******************************************************************************
```

PER PROVIDER OPTIONS

```
# PER PROVIDER OPTIONS
# These options affect External Resource Providers. All provider stanzas
# begin with "provider:". For example:
#   [provider:provider_name]
#   vix.family                  = hadoop
#   vix.env.JAVA_HOME           = /path/to/java/home
#   vix.env.HADOOP_HOME         = /path/to/hadoop/client/libraries
#
# Each virtual index must reference a provider.
#*******************************************************************************
vix.family = <family>
* A provider family to which this provider belongs.
* The only family available by default is "hadoop". Others may be added.

vix.mode = stream|report
* Usually specified at the family level.
* Typically should be "stream". In general, do not use "report" without
  consulting Splunk Support.

vix.command = <command>
* The command to be used to launch an external process for searches on
this
  provider.
* Usually specified at the family level.

vix.command.arg.<N> = <argument>
```

```
* The Nth argument to the command specified by vix.command.
* Usually specified at the family level, but frequently overridden at
the
  provider level, for example to change the jars used depending on the
  version of Hadoop to which a provider connects.

vix.<property name> = <property value>
* All such properties will be made available as "configuration
properties" to
  search processes on this provider.
* For example, if this provider is in the Hadoop family, the
configuration
  property "mapreduce.foo = bar" can be made available to the Hadoop
  via the property "vix.mapreduce.foo = bar".

vix.env.<env var name> = <env var variable>
* Will create an environment variable available to search processes on
this
  provider.
* For example, to set the JAVA_HOME variable to "/path/java" for search
  processes on this provider, use "vix.env.JAVA_HOME = /path/java".

#********************************************************************
# PER PROVIDER OPTIONS -- HADOOP
# These options are specific to ERPs with the Hadoop family.
# NOTE: Many of these properties specify behavior if the property is not
#       set. However, default values set in system/default/indexes.conf
#       take precedence over the "unset" behavior.
#********************************************************************

vix.javaprops.<JVM system property name> = <value>
* All such properties will be used as Java system properties.
* For example, to specify a Kerberos realm (say "foo.com") as a Java
  system property, use the property
  "vix.javaprops.java.security.krb5.realm = foo.com".

vix.mapred.job.tracker = <logical name or server:port>
* In high-availability mode, use the logical name of the Job Tracker.
* Otherwise, should be set to server:port for the single Job Tracker.
* Note: this property is passed straight to Hadoop. Not all such
properties
  are documented here.

vix.fs.default.name = <logical name or hdfs://server:port>
* In high-availability mode, use the logical name for a list of Name
Nodes.
* Otherwise, use the URL for the single Name Node.
* Note: this property is passed straight to Hadoop. Not all such
properties
  are documented here.

vix.splunk.setup.onsearch = true|false
```

```
* Whether to perform setup (install & bundle replication) on search.
* Defaults to false.

vix.splunk.setup.package = current|<path to file>
* Splunk .tgz package to install and use on data nodes
  (in vix.splunk.home.datanode).
* Uses the current install if set to value 'current' (without quotes).

vix.splunk.home.datanode = <path to dir>
* Path to where splunk should be installed on datanodes/tasktrackers,
i.e.
  SPLUNK_HOME.
* Required.

vix.splunk.home.hdfs = <path to dir>
* Scratch space for this Splunk instance on HDFS
* Required.

vix.splunk.search.debug = true|false
* Whether to run searches against this index in debug mode. In debug
mode,
  additional information is logged to search.log.
* Optional. Defaults to false.

vix.splunk.search.recordreader = <list of classes>
* Comma separated list of data preprocessing classes.
* Each such class must extend BaseSplunkRecordReader and return data to
be
  consumed by Splunk as the value.

vix.splunk.search.splitter = <class name>
* Set to override the class used to generate splits for MR jobs.
* Classes must implement com.splunk.mr.input.SplitGenerator.
* Unqualified classes will be assumed to be in the package
com.splunk.mr.input.
* To search Parquet files, use ParquetSplitGenerator.
* To search Hive files, use HiveSplitGenerator.

vix.splunk.search.mr.threads = <postive integer>
* Number of threads to use when reading map results from HDFS
* Numbers less than 1 will be treated as 1.
* Numbers greater than 50 will be treated as 50.
* If not set, defaults to 10.

vix.splunk.search.mr.maxsplits = <positive integer>
* Maximum number of splits in an MR job.
* If not set, defaults to 10000.

vix.splunk.search.mr.minsplits = <positive integer>
* Number of splits for first MR job associated with a given search.
* If not set, defaults to 100.
```

```
vix.splunk.search.mr.splits.multiplier = <decimal greater than or equal
to 1.0>
* Factor by which the number of splits is increased in consecutive MR
jobs for
  a given search, up to the value of maxsplits.
* If not set, defaults to 10.

vix.splunk.search.mr.poll = <positive integer>
* Polling period for job status, in milliseconds.
* If not set, defaults to 1000 (ie. 1 second).

vix.splunk.search.mr.mapper.output.replication = <positive integer>
* Replication level for mapper output.
* Defaults to 3.

vix.splunk.search.mr.mapper.output.gzlevel = <integer between 0 and 9,
inclusive>
* The compression level used for the mapper output.
* Defaults to 2.

vix.splunk.search.mixedmode = true|false
* Whether mixed mode execution is enabled.
* Defaults to true.

vix.splunk.search.mixedmode.maxstream = <nonnegative integer>
* Max # of bytes to stream during mixed mode.
* Value = 0 means there's no stream limit.
* Will stop streaming after the first split that took the value over
the limit.
* If not set, defaults to 10 GB.

vix.splunk.jars = <list of paths>
* Comma delimited list of Splunk dirs/jars to add to the classpath in
the
  Search Head and MR.

vix.env.HUNK_THIRDPARTY_JARS = <list of paths>
* Comma delimited list of 3rd-party dirs/jars to add to the classpath in
the
  Search Head and MR.

vix.splunk.impersonation = true|false
* Enable/disable user impersonation.

vix.splunk.setup.bundle.replication = <positive integer>
* Set custom replication factor for bundles on HDFS.
* Must be an integer between 1 and 32767.
* Increasing this setting may help performance on large clusters by
decreasing
  the average access time for a bundle across Task Nodes.
* Optional. If not set, the default replication factor for the
file-system
```

```
  will apply.

vix.splunk.setup.bundle.max.inactive.wait = <positive integer>
* A positive integer represent a time interval in seconds.
* Defaults to 5.
* While a task waits for a bundle being replicated to the same node by
another
  task, if the bundle file is not modified for this amount of time, the
task
  will begin its own replication attempt.

vix.splunk.setup.bundle.poll.interval = <positive integer>
* A positive number, representing a time interval in milliseconds.
* Defaults to 100.
* While a task waits for a bundle to be installed by another task on the
same
  node, it will check once per interval whether that installation is
complete.

vix.splunk.setup.bundle.setup.timelimit = <positive integer>
* A postive number, representing a time duration in milliseconds.
* Defaults to 20,000 (i.e. 20 seconds).
* A task will wait this long for a bundle to be installed before it
quits.

vix.splunk.setup.package.replication = true|false
* Set custom replication factor for the Splunk package on HDFS. This is
the
  package set in the property vix.splunk.setup.package.
* Must be an integer between 1 and 32767.
* Increasing this setting may help performance on large clusters by
decreasing
  the average access time for the package across Task Nodes.
* Optional. If not set, the default replication factor for the
file-system
  will apply.

vix.splunk.setup.package.max.inactive.wait = <positive integer>
* A positive integer represent a time interval in seconds.
* Defaults to 5.
* While a task waits for a Splunk package being replicated to the same
node by
  another task, if the package file is not modified for this amount of
time,
  the task will begin its own replication attempt.

vix.splunk.setup.package.poll.interval = <positive integer>
* A positive number, representing a time interval in milliseconds.
* Defaults to 100.
* While a task waits for a Splunk package to be installed by another
task on
  the same node, it will check once per interval whether that
```

```
installation is
  complete.

vix.splunk.setup.package.setup.timelimit = <positive integer>
* A positive number, representing a time duration in milliseconds.
* Defaults to 20,000 (i.e. 20 seconds).
* A task will wait this long for a Splunk package to be installed before
it quits.

vix.splunk.search.column.filter = true|false
* Enables/disables column filtering. When enabled, Hunk will trim
columns that
  are not necessary to a query on the Task Node, before returning the
results
  to the search process.
* Should normally increase performance, but does have its own small
overhead.
* Works with these formats: CSV, Avro, Parquet, Hive.
* If not set, defaults to true.

#
# Kerberos properties
#

vix.kerberos.principal = <kerberos principal name>
* Specifies principal for Kerberos authentication.
* Should be used with vix.kerberos.keytab and either
  1) vix.javaprops.java.security.krb5.realm and
     vix.javaprops.java.security.krb5.kdc, or
  2) security.krb5.conf

vix.kerberos.keytab = <kerberos keytab path>
* Specifies path to keytab for Kerberos authentication.
* See usage note with vix.kerberos.principal.


#
# The following properties affect the SplunkMR heartbeat mechanism. If
this
# mechanism is turned on, the SplunkMR instance on the Search Head
updates a
# heartbeat file on HDFS. Any MR job spawned by report or mix-mode
searches
# checks the heartbeat file. If it is not updated for a certain time, it
will
# consider SplunkMR to be dead and kill itself.
#

vix.splunk.heartbeat = true|false
* Turn on/off heartbeat update on search head, and checking on MR side.
* If not set, defaults to true.
```

```
vix.splunk.heartbeat.path = <path on HDFS>
* Path to heartbeat file.
* If not set, defaults to <vix.splunk.home.hdfs>/dispatch/<sid>/

vix.splunk.heartbeat.interval = <positive integer>
* Frequency with which the Heartbeat will be updated on the Search Head.
* Unit is millisecond.
* Default value is 6 seconds (6000).
* Minimum value is 1000. Smaller values will cause an exception to be
thrown.

vix.splunk.heartbeat.threshold = <postive integer>
* The number of times the MR job will detect a missing heartbeat update
before
  it considers SplunkMR dead and kills itself.
* Default value is 10.

## The following sections are specific to data input types.

#
# Sequence file
#

vix.splunk.search.recordreader.sequence.ignore.key = true|false
* When reading sequence files, if this key is enabled, events will be
expected
  to only include a value. Otherwise, the expected representation is
  key+"\t"+value.
* Defaults to true.

#
# Avro
#

vix.splunk.search.recordreader.avro.regex = <regex>
* Regex that files must match in order to be considered avro files.
* Optional. Defaults to \.avro$

#
# Parquet
#

vix.splunk.search.splitter.parquet.simplifyresult = true|false
* If enabled, field names for map and list type fields will be
simplified by
  dropping intermediate "map" or "element" subfield names. Otherwise, a
field
  name will match parquet schema completely.
* Defaults to true.

#
# Hive
```

```
#

vix.splunk.search.splitter.hive.ppd = true|false
* Enable or disable Hive ORC Predicate Push Down.
* If enabled, ORC PPD will be applied whenever possible to prune
unnecessary
  data as early as possible to optimize the search.
* If not set, defaults to true.

vix.splunk.search.splitter.hive.fileformat =
textfile|sequencefile|rcfile|orc
* Format of the Hive data files in this provider.
* If not set, defaults to "textfile".

vix.splunk.search.splitter.hive.dbname = <DB name>
* Name of Hive database to be accessed by this provider.
* Optional. If not set, defaults to "default".

vix.splunk.search.splitter.hive.tablename = <table name>
* Table accessed by this provider.
* Required property.

vix.splunk.search.splitter.hive.columnnames = <list of column names>
* Comma-separated list of file names.
* Required if using Hive, not using metastore.

vix.splunk.search.splitter.hive.columntypes = string:float:int # COLON
separated list of column types, required
* Colon-separated list of column- types.
* Required if using Hive, not using metastore.

vix.splunk.search.splitter.hive.serde = <SerDe class>
* Fully-qualified class name of SerDe.
* Required if using Hive, not using metastore, and if specified in
creation of Hive table.

vix.splunk.search.splitter.hive.serde.properties = <list of key-value
pairs>
* Comma-separated list of "key=value" pairs.
* Required if using Hive, not using metastore, and if specified in
creation of Hive table.

vix.splunk.search.splitter.hive.fileformat.inputformat = <InputFormat
class>
* Fully-qualified class name of an InputFormat to be used with Hive
table data.

vix.splunk.search.splitter.hive.rowformat.fields.terminated =
<delimiter>
* Will be set as the Hive SerDe property "field.delim".
* Optional.
```

```
vix.splunk.search.splitter.hive.rowformat.escaped = <escape char>
* Will be set as the Hive SerDe property "escape.delim".
* Optional.

vix.splunk.search.splitter.hive.rowformat.lines.terminated =
<delimiter>
* Will be set as the Hive SerDe property "line.delim".
* Optional.

vix.splunk.search.splitter.hive.rowformat.mapkeys.terminated  =
<delimiter>
* Will be set as the Hive SerDe property "mapkey.delim".
* Optional.

vix.splunk.search.splitter.hive.rowformat.collectionitems.terminated =
<delimiter>
* Will be set as the Hive SerDe property "colelction.delim".
* Optional.

#
# Archiving
#

vix.output.buckets.max.network.bandwidth = 0|<bits per second>
* Throttles network bandwidth to <bits per second>
* Defaults to 0, meaning no throttling.
* Set at provider level. Applied to all virtual indexes using a provider
with this setting.

#*******************************************************************************
```

PER VIRTUAL INDEX OPTIONS

```
# PER VIRTUAL INDEX OPTIONS
# These options affect virtual indexes. Like indexes, these options may
# be set under an [<virtual-index>] entry.
#
# Virtual index names have the same constraints as normal index names.
#
# Each virtual index must reference a provider. I.e:
# [virtual_index_name]
# vix.provider = <provider_name>
#
# All configuration keys starting with "vix." will be passed to the
# external resource provider (ERP).
#*******************************************************************************

vix.provider = <provider_name>
* Name of the external resource provider to use for this virtual index.
```

```
#*******************************************************************************
# PER VIRTUAL INDEX OPTIONS -- HADOOP
# These options are specific to ERPs with the Hadoop family.
#*******************************************************************************

#
# The vix.input.* configurations are grouped by an id.
# Inputs configured via the UI always use '1' as the id.
# In this spec we'll use 'x' as the id.
#

vix.input.x.path = <path>
* Path in a hadoop filesystem (usually HDFS or S3).
* May contain wildcards.
* Checks the path for data recursively when ending with '...'
* Can extract fields with ${field}. I.e: "/data/${server}/...", where
server
  will be extracted.
* May start with a schema.
  * The schema of the path specifies which hadoop filesystem
implementation to
    use. Examples:
    * hdfs://foo:1234/path, will use a HDFS filesystem implementation
    * s3a://s3-bucket/path, will use a S3 filesystem implementation

vix.input.x.accept = <regex>
* Specifies a whitelist regex.
* Only files within the location given by matching vix.input.x.path,
whose
  paths match this regex, will be searched.

vix.input.x.ignore = <regex>
* Specifies a blacklist regex.
* Searches will ignore paths matching this regex.
* These matches take precedence over vix.input.x.accept matches.

vix.input.x.required.fields = <comma separated list of fields>
* Fields that will be kept in search results even if the field is not
required by the search

# Earliest time extractions - For all 'et' settings, there's an
equivalent 'lt' setting.
vix.input.x.et.regex = <regex>
* Regex extracting earliest time from vix.input.x.path

vix.input.x.et.format = <java.text.SimpleDateFormat date pattern>
* Format of the extracted earliest time.
* See documentation for java.text.SimpleDateFormat

vix.input.x.et.offset = <seconds>
* Offset in seconds to add to the extracted earliest time.
```

```
vix.input.x.et.timezone = <java.util.SimpleTimeZone timezone id>
* Timezone in which to interpret the extracted earliest time.
* Examples: "America/Los_Angeles" or "GMT-8:00"

vix.input.x.et.value = mtime|<epoch time in milliseconds>
* Sets the earliest time for this virtual index.
* Can be used instead of extracting times from the path via
vix.input.x.et.regex
* When set to "mtime", uses the file modification time as the earliest
time.

# Latest time extractions - See "Earliest time extractions"

vix.input.x.lt.regex = <regex>
* Latest time equivalent of vix.input.x.et.regex

vix.input.x.lt.format = <java.text.SimpleDateFormat date pattern>
* Latest time equivalent of vix.input.x.et.format

vix.input.x.lt.offset = <seconds>
* Latest time equivalent of vix.input.x.et.offset

vix.input.x.lt.timezone = <java.util.SimpleTimeZone timezone id>
* Latest time equivalent of vix.input.x.et.timezone

vix.input.x.lt.value = <mod time>
* Latest time equivalent of vix.input.x.et.value

#
# Archiving
#

vix.output.buckets.path = <hadoop path>
* Path to a hadoop filesystem where buckets will be archived

vix.output.buckets.older.than = <seconds>
* Buckets must be this old before they will be archived.
* A bucket's age is determined by the the earliest _time field of any
event in
  the bucket.

vix.output.buckets.from.indexes = <comma separated list of splunk
indexes>
* List of (non-virtual) indexes that will get archived to this
(virtual) index.

vix.unified.search.cutoff_sec = <seconds>
* Window length before present time that configures where events are
retrieved
  for unified search
* Events from now to now-cutoff_sec will be retrieved from the splunk
index
```

```
  and events older than cutoff_sec will be retrieved from the archive
index

#********************************************************************************
# PER VIRTUAL INDEX OR PROVIDER OPTIONS -- HADOOP
# These options can be set at either the virtual index level or provider
# level, for the Hadoop ERP.
#
# Options set at the virtual index level take precedence over options
set
# at the provider level.
#
# Virtual index level prefix:
# vix.input.<input_id>.<option_suffix>
#
# Provider level prefix:
# vix.splunk.search.<option_suffix>
#********************************************************************************

# The following options are just defined by their <option_suffix>

#
# Record reader options
#

recordreader.<name>.<conf_key> = <conf_value>
* Sets a configuration key for a RecordReader with <name> to
<conf_value>

recordreader.<name>.regex = <regex>
* Regex specifying which files this RecordReader can be used for.

recordreader.journal.buffer.size = <bytes>
* Buffer size used by the journal record reader

recordreader.csv.dialect = default|excel|excel-tab|tsv
* Set the csv dialect for csv files
* A csv dialect differs on delimiter_char, quote_char and escape_char.
* Here is a list of how the different dialects are defined in order
delim,
  quote, and escape:
  * default   = ,  " \
  * excel     = ,  " "
  * excel-tab = \t " "
  * tsv       = \t " \

#
# Splitter options
#

splitter.<name>.<conf_key> = <conf_value>
* Sets a configuration key for a split generator with <name> to
```

```
<conf_value>

splitter.file.split.minsize = <bytes>
* Minimum size in bytes for file splits.
* Defaults to 1.

splitter.file.split.maxsize = <bytes>
* Maximum size in bytes for file splits.
* Defaults to Long.MAX_VALUE.

#*******************************************************************************
# Volume settings.  This section describes settings that affect the
volume-
# optional and volume-mandatory parameters only.
#
# All volume stanzas begin with "volume:". For example:
#   [volume:volume_name]
#   path = /foo/bar
#
# These volume stanzas can then be referenced by individual index
# parameters, e.g. homePath or coldPath.  To refer to a volume stanza,
use
# the "volume:" prefix. For example, to set a cold DB to the example
stanza
# above, in index "hiro", use:
#   [hiro]
#   coldPath = volume:volume_name/baz
# This will cause the cold DB files to be placed under /foo/bar/baz.
 If the
# volume spec is not followed by a path
# (e.g.  "coldPath=volume:volume_name"), then the cold path would be
# composed by appending the index name to the volume name
("/foo/bar/hiro").
#
# Note: thawedPath may not be defined in terms of a volume.
# Thawed allocations are manually controlled by Splunk administrators,
# typically in recovery or archival/review scenarios, and should not
# trigger changes in space automatically used by normal index activity.
#*******************************************************************************

path = <path on server>
* Required.
* Points to the location on the file system where all databases that
use
  this volume will reside.  You must make sure that this location does
not
  overlap with that of any other volume or index database.

maxVolumeDataSizeMB = <positive integer>
* Optional.
* If set, this attribute limits the total size of all databases that
reside
```

```
  on this volume to the maximum size specified, in MB.  Note that this
it
  will act only on those indexes which reference this volume, not on the
  total size of the path set in the path attribute of this volume.
* If the size is exceeded, Splunk will remove buckets with the oldest
value
  of latest time (for a given bucket) across all indexes in the volume,
  until the volume is below the maximum size.  This is the trim
operation.
  Note that this can cause buckets to be chilled [moved to cold]
directly
  from a hot DB, if those buckets happen to have the least value of
  latest-time (LT) across all indexes in the volume.
* Highest legal value is 4294967295, lowest legal value is 1.

rotatePeriodInSecs = <nonnegative integer>
* Optional.
* Specifies period of trim operation for this volume.
* If not set, the value of global rotatePeriodInSecs attribute is
inherited.
* Highest legal value is 4294967295
```

indexes.conf.example

```
#   Version 6.5.0
#
# This file contains an example indexes.conf.  Use this file to
configure
# indexing properties.
#
# To use one or more of these configurations, copy the configuration
block
# into indexes.conf in $SPLUNK_HOME/etc/system/local/. You must restart
# Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
#

# The following example defines a new high-volume index, called "hatch",
and
# sets this to be the default index for both incoming data and search.
#
# Note that you may want to adjust the indexes that your roles have
access
# to when creating indexes (in authorize.conf)
```

```
defaultDatabase = hatch

[hatch]

homePath   = $SPLUNK_DB/hatchdb/db
coldPath   = $SPLUNK_DB/hatchdb/colddb
thawedPath = $SPLUNK_DB/hatchdb/thaweddb
maxDataSize = 10000
maxHotBuckets = 10



# The following example changes the default amount of space used on a
# per-index basis.

[default]
maxTotalDataSizeMB = 650000


# The following example changes the time data is kept around by default.
# It also sets an export script.  NOTE: You must edit this script to
set
# export location before running it.

[default]
maxWarmDBCount = 200
frozenTimePeriodInSecs = 432000
rotatePeriodInSecs = 30
coldToFrozenScript = "$SPLUNK_HOME/bin/python"
"$SPLUNK_HOME/bin/myColdToFrozenScript.py"

# This example freezes buckets on the same schedule, but lets Splunk do
the
# freezing process as opposed to a script
[default]
maxWarmDBCount = 200
frozenTimePeriodInSecs = 432000
rotatePeriodInSecs = 30
coldToFrozenDir = "$SPLUNK_HOME/myfrozenarchive"

### This example demonstrates the use of volumes ###

# volume definitions; prefixed with "volume:"

[volume:hot1]
path = /mnt/fast_disk
maxVolumeDataSizeMB = 100000

[volume:cold1]
path = /mnt/big_disk
# maxVolumeDataSizeMB not specified: no data size limitation on top of
```

```
the
# existing ones

[volume:cold2]
path = /mnt/big_disk2
maxVolumeDataSizeMB = 1000000

# index definitions

[idx1]
homePath = volume:hot1/idx1
coldPath = volume:cold1/idx1

# thawedPath must be specified, and cannot use volume: syntax
# choose a location convenient for reconstitition from archive goals
# For many sites, this may never be used.
thawedPath = $SPLUNK_DB/idx1/thaweddb

[idx2]
# note that the specific indexes must take care to avoid collisions
homePath = volume:hot1/idx2
coldPath = volume:cold2/idx2
thawedPath = $SPLUNK_DB/idx2/thaweddb

[idx3]
homePath = volume:hot1/idx3
coldPath = volume:cold2/idx3
thawedPath = $SPLUNK_DB/idx3/thaweddb

### Indexes may be allocated space in effective groups by sharing
volumes  ###

# perhaps we only want to keep 100GB of summary data and other
# low-volume information
[volume:small_indexes]
path = /mnt/splunk_indexes
maxVolumeDataSizeMB = 100000

# and this is our main event series, allowing 50 terabytes
[volume:large_indexes]
path = /mnt/splunk_indexes
maxVolumeDataSizeMB = 50000000

# summary and rare_data together will be limited to 100GB
[summary]
homePath=volume:small_indexes/summary/db
coldPath=volume:small_indexes/summary/colddb
thawedPath=$SPLUNK_DB/summary/thaweddb
# low-volume indexes probably don't want a lot of hot buckets
maxHotBuckets = 2
# if the volume is quite low, and you have data sunset goals you may
# want to have smaller buckets
```

```
maxDataSize = 500


[rare_data]
homePath=volume:small_indexes/rare_data/db
coldPath=volume:small_indexes/rare_data/colddb
thawedPath=$SPLUNK_DB/rare_data/thaweddb
maxHotBuckets = 2

# main, and any other large volume indexes you add sharing
large_indexes
# will be together be constrained to 50TB, separately from the 100GB of
# the small_indexes
[main]
homePath=volume:large_indexes/main/db
coldPath=volume:large_indexes/main/colddb
thawedPath=$SPLUNK_DB/main/thaweddb
# large buckets and more hot buckets are desirable for higher volume
# indexes, and ones where the variations in the timestream of events is
# hard to predict.
maxDataSize = auto_high_volume
maxHotBuckets = 10

[idx1_large_vol]
homePath=volume:large_indexes/idx1_large_vol/db
coldPath=volume:large_indexes/idx1_large_vol/colddb
homePath=$SPLUNK_DB/idx1_large/thaweddb
# this index will exceed the default of .5TB requiring a change to
maxTotalDataSizeMB
maxTotalDataSizeMB = 750000
maxDataSize = auto_high_volume
maxHotBuckets = 10
# but the data will only be retained for about 30 days
frozenTimePeriodInSecs = 2592000

### This example demonstrates database size constraining ###

# In this example per-database constraint is combined with volumes.
 While a
# central volume setting makes it easy to manage data size across
multiple
# indexes, there is a concern that bursts of data in one index may
# significantly displace data from others.  The homePath.maxDataSizeMB
setting
# can be used to assure that no index will ever take more than certain
size,
# therefore alleviating the concern.

# global settings

# will be inherited by all indexes: no database will exceed 1TB
homePath.maxDataSizeMB = 1000000
```

```
# volumes

[volume:caliente]
path = /mnt/fast_disk
maxVolumeDataSizeMB = 100000

[volume:frio]
path = /mnt/big_disk
maxVolumeDataSizeMB = 1000000

# and this is our main event series, allowing about 50 terabytes
[volume:large_indexes]
path = /mnt/splunk_indexes
maxVolumeDataSizeMB = 50000000

# indexes

[i1]
homePath = volume:caliente/i1
# homePath.maxDataSizeMB is inherited
coldPath = volume:frio/i1
# coldPath.maxDataSizeMB not specified: no limit - old-style behavior

thawedPath = $SPLUNK_DB/i1/thaweddb

[i2]
homePath = volume:caliente/i2
# overrides the default maxDataSize
homePath.maxDataSizeMB = 1000
coldPath = volume:frio/i2
# limits the cold DB's
coldPath.maxDataSizeMB = 10000
thawedPath = $SPLUNK_DB/i2/thaweddb

[i3]
homePath = /old/style/path
homePath.maxDataSizeMB = 1000
coldPath = volume:frio/i3
coldPath.maxDataSizeMB = 10000
thawedPath = $SPLUNK_DB/i3/thaweddb

# main, and any other large volume indexes you add sharing
large_indexes
# will together be constrained to 50TB, separately from the rest of
# the indexes
[main]
homePath=volume:large_indexes/main/db
coldPath=volume:large_indexes/main/colddb
thawedPath=$SPLUNK_DB/main/thaweddb
# large buckets and more hot buckets are desirable for higher volume
indexes
```

```
maxDataSize = auto_high_volume
maxHotBuckets = 10
```

inputs.conf

The following are the spec and example files for inputs.conf.

inputs.conf.spec

```
#   Version 6.5.0

# This file contains possible settings you can use to configure inputs,
# distributed inputs such as forwarders, and file system monitoring in
# inputs.conf.
#
# There is an inputs.conf in $SPLUNK_HOME/etc/system/default/.  To set
custom
# configurations, place an inputs.conf in
$SPLUNK_HOME/etc/system/local/.  For
# examples, see inputs.conf.example. You must restart Splunk to enable
new
# configurations.
#
# To learn more about configuration files (including precedence), see
the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
#
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
top of
#     the file.
#   * Each conf file should have at most one default stanza. If there
are
#     multiple default stanzas, settings are combined. In the case of
#     multiple definitions of the same setting, the last definition in
the
#     file wins.
```

```
#   * If an setting is defined at both the global level and in a
specific
#   stanza, the value in the specific stanza takes precedence.

#*******
# GENERAL SETTINGS:
# The following settings are valid for all input types (except file
system
# change monitor, which is described in a separate section in this
file).
# You must first enter a stanza header in square brackets, specifying
the input
# type. See further down in this file for examples.
# Then, use any of the following settings.
#*******

host = <string>
* Sets the host key/field to a static value for this stanza.
* Primarily used to control the host field, which the input applies to
events
  that come in through this input stanza.
* Detail: Sets the host key initial value. The input uses this key
during
  parsing/indexing, in particular to set the host field. It also uses
this
  field at search time.
* As a convenience, the input prepends the chosen string with 'host::'.
* WARNING: Do not put the <string> value in quotes. Use host=foo, not
host="foo".
* If set to '$decideOnStartup', will be interpreted as hostname of
executing
  machine; this will occur on each splunkd startup.
* If you run multiple instances of the software on the same system
(hardware
  or virtual machine), choose unique values for 'host' to differentiate
  your data, e.g. myhost-sh-1 or myhost-idx-2.
* The literal default conf value is $decideOnStartup, but at
installation
  time, the setup logic adds the local hostname as determined by DNS to
the
  $SPLUNK_HOME/etc/system/local/inputs.conf default stanza, which is
the
  effective default value.

index = <string>
* Sets the index to store events from this input.
* Primarily used to specify the index to store events coming in via this
input
  stanza.
* Detail: Sets the index key's initial value. The key is used when
selecting an
  index to store the events.
```

```
* Defaults to "main" (or whatever you have set as your default index).

source = <string>
* Sets the source key/field for events from this input.
* NOTE: Overriding the source key is generally not recommended.
Typically, the
  input layer will provide a more accurate string to aid problem
  analysis and investigation, accurately recording the file from which
the data
  was retrieved.  Please consider use of source types, tagging, and
search
  wildcards before overriding this value.
* Detail: Sets the source key's initial value. The key is used during
  parsing/indexing, in particular to set the source field during
  indexing.  It is also the source field used at search time.
* As a convenience, the chosen string is prepended with 'source::'.
* WARNING: Do not quote the <string> value: source=foo, not
source="foo".
* Defaults to the input file path.

sourcetype = <string>
* Sets the sourcetype key/field for events from this input.
* Primarily used to explicitly declare the source type for this data, as
  opposed to allowing it to be determined via automated methods.  This
is
  typically important both for searchability and for applying the
relevant
  configuration for this type of data during parsing and indexing.
* Detail: Sets the sourcetype key's initial value. The key is used
during
  parsing/indexing, in particular to set the source type field during
  indexing. It is also the source type field used at search time.
* As a convenience, the chosen string is prepended with 'sourcetype::'.
* WARNING: Do not quote the <string> value: sourcetype=foo, not
sourcetype="foo".
* If unset, Splunk picks a source type based on various aspects of the
data.
  There is no hard-coded default.

queue = [parsingQueue|indexQueue]
* Specifies where the input processor should deposit the events it
reads.
* Set queue to "parsingQueue" to apply props.conf and other parsing
rules to
  your data. For more information about props.conf and rules for
timestamping
  and linebreaking, refer to props.conf and the online documentation at
  http://docs.splunk.com/Documentation.
* Set queue to "indexQueue" to send your data directly into the index.
* Defaults to parsingQueue.

# Pipeline Key defaulting.
```

```
* Pipeline keys in general can be defaulted in inputs stanzas.
* The list of user-available modifiable pipeline keys is described in
  transforms.conf.spec; see transforms.conf.spec for further
information on
  these keys.
* The currently-defined keys which are available literally in inputs
stanzas
  are as follows:

queue = <value>
_raw  = <value>
_meta = <value>
_time = <value>
* Inputs have special support for mapping host, source, sourcetype, and
index
  to their metadata names such as host -> Metadata:Host
* Defaulting these values is not recommended, and is
  generally only useful as a workaround to other product issues.
* Defaulting these keys in most cases will override the default behavior
of
  input processors; but this behavior is not guaranteed in all cases.
* Values defaulted here, as with all values provided by inputs, can be
  altered by transforms at parse-time.

# ***********
# This section contains options for routing data using inputs.conf
rather than
# outputs.conf.
# Note concerning routing via inputs.conf:
# This is a simplified set of routing options you can use as data comes
in.
# For more flexible options or details on configuring required or
optional
# settings, see outputs.conf.spec.

_TCP_ROUTING =
<tcpout_group_name>,<tcpout_group_name>,<tcpout_group_name>, ...
* Comma-separated list of tcpout group names.
* Using this, you can selectively forward the data to specific
indexer(s).
* Specify the tcpout group the forwarder should use when forwarding the
data.
  The tcpout group names are defined in outputs.conf with
  [tcpout:<tcpout_group_name>].
* Defaults to groups specified in "defaultGroup" in [tcpout] stanza in
  outputs.conf.
* To forward data from the "_internal" index, _TCP_ROUTING must
explicitly be
  set to either "*" or a specific splunktcp target group.

_SYSLOG_ROUTING =
```

```
<syslog_group_name>,<syslog_group_name>,<syslog_group_name>, ...
* Comma-separated list of syslog group names.
* Using this, you can selectively forward the data to specific
destinations as
  syslog events.
* Specify the syslog group to use when forwarding the data.
  The syslog group names are defined in outputs.conf with
  [syslog:<syslog_group_name>].
* Defaults to groups present in "defaultGroup" in [syslog] stanza in
  outputs.conf.
* The destination host must be configured in outputs.conf, using
  "server=[<ip>|<servername>]:<port>".

_INDEX_AND_FORWARD_ROUTING = <string>
* Only has effect if using selectiveIndexing feature in outputs.conf.
* If set for any input stanza, should cause all data coming from that
input
  stanza to be labeled with this setting.
* When selectiveIndexing is in use on a forwarder:
  * data without this label will not be indexed by that forwarder.
  * data with this label will be indexed in addition to any forwarding.
* This setting does not actually cause data to be forwarded or not
forwarded in
  any way, nor does it control where the data is forwarded in
multiple-forward
  path cases.
* Defaults to not present.
```

Blacklist

```
#*************
# Blacklist
#*************Blacklist

[blacklist:<path>]
* Protect files on the file system from being indexed or previewed.
* The input treats a file as blacklisted if the file starts with any of
the
  defined blacklisted <paths>.
* The preview endpoint will return and error when asked to preview a
  blacklisted file.
* The oneshot endpoint and command will also return an error.
* When a blacklisted file is monitored (monitor:// or batch://),
filestatus
  endpoint will show an error.
* For fschange with the 'sendFullEvent' option enabled, contents of
  blacklisted files will not be indexed.
```

Valid input types follow, along with their input-specific settings:

```
#*******
# Valid input types follow, along with their input-specific settings:
#*******Valid input types follow, along with their input-specific
settings:
```

MONITOR:

```
#*******
# MONITOR:
#*******MONITOR:

[monitor://<path>]
* This directs a file monitor input to watch all files in <path>.
* <path> can be an entire directory or a single file.
* You must specify the input type and then the path, so put three
slashes in
  your path if you are starting at the root on *nix systems (to include
the
  slash that indicates an absolute path).

# Additional settings:

host_regex = <regular expression>
* If specified, <regular expression> extracts host from the path to the
file
  for each input file.
    * Detail: This feature examines the source key; if source is set
      explicitly in the stanza, that string will be matched, not the
original
      filename.
* Specifically, the first group of the regex is used as the host.
* If the regex fails to match, the default "host =" setting is used.
* If host_regex and host_segment are both set, the input ignores
host_regex.
* Defaults to unset.

host_segment = <integer>
* If set to N, the Nth "/"-separated segment of the path is set as
host. If
  host_segment=3, for example, the third segment is used.
* If the value is not an integer or is less than 1, the default "host
="
  setting is used.
* Defaults to unset.
```

```
whitelist = <regular expression>
* If set, files from this input are monitored only if their path matches
the
  specified regex.
* Takes precedence over the deprecated _whitelist setting, which
functions
  the same way.

blacklist = <regular expression>
* If set, files from this input are NOT monitored if their path matches
the
  specified regex.
* Takes precedence over the deprecated _blacklist setting, which
functions
  the same way.

Note concerning wildcards and monitor:
* You can use wildcards to specify your input path for monitored input.
Use
  "..." for recursive directory matching and "*" for wildcard matching
in a
  single directory segment.
* "..." recurses through directories. This means that /foo/.../bar will
match
  foo/bar, foo/1/bar, foo/1/2/bar, etc.
* You can use multiple "..." specifications in a single input path. For
  example: /foo/.../bar/...
* The asterisk (*) matches anything in a single path segment; unlike
"...", it
  does not recurse. For example, /foo/*/bar matches the files /foo/bar,
  /foo/1/bar, /foo/2/bar, etc. However, it does not match /foo/1/2/bar.
  A second example: /foo/m*r/bar matches /foo/mr/bar, /foo/mir/bar,
  /foo/moor/bar, etc.
* You can combine "*" and "..." as needed: foo/.../bar/* matches any
file in
  the bar directory within the specified path.

crcSalt = <string>
* Use this setting to force the input to consume files that have
matching CRCs
  (cyclic redundancy checks).
    * (The input only performs CRC checks against, by default, the first
256
      bytes of a file. This behavior prevents the input from indexing
the same
      file twice, even though you may have renamed it -- as, for
example, with
      rolling log files. However, because the CRC is based on only the
first
      few lines of the file, it is possible for legitimately different
files
      to have matching CRCs, particularly if they have identical
```

```
headers.)
* If set, <string> is added to the CRC.
* If set to the literal string <SOURCE> (including the angle brackets),
the
  full directory path to the source file is added to the CRC. This
ensures
  that each file being monitored has a unique CRC.   When crcSalt is
invoked,
  it is usually set to <SOURCE>.
* Be cautious about using this setting with rolling log files; it could
lead
  to the log file being re-indexed after it has rolled.
* In many situations, initCrcLength can be used to achieve the same
goals.
* Defaults to empty.

initCrcLength = <integer>
* This setting adjusts how much of a file the input reads before trying
to
  identify whether it is a file that has already been seen. You might
want to
  adjust this if you have many files with common headers (comment
headers,
  long CSV headers, etc) and recurring filenames.
* CAUTION: Improper use of this setting will cause data to be
re-indexed.  You
  might want to consult with Splunk Support before adjusting this value
- the
  default is fine for most installations.
* Defaults to 256 (bytes).
* Must be in the range 256-1048576.

ignoreOlderThan = <nonnegative integer>[s|m|h|d]
* The monitor input will compare the modification time on files it
encounters
  with the current time.  If the time elapsed since the modification
time
  is greater than this setting, it will be placed on the ignore list.
* Files placed on the ignore list will not be checked again for any
  reason until the Splunk software restarts, or the file monitoring
subsystem
  is reconfigured.  This is true even if the file becomes newer again
at a
  later time.
  * Reconfigurations occur when changes are made to monitor or batch
    inputs via the UI or command line.
* Use IgnoreOlderThan to increase file monitoring performance when
  monitoring a directory hierarchy containing many unchanging older
  files, and when removing or blacklisting those files from the
monitoring
  location is not a reasonable option.
* Do NOT select a time that files you want to read could reach in
```

```
  age, even temporarily.  Take potential downtime into consideration!
  * Suggested value: 14d, which means 2 weeks
  * For example, a time window in significant numbers of days or small
    numbers of weeks are probably reasonable choices.
  * If you need a time window in small numbers of days or hours,
    there are other approaches to consider for performant monitoring
    beyond the scope of this one setting.
* NOTE: Most modern Windows file access APIs do not update file
  modification time while the file is open and being actively written
to.
  Windows delays updating modification time until the file is closed.
  Therefore you might have to choose a larger time window on Windows
  hosts where files may be open for long time periods.
* Value must be: <number><unit>. For example, "7d" indicates one week.
* Valid units are "d" (days), "h" (hours), "m" (minutes), and "s"
  (seconds).
* Defaults to unset, meaning there is no threshold and no files are
  ignored for modification time reasons.

followTail = [0|1]
* WARNING: Use of followTail should be considered an advanced
administrative
  action.
* Treat this setting as an 'action':
  * Enable this setting and start the Splunk software.
  * Wait enough time for the input to identify the related files.
  * Disable the setting and restart.
* DO NOT leave followTail enabled in an ongoing fashion.
* Do not use followTail for rolling log files (log files that get
renamed as
  they age), or files whose names or paths vary.
* You can use this to force the input to skip past all current data for
a
  given stanza.
  * In more detail: this is intended to mean that if you start the
monitor
    with a stanza configured this way, all data in the file at the time
it is
    first encountered will not be read. Only data that arrives after the
first
    encounter time will be read.
  * This can be used to "skip over" data from old log files, or old
portions of
    log files, to get started on current data right away.
* If set to 1, monitoring starts at the end of the file (like tail -f).
* If set to 0, monitoring starts at the beginning of the file.
* Defaults to 0.

alwaysOpenFile = [0|1]
* Opens a file to check whether it has already been indexed, by skipping
the
  modification time/size checks.
```

```
* Only useful for files that do not update modification time or size.
* Only known to be needed when monitoring files on Windows, mostly for
  Internet Information Server logs.
* This flag should only be used as a last resort, as it increases load
and
  slows down indexing.
* Defaults to 0.

time_before_close = <integer>
* Modification time delta required before the file monitor can close a
file on
  EOF.
* Tells the system not to close files that have been updated in past
<integer>
  seconds.
* Defaults to 3.

multiline_event_extra_waittime = [true|false]
* By default, the file monitor sends an event delimiter when:
  * It reaches EOF of a file it monitors and
  * Ihe last character it reads is a newline.
* In some cases, it takes time for all lines of a multiple-line event
to
  arrive.
* Set to true to delay sending an event delimiter until the time that
the
  file monitor closes the file, as defined by the 'time_before_close'
setting,
  to allow all event lines to arrive.
* Defaults to false.

recursive = [true|false]
* If false, the input will not monitor sub-directories that it finds
within
  a monitored directory.
* Defaults to true.

followSymlink = [true|false]
* Whether or not to follow any symbolic links within a monitored
directory.
* If set to false, the input ignores symbolic links found within a
monitored
  directory.
* If set to true, the input follows symbolic links and monitor files at
the
  symbolic link destination.
* Additionally, any whitelists or blacklists that the input stanza
defines
  also apply to files at the symbolic link's destination.
* Defaults to true.

_whitelist = ...
```

```
* This setting is deprecated.
* It is still honored, unless the 'whitelist' setting also exists.

_blacklist = ...
* This setting is deprecated.
* It is still honored, unless the 'blacklist' setting also exists.

# dedicatedFD = ...
* This setting has been removed.  It is no longer needed.
```

BATCH ("Upload a file" in Splunk Web):

```
#*************************************
# BATCH  ("Upload a file" in Splunk Web):
#*************************************BATCH  ("Upload a file" in
Splunk Web):

NOTE: Batch should only be used for large archives of historic data. If
you
want to continuously monitor a directory or index small archives, use
'monitor'
(see above). 'batch' reads in the file and indexes it, and then deletes
the
file on disk.

[batch://<path>]
* A one-time, destructive input of files in <path>.
* For continuous, non-destructive inputs of files, use 'monitor'
instead.

# Additional settings:

move_policy = sinkhole
* IMPORTANT: This setting is required. You *must* include
  "move_policy = sinkhole" when you define batch inputs.
* This setting causes the input to load the file destructively.
* Do not use the 'batch' input type for files you do not want to delete
after
  indexing.
* The "move_policy" setting exists for historical reasons, but remains
as an
  explicit double check.  As an administrator you must very explicitly
declare
  that you want the data in the monitored directory (and its
sub-directories) to
  be deleted after being read and indexed.

host_regex = see MONITOR, above.
host_segment = see MONITOR, above.
```

```
crcSalt = see MONITOR, above.

# IMPORTANT: 'batch' inputs do not use the following setting:
# source = <string>

followSymlink = [true|false]
* Works similarly to the same setting for monitor, but does not delete
files
  after following a symbolic link out of the monitored directory.

# The following settings work identically as for [monitor::] stanzas,
# documented above
host_regex = <regular expression>
host_segment = <integer>
crcSalt = <string>
recursive = [true|false]
whitelist = <regular expression>
blacklist = <regular expression>
initCrcLength = <integer>
```

TCP:

```
#*******
# TCP:
#*******TCP:

[tcp://<remote server>:<port>]
* Configures the input to listen on a specific TCP network port.
* If a <remote server> makes a connection to this instance, this stanza
is
  used to configure the input.
* If you do not specify <remote server>, this stanza matches all
connections
  on the specified port.
* Generates events with source set to tcp:portnumber, for example:
tcp:514
* If you do not specify a sourcetype, generates events with sourcetype
  set to tcp-raw.

# Additional settings:

connection_host = [ip|dns|none]
* "ip" sets the host to the IP address of the system sending the data.
* "dns" sets the host to the reverse DNS entry for the IP address of
the system
  sending the data.
* "none" leaves the host as specified in inputs.conf, typically the
splunk
  system hostname.
```

```
* Defaults to "dns".

queueSize = <integer>[KB|MB|GB]
* The maximum size of the in-memory input queue.
* Defaults to 500KB.

persistentQueueSize = <integer>[KB|MB|GB|TB]
* Maximum size of the persistent queue file.
* Defaults to 0 (no persistent queue).
* If set to some value other than 0, persistentQueueSize must be larger
than
  the in-memory queue size (as defined by the 'queueSize' setting in
  inputs.conf or 'maxSize' settings in [queue] stanzas in server.conf).
* Persistent queues can help prevent loss of transient data. For
information on
  persistent queues and how the 'queueSize' and 'persistentQueueSize'
settings
  interact, see the online documentation.
* Defaults to 0 (no persistent queue).

requireHeader = <bool>
* Require a header be present at the beginning of every stream.
* This header may be used to override indexing settings.
* Defaults to false.

listenOnIPv6 = <no | yes | only>
* Select whether the input listens on IPv4, IPv6, or both
* Set this to 'yes' to listen on both IPv4 and IPv6 protocols.
* Set to 'only' to listen on only the IPv6 protocol.
* If not present, the input uses the setting in the [general] stanza
  of server.conf.

acceptFrom = <network_acl> ...
* Lists a set of networks or addresses to accept connections from.
* Separate multiple rules with commas or spaces.
* Each rule can be in one of the following formats:
    1. A single IPv4 or IPv6 address (examples: "10.1.2.3", "fe80::4a3")
    2. A Classless Inter-Domain Routing (CIDR) block of addresses
       (examples: "10/8", "fe80:1234/32")
    3. A DNS name, possibly with a '*' used as a wildcard
       (examples: "myhost.example.com", "*.splunk.com")
    4. A single '*' which matches anything
* You can also prefix an entry with '!' to cause the rule to reject the
  connection. The input applies rules in order, and uses the first one
that
  matches.
  For example, "!10.1/16, *" allows connections from everywhere except
  the 10.1.*.* network.
* Defaults to "*" (accept from anywhere)

rawTcpDoneTimeout = <seconds>
* Specifies timeout value for sending Done-key.
```

```
* If a connection over this port remains idle for more than
  'rawTcpDoneTimeout' seconds after receiving data, it adds a Done-key.
This
  declares that the last event has been completely received.
* Defaults to 10 seconds.

[tcp:<port>]
* Configures the input listen on the specified TCP network port.
* This stanza is similar to [tcp://<remote server>:<port>], but listens
for
  connections to the specified port from any host.
* Generates events with a source of tcp:<port>.
* If you do not specify a sourcetype, generates events with a source
type of
  tcp-raw.
* This stanza supports the following settings:

connection_host = [ip|dns|none]
queueSize = <integer>[KB|MB|GB]
persistentQueueSize = <integer>[KB|MB|GB|TB]
requireHeader = <bool>
listenOnIPv6 = <no | yes | only>
acceptFrom = <network_acl> ...
rawTcpDoneTimeout = <seconds>
```

Data distribution:

```
#*******
# Data distribution:
#*******Data distribution:

# Global settings for splunktcp. Used on the receiving side for data
forwarded
# from a forwarder.

[splunktcp]
route = [has_key|absent_key:<key>:<queueName>;...]
* Settings for the light forwarder.
* The receiver sets these parameters automatically -- you DO NOT need to
set
  them.
* The property route is composed of rules delimited by ';' (semicolon).
* The receiver checks each incoming data payload via cooked tcp port
against
  the route rules.
* If a matching rule is found, the receiver sends the payload to the
specified
  <queueName>.
* If no matching rule is found, the receiver sends the payload to the
```

```
default
  queue specified by any queue= for this stanza. If no queue= key is set
in
  the stanza or globally, the events will be sent to the parsingQueue.

enableS2SHeartbeat = [true|false]
* This specifies the global keepalive setting for all splunktcp ports.
* This option is used to detect forwarders which might have become
unavailable
  due to network, firewall, or other problems.
* The receiver monitors each connection for presence of heartbeat, and
if the
  heartbeat is not seen for s2sHeartbeatTimeout seconds, it closes the
  connection.
* Defaults to true (heartbeat monitoring enabled).

s2sHeartbeatTimeout = <seconds>
* This specifies the global timeout value for monitoring heartbeats.
* The receiver closes a forwarder connection if it does not receive
  a heartbeat for 's2sHeartbeatTimeout' seconds.
* Defaults to 600 seconds (10 minutes).

inputShutdownTimeout = <seconds>
* Used during shutdown to minimize data loss when forwarders are
connected to a
  receiver.
* During shutdown, the tcp input processor waits for the specified
number of
  seconds and then closes any remaining open connections. If, however,
all
  connections close before the end of the timeout period, shutdown
proceeds
  immediately, without waiting for the timeout.

stopAcceptorAfterQBlock = <seconds>
* Specifies the time, in seconds, to wait before closing the splunktcp
port.
* If the receiver is unable to insert received data into the configured
queue
  for more than the specified number of seconds, it closes the splunktcp
port.
* This action prevents forwarders from establishing new connections to
this
  receiver.
* Forwarders that have an existing connection will notice the port is
closed
  upon test-connections and move to other receivers.
* Once the queue unblocks, and TCP Input can continue processing data,
the
  receiver starts listening on the port again.
* This setting should not be adjusted lightly as extreme values can
interact
```

```
  poorly with other defaults.
* Defaults to 300 (5 minutes).

listenOnIPv6 = no|yes|only
* Select whether this receiver listens on IPv4, IPv6, or both protocols.
* Set this to 'yes' to listen on both IPv4 and IPv6 protocols.
* Set to 'only' to listen on only the IPv6 protocol.
* If not present, the input uses the setting in the [general] stanza
  of server.conf.

acceptFrom = <network_acl> ...
* Lists a set of networks or IP addresses from which to accept
connections.
* Specify multiple rules with commas or spaces.
* Each rule can be in the following forms:
    1. A single IPv4 or IPv6 address (examples: "10.1.2.3", "fe80::4a3")
    2. A CIDR block of addresses (examples: "10/8", "fe80:1234/32")
    3. A DNS name, possibly with a '*' used as a wildcard (examples:
       "myhost.example.com", "*.splunk.com")
    4. A single '*', which matches anything.
* You can also prefix an entry with '!' to cause the rule to reject the
  connection. The input applies rules in order, and uses the first one
  that matches. For example, "!10.1/16, *" allows connections from
everywhere
  except the 10.1.*.* network.
* Defaults to "*" (accept from anywhere)

negotiateNewProtocol = [true|false]
* If set to true, lets forwarders that connect to this indexer (or
  specific port) send data using the new forwarder protocol.
* If set to false, denies the use of the new forwarder protocol during
  connection negotiation.
* Defaults to true.

concurrentChannelLimit = <unsigned integer>
* Each forwarder that connects to this indexer may use up to
  <concurrentChannelLimit> unique channel codes.
* In other words, each forwarder may have up to
<concurrentChannelLimit>
  channels in flight concurrently.
* The receiver closes a forwarder connection if a forwarder attempts to
exceed
  this value.
* This setting only applies when the new forwarder protocol is in use.
* Defaults to 300.

# Forwarder-specific settings for splunktcp.

[splunktcp://[<remote server>]:<port>]
* Receivers use this input stanza.
* This is the same as the [tcp://] stanza, except the remote server is
assumed
```

```
  to be a Splunk instance, most likely a forwarder.
* <remote server> is optional. If you specify it, the receiver only
listen for
  data from <remote server>.
  * Use of <remote server is not recommended. Use the 'acceptFrom'
setting,
    which supersedes this setting.

connection_host = [ip|dns|none]
* For splunktcp, the host or connection_host will be used if the remote
Splunk
  instance does not set a host, or if the host is set to
"<host>::<localhost>".
* "ip" sets the host to the IP address of the system sending the data.
* "dns" sets the host to the reverse DNS entry for IP address of the
system
  sending the data.
* "none" leaves the host as specified in inputs.conf, typically the
splunk
  system hostname.
* Defaults to "ip".

compressed = [true|false]
* Specifies whether the receiver receives compressed data.
* Applies to non-SSL receiving only. There is no compression setting
required
  for SSL.
* If set to true, the forwarder port(s) should also have compression
turned on;
  otherwise, the receiver rejects the connection.
* Defaults to false.

enableS2SHeartbeat = [true|false]
* This specifies the keepalive setting for the splunktcp port.
* This option is used to detect forwarders which might have become
unavailable
  due to network, firewall, or other problems.
* The receiver monitors the connection for presence of heartbeat, and if
it
  does not see the heartbeat s2sHeartbeatTimeout seconds, it closes the
  connection.
* This overrides the default value specified at the global [splunktcp]
stanza.
* Defaults to true (heartbeat monitoring enabled).

s2sHeartbeatTimeout = <seconds>
* This specifies the timeout value for monitoring heartbeats.
* The receiver closes the forwarder connection if it does not see a
heartbeat
  for 's2sHeartbeatTimeout' seconds.
* This overrides the default value specified at the global [splunktcp]
stanza.
```

```
* Defaults to 600 seconds (10 minutes).

queueSize = <integer>[KB|MB|GB]
* The maximum size of the in-memory input queue.
* Defaults to 500KB.

negotiateNewProtocol = [true|false]
* See the description for [splunktcp].

concurrentChannelLimit = <unsigned integer>
* See the description for [splunktcp].

[splunktcp:<port>]
* This input stanza is the same as [splunktcp://[<remote
server>]:<port>], but
  does not have a remote server restriction.
* Please see documentation for [splunktcp://[<remote server>]:<port>]
for
  following supported settings:

connection_host = [ip|dns|none]
compressed = [true|false]
enableS2SHeartbeat = [true|false]
s2sHeartbeatTimeout = <seconds>
queueSize = <integer>[KB|MB|GB]
negotiateNewProtocol = [true|false]
concurrentChannelLimit = <unsigned integer>

# Access control settings.
[splunktcptoken://<token name>]
* This stanza is optional.
* Use this stanza to specify forwarders from which to accept data.
* You must configure a token on the receiver, then configure the same
  token on forwarders.
* The receiver discards data from forwarders that do not have the
  token configured.
* This setting is enabled for all receiving ports.

token = <string>
* Value of token.

# SSL settings for data distribution:

[splunktcp-ssl:<port>]
* Use this stanza type if you are receiving encrypted, parsed data from
a
  forwarder.
* Set <port> to the port on which the forwarder sends the encrypted
data.
* Forwarder settings are set in outputs.conf on the forwarder.
* Compression for SSL is enabled by default. On the forwarder you can
still
```

```
  specify compression with the 'useClientSSLCompression' setting in
  outputs.conf.
* The 'compressed' setting is used for non-SSL connections. However, if
you
  still specify 'compressed' for SSL, ensure that the 'compressed'
setting is
  the same as on the forwarder, as splunktcp protocol expects the same
  'compressed' setting from forwarders.

connection_host = [ip|dns|none]
* For splunktcp, the host or connection_host will be used if the remote
Splunk
  instance does not set a host, or if the host is set to
"<host>::<localhost>".
* "ip" sets the host to the IP address of the system sending the data.
* "dns" sets the host to the reverse DNS entry for IP address of the
system
  sending the data.
* "none" leaves the host as specified in inputs.conf, typically the
splunk
  system hostname.
* Defaults to "ip".

compressed = [true|false]
* See comments for [splunktcp:<port>].

enableS2SHeartbeat = true|false
* See comments for [splunktcp:<port>].

s2sHeartbeatTimeout = <seconds>
* See comments for [splunktcp:<port>].

listenOnIPv6 = no|yes|only
* Select whether this receiver listens on IPv4, IPv6, or both protocols.
* Set this to 'yes' to listen on both IPv4 and IPv6 protocols.
* Set to 'only' to listen on only the IPv6 protocol.
* If not present, the input uses the setting in the [general] stanza
  of server.conf.

acceptFrom = <network_acl> ...
* Lists a set of networks or IP addresses from which to accept
connections.
* Specify multiple rules with commas or spaces.
* Each rule can be in the following forms:
    1. A single IPv4 or IPv6 address (examples: "10.1.2.3", "fe80::4a3")
    2. A CIDR block of addresses (examples: "10/8", "fe80:1234/32")
    3. A DNS name, possibly with a '*' used as a wildcard (examples:
       "myhost.example.com", "*.splunk.com")
    4. A single '*', which matches anything.
* You can also prefix an entry with '!' to cause the rule to reject the
  connection. The input applies rules in order, and uses the first one
that
```

```
  matches. For example, "!10.1/16, *" allows connections from everywhere
except
  the 10.1.*.* network.
* Defaults to "*" (accept from anywhere)

negotiateNewProtocol = [true|false]
* See comments for [splunktcp].

concurrentChannelLimit = <unsigned integer>
* See comments for [splunktcp].

# To specify global ssl settings, that are applicable for all ports, add
the
# settings to the SSL stanza.
# Specify any ssl setting that deviates from the global setting here.
# For a detailed description of each ssl setting, refer to the [SSL]
stanza.

serverCert = <path>
sslPassword = <password>
rootCA = <path>
requireClientCert = <bool>
sslVersions = <string>
cipherSuite = <cipher suite string>
ecdhCurves = <comma separated list of ec curves>
dhFile = <path>
allowSslRenegotiation = true|false
sslQuietShutdown = [true|false]
sslCommonNameToCheck = <commonName1>, <commonName2>, ...
sslAltNameToCheck = <alternateName1>, <alternateName2>, ...

[tcp-ssl:<port>]
* Use this stanza type if you are receiving encrypted, unparsed data
from a
  forwarder or third-party system.
* Set <port> to the port on which the forwarder/third-party system is
sending
  unparsed, encrypted data.

listenOnIPv6 = <no | yes | only>
* Select whether the receiver listens on IPv4, IPv6, or both protocols.
* Set this to 'yes' to listen on both IPv4 and IPv6 protocols.
* Set to 'only' to listen on only the IPv6 protocol.
* If not present, the receiver uses the setting in the [general] stanza
  of server.conf.

acceptFrom = <network_acl> ...
* Lists a set of networks or IP addresses from which to accept
connections.
* Specify multiple rules with commas or spaces.
* Each rule can be in the following forms:
    1. A single IPv4 or IPv6 address (examples: "10.1.2.3", "fe80::4a3")
```

```
    2. A CIDR block of addresses (examples: "10/8", "fe80:1234/32")
    3. A DNS name, possibly with a '*' used as a wildcard (examples:
       "myhost.example.com", "*.splunk.com")
    4. A single '*', which matches anything.
* You can also prefix an entry with '!' to cause the rule to reject the
  connection. The input applies rules in order, and uses the first one
that
  matches. For example, "!10.1/16, *" allows connections from everywhere
except
  the 10.1.*.* network.
* Defaults to "*" (accept from anywhere)

[SSL]
* Set the following specifications for receiving Secure Sockets Layer
(SSL)
  communication underneath this stanza name.

serverCert = <path>
* The full path to the server certificate Privacy-Enhanced Mail (PEM)
format
  file.
* PEM is the most common text-based storage format for SSL certificate
files.
* There is no default.

sslPassword = <password>
* Server certificate password, if any.
* Initially set to plain-text password.
* Upon first use, the input encrypts and rewrites the password to
  $SPLUNK_HOME/etc/system/local/inputs.conf.

password = <password>
* This setting is DEPRECATED.
* Do not use this setting. Use the 'sslPassword' setting instead.

rootCA = <path>
* This setting is DEPRECATED.
* Do not use this setting. Use 'server.conf/[sslConfig]/sslRootCAPath'
instead.
* Used only if 'sslRootCAPath' is unset.

requireClientCert = <bool>
* Determines whether a client must present an SSL certificate to
authenticate.
* Full path to the root CA (Certificate Authority) certificate store.
* The <path> must refer to a PEM format file containing one or more
root CA
  certificates concatenated together.
* Defaults to false.

sslVersions = <string>
* A comma-separated list of SSL versions to support.
```

```
* The versions available are "ssl3", "tls1.0", "tls1.1", and "tls1.2"
* The special version "*" selects all supported versions. The version
"tls"
  selects all versions "tls1.0" or newer.
* To remove a version from the list, prefix it with "-".
* SSLv2 is always disabled. You can specify "-ssl2" in the version list,
but
  doing so has no effect.
* When configured in Federal Information Processing Standard (FIPS)
mode, the
  "ssl3" version is always disabled, regardless of this configuration.
* Defaults to "*,-ssl2".  (anything newer than SSLv2)

supportSSLV3Only = <bool>
* This setting is DEPRECATED.
* SSLv2 is now always disabled.
* Use the "sslVersions" setting to set the list of supported SSL
versions.

cipherSuite = <cipher suite string>
* If set, uses the specified cipher string for the input processors.
* If not set, the default cipher string is used.
* Provided by OpenSSL. This is used to ensure that the server does not
  accept connections using weak encryption protocols.
* Must specify 'dhFile' to enable any Diffie-Hellman ciphers.

ecdhCurveName = <string>
* This setting is DEPRECATED.
* Use the 'ecdhCurves' setting instead.
* This setting specifies the Elliptic Curve Diffie-Hellman (ECDH) curve
to
  use for ECDH key negotiation.
* Splunk only supports named curves that have been specified by their
SHORT name.
* The list of valid named curves by their short/long names
  can be obtained by executing this command:
  $SPLUNK_HOME/bin/splunk cmd openssl ecparam -list_curves
* Default is empty string.

ecdhCurves = <comma separated list of ec curves>
* ECDH curves to use for ECDH key negotiation.
* The curves should be specified in the order of preference.
* The client sends these curves as a part of Client Hello.
* The server supports only the curves specified in the list.
* Splunk only supports named curves that have been specified by their
SHORT names.
  (see struct ASN1_OBJECT in asn1.h)
* The list of valid named curves by their short/long names can be
obtained
  by executing this command:
  $SPLUNK_HOME/bin/splunk cmd openssl ecparam -list_curves
* Default is empty string.
```

```
* Example setting: ecdhCurves = prime256v1,secp384r1,secp521r1

dhFile = <path>
* Full path to the Diffie-Hellman parameter file.
* DH group size should be no less than 2048 bits.
* This file is required in order to enable any Diffie-Hellman ciphers.
* Not set by default.

dhfile = <path>
* This setting is DEPRECATED.
* Use the 'dhFile' setting instead.

allowSslRenegotiation = true|false
* In the SSL protocol, a client may request renegotiation of the
connection
  settings from time to time.
* Setting this to false causes the server to reject all renegotiation
  attempts, which breaks the connection.
* This limits the amount of CPU a single TCP connection can use, but it
can
  cause connectivity problems, especially for long-lived connections.
* Defaults to true.

sslQuietShutdown = [true|false]
* Enables quiet shutdown mode in SSL.
* Defaults to false.

sslCommonNameToCheck = <commonName1>, <commonName2>, ...
* Check the common name of the client's certificate against this list of
names.
* If there is no match, assume that the Splunk instance is not
authenticated
  against this server.
* This setting is optional.
* Defaults to no common name checking.
* requireClientCert must be set to true for this setting to work.

sslAltNameToCheck = <alternateName1>, <alternateName2>, ...
* Check the alternate name of the client certificate against this list
of names.
* If there is no match, assume that the Splunk instance is not
authenticated
  against this server.
* This setting is optional.
* Defaults to no alternate name checking.
* For this setting to work, the 'requireClientCert'
  setting must be set to true.
```

UDP:

```
#*******
# UDP:
#*******UDP:

[udp://<remote server>:<port>]
* Similar to the [tcp://] stanza, except that this stanza causes the
Splunk
  instance to listen on a UDP port.
* Only one stanza per port number is currently supported.
* Configures the instance to listen on a specific port.
* If you specify <remote server>, the specified port only accepts data
  from that host.
* If <remote server> is empty - [udp://<port>] - the port accepts data
sent
  from any host.
  * The use of <remote server> is not recommended. Use the 'acceptFrom'
    setting, which supersedes this setting.
* Generates events with source set to udp:portnumber, for example:
udp:514
* If you do not specify a sourcetype, generates events with sourcetype
set
  to udp:portnumber.

# Additional settings:

connection_host = [ip|dns|none]
* "ip" sets the host to the IP address of the system sending the data.
* "dns" sets the host to the reverse DNS entry for IP address of the
system
  sending the data.
* "none" leaves the host as specified in inputs.conf, typically the
splunk
  system hostname.
* Defaults to "ip".

_rcvbuf = <integer>
* Specifies the receive buffer for the UDP port (in bytes).
* If you set the value to 0 or a negative number, the input ignores the
value.
* Note: If the default value is too large for an OS, the instance tries
to set
  the value to 1572864/2. If that value is also too large, the instance
  retries with 1572864/(2*2). It continues to retry by halving the value
until
  it succeeds.
* Defaults to 1,572,864.

no_priority_stripping = [true|false]
```

```
* Setting for receiving syslog data.
* If you set this setting to true, the instance does NOT strip the
<priority>
  syslog field from received events.
* NOTE: Do NOT set this setting if you want to strip <priority>.
* Default is false.

no_appending_timestamp = [true|false]
* Whether or not to append a timestamp and host to received events.
* If you set this setting to true, the instance does NOT append a
timestamp
  and host to received events.
* NOTE: Do NOT set this setting if you want to append timestamp and
host
  to received events.
* Default is false.

queueSize = <integer>[KB|MB|GB]
* Maximum size of the in-memory input queue.
* Defaults to 500KB.

persistentQueueSize = <integer>[KB|MB|GB|TB]
* Maximum size of the persistent queue file.
* Defaults to 0 (no persistent queue).
* If set to some value other than 0, persistentQueueSize must be larger
than
  the in-memory queue size (as defined by the 'queueSize' setting in
  inputs.conf or 'maxSize' settings in [queue] stanzas in server.conf).
* Persistent queues can help prevent loss of transient data. For
information on
  persistent queues and how the 'queueSize' and 'persistentQueueSize'
settings
  interact, see the online documentation.

listenOnIPv6 = <no | yes | only>
* Select whether the instance listens on the IPv4, IPv6, or both
protocols.
* Set this to 'yes' to listen on both IPv4 and IPv6 protocols.
* Set to 'only' to listen on only the IPv6 protocol.
* If not present, the input uses the setting in the [general] stanza
  of server.conf.

acceptFrom = <network_acl> ...
* Lists a set of networks or IP addresses from which to accept
connections.
* Specify multiple rules with commas or spaces.
* Each rule can be in the following forms:
    1. A single IPv4 or IPv6 address (examples: "10.1.2.3", "fe80::4a3")
    2. A CIDR block of addresses (examples: "10/8", "fe80:1234/32")
    3. A DNS name, possibly with a '*' used as a wildcard (examples:
       "myhost.example.com", "*.splunk.com")
    4. A single '*', which matches anything.
```

```
* You can also prefix an entry with '!' to cause the rule to reject the
  connection. The input applies rules in order, and uses the first one
that
  matches.
  For example, "!10.1/16, *" allows connections from everywhere except
  the 10.1.*.* network.
* Defaults to "*" (accept from anywhere)

[udp:<port>]
* This input stanza is the same as [udp://<remote server>:<port>], but
does
  not have a <remote server> restriction.
* See the documentation for [udp://<remote server>:<port>] to configure
  supported settings:

connection_host = [ip|dns|none]
_rcvbuf = <integer>
no_priority_stripping = [true|false]
no_appending_timestamp = [true|false]
queueSize = <integer>[KB|MB|GB]
persistentQueueSize = <integer>[KB|MB|GB|TB]
listenOnIPv6 = <no | yes | only>
acceptFrom = <network_acl> ...
```

FIFO (First In, First Out queue):

```
#*******
# FIFO (First In, First Out queue):
#*******FIFO (First In, First Out queue):

[fifo://<path>]
* This stanza configures the monitoring of a FIFO at the specified path.

queueSize = <integer>[KB|MB|GB]
* Maximum size of the in-memory input queue.
* Defaults to 500KB.

persistentQueueSize = <integer>[KB|MB|GB|TB]
* Maximum size of the persistent queue file.
* Defaults to 0 (no persistent queue).
* If set to some value other than 0, persistentQueueSize must be larger
than
  the in-memory queue size (as defined by the 'queueSize' setting in
  inputs.conf or 'maxSize' settings in [queue] stanzas in server.conf).
* Persistent queues can help prevent loss of transient data. For
information on
  persistent queues and how the 'queueSize' and 'persistentQueueSize'
settings
  interact, see the online documentation.
```

Scripted Input:

```
#*******
# Scripted Input:
#*******Scripted Input:

[script://<cmd>]
* Runs <cmd> at a configured interval (see below) and indexes the output
  that <cmd> returns.
* The <cmd> must reside in one of the following directories:
  * $SPLUNK_HOME/etc/system/bin/
  * $SPLUNK_HOME/etc/apps/$YOUR_APP/bin/
  * $SPLUNK_HOME/bin/scripts/
* The path to <cmd> can be an absolute path, make use of an environment
  variable such as $SPLUNK_HOME, or use the special pattern of an
initial '.'
  as the first directory to indicate a location inside the current app.
* The '.' specification must be followed by a platform-specific
directory
  separator.
  * For example, on UNIX:
        [script://./bin/my_script.sh]
    Or on Windows:
        [script://.\bin\my_program.exe]
    This '.' pattern is strongly recommended for app developers, and
necessary
    for operation in search head pooling environments.
* <cmd> can also be a path to a file that ends with a ".path" suffix. A
file
  with this suffix is a special type of pointer file that points to a
command
  to be run. Although the pointer file is bound by the same location
  restrictions mentioned above, the command referenced inside it can
reside
  anywhere on the file system. The .path file must contain exactly one
line:
  the path to the command to run, optionally followed by command-line
  arguments. The file can contain additional empty lines and lines that
begin
  with '#'. The input ignores these lines.

interval = [<number>|<cron schedule>]
* How often to run the specified command (in seconds), or a valid cron
  schedule.
* NOTE: when you specify a cron schedule, the input does not run the
  script on start-up.
* If you specify the interval as a number, it may have a fractional
  component; e.g., 3.14
```

```
* The cron implementation for data inputs does not currently support
names
  of months or days.
* Defaults to 60.0 seconds.
* The special value 0 forces this scripted input to be run continuously;
  that is, as soon as the script exits, the input restarts it.
* The special value -1 causes the scripted input to run once on
start-up.

passAuth = <username>
* User to run the script as.
* If you provide a username, the instance generates an auth token for
that
  user and passes it to the script via stdin.

queueSize = <integer>[KB|MB|GB]
* Maximum size of the in-memory input queue.
* Defaults to 500KB.

persistentQueueSize = <integer>[KB|MB|GB|TB]
* Maximum size of the persistent queue file.
* Defaults to 0 (no persistent queue).
* If set to some value other than 0, persistentQueueSize must be larger
than
  the in-memory queue size (as defined by the 'queueSize' setting in
  inputs.conf or 'maxSize' settings in [queue] stanzas in server.conf).
* Persistent queues can help prevent loss of transient data. For
information on
  persistent queues and how the 'queueSize' and 'persistentQueueSize'
settings
  interact, see the online documentation.

index = <index name>
* The index where the input sends the data.
* Note: this parameter will be passed as a command-line argument to
<cmd> in
  the format: -index <index name>.
  If the script does not need the index info, it can ignore this
argument.
* If you do not specify an index, the script uses the default index.

send_index_as_argument_for_path = [true|false]
* Whether or not to pass the index as an argument when specified for
  stanzas that begin with 'script://'
* When you set this setting to true, the script passes the argument as
  '-index <index name>'.
* To avoid passing the index as a command line argument, set this to
false.
* Defaults to true.

start_by_shell = [true|false]
* Whether or not to run the specified command through the operating
```

```
system
  shell or command prompt.
* If you set this setting to true, the host operating system runs the
  specified command through the OS shell ("/bin/sh -c" on UNIX,
  "cmd.exe /c" on Windows.)
* If you set the setting to false, the input runs the program directly
  without attempting to expand shell metacharacters.
* On Unix hosts, defaults to true.
* On Windows hosts defaults to false.
* You might want to explicitly set the setting to false for scripts
  that you know do not need UNIX shell metacharacter expansion. This is
  a Splunk best practice.
```

File system change monitor (fschange monitor)

```
#*******
# File system change monitor (fschange monitor)
#*******File system change monitor (fschange monitor)
#
# The file system change monitor has been deprecated as of Splunk
Enterprise
# version 5.0 and might be removed in a future version of the product.
#
# You cannot simultaneously monitor a directory with both the 'fschange'
# and 'monitor' stanza types.

[fschange:<path>]
* Monitors changes (such as additions, updates, and deletions) to this
  directory and any of its sub-directories.
* <path> is the direct path. Do not preface it with '//' like with
  other inputs.
* Sends an event for every change.

# Additional settings:
# NOTE: The 'fschange' stanza type does not use the same settings as
# other input types. It uses only the following settings:

index = <index name>
* The index where the input sends the data.
* Defaults to _audit, unless you either do not set the 'signedaudit'
  setting, or set 'signedaudit' to false.
* If you set 'signedaudit' to false, events go into the default index.

signedaudit = [true|false]
* Whether or not to send cryptographically signed add/update/delete
events.
* If you set this setting to true, the input does the following to
  events that it generates:
  * Puts the events in the _audit index.
```

```
  * Sets the event sourcetype to 'audittrail'
* If you set the setting to false, the input:
  * Places events in the default index.
  * Sets the sourcetype to whatever you specify (or "fs_notification"
    by default).
* You must set 'signedaudit' to false if you want to set the index for
  fschange events.
* You must also enable auditing in audit.conf.
* Defaults to false.

filters = <filter1>,<filter2>,...
* Each filter is applied left to right for each file or directory
  found during the monitor poll cycle.
* See the "File System Monitoring Filters" section below for help
  on how to define a fschange filter.

recurse = [true|false]
* Whether or not the fschange input should look through all
sub-directories
  for changes to files in a directory.
* If you set this setting to true, the input recurses through
  sub-directories within the directory specified in [fschange].
* Defaults to true.

followLinks = [true|false]
* Whether or not the fschange input should follow any symbolic
  links it encounters.
* If you set this setting to true, the input follows symbolic links.
* Do not set this setting to true unless you can confirm that
  doing so will not create a file system loop (For example, in
  Directory A, symbolic link B points back to Directory A.)
* Defaults to false.

pollPeriod = <integer>
* How often, in seconds, to check a directory for changes.
* Defaults to 3600 seconds (1 hour).

hashMaxSize = <integer>
* Calculate a SHA256 hash for every file that is less than or equal to
  <integer> bytes.
* The input uses this hash as an additional method for detecting changes
to the
  file/directory.
* Defaults to -1 (disabled).

fullEvent = [true|false]
* Whether or not to send the full event if the input detects an add or
  update change.
* Set to true to send the full event if an add or update change is
detected.
* Further qualified by the 'sendEventMaxSize' setting.
* Defaults to false.
```

```
sendEventMaxSize  = <integer>
* Limits the size of event data that the fschange input sends.
* Only send the full event if the size of the event is less than or
equal to
  <integer> bytes.
* This limits the size of indexed file data.
* Defaults to -1, which is unlimited.

sourcetype = <string>
* Set the source type for events from this input.
* The input automatically prepends "sourcetype=" to <string>.
* Defaults to "audittrail" if you set the 'signedaudit' setting to true.
* Defaults to "fs_notification" if you set the 'signedaudit' setting to
false.

host = <string>
* Set the host name for events from this input.
* Defaults to whatever host sent the event.

filesPerDelay = <integer>
* The number of files that the fschange input processes between
processing
  delays, as specified by the 'delayInMills' setting.
* After a delay of 'delayInMills' milliseconds, the fschange input
processes
  <integer> files, then waits 'delayInMills' milliseconds again before
  repeating this process.
* This is used to throttle file system monitoring so it consumes less
CPU.
* Defaults to 10.

delayInMills = <integer>
* The delay, in milliseconds, that the fschange input waits prior to
  processing 'filesPerDelay' files.
* After a delay of 'delayInMills' milliseconds, the fschange input
processes
  <integer> files, then waits 'delayInMills' milliseconds again before
  repeating this process.
* This is used to throttle file system monitoring so it consumes less
CPU.
* Defaults to 100.
```

File system monitoring filters:

```
#*******
# File system monitoring filters:
#*******File system monitoring filters:
```

```
[filter:<filtertype>:<filtername>]
* Defines a filter of type <filtertype> and names it <filtername>.
* <filtertype>:
  * Filter types are either 'blacklist' or 'whitelist.'
  * A whitelist filter processes all file names that match the
    regular expression list that you define within the stanza.
  * A blacklist filter skips all file names that match the
    regular expression list.
* <filtername>
  * The fschange input uses filter names that you specify with
    the 'filters' setting for a given fschange stanza.
  * You can specify multiple filters buy separating them with commas.

regex<integer> = <regex>
* Blacklist and whitelist filters can include a set of regular
expressions.
* The name of each regex MUST be 'regex<integer>', where <integer>
  starts at 1 and increments.
* The input applies each regular expression in numeric order:
  regex1=<regex>
  regex2=<regex>
  ...
```

http: (HTTP Event Collector)

```
#*******
# http: (HTTP Event Collector)
#*******http: (HTTP Event Collector)

# Global settings for the HTTP Event Collector (HEC) Input.

[http]
port = <number>
* The event collector data endpoint server port.
* Defaults to 8088.

disabled = [0|1]
* Whether or not the event collector input is active.
* Set this setting to 1 to disable the input, and 0 to enable it.
* Defaults to 1 (disabled).

outputgroup = <string>
* The name of the output group that the event collector forwards data
to.
* Defaults to empty string.

useDeploymentServer = [0|1]
* Whether or not the event collector input should write its
```

```
configuration to
  a deployment server repository.
* When you set this setting to 1 (enabled), the input writes its
  configuration to the directory that you specify with the
  'repositoryLocation' setting in serverclass.conf.
* You must copy the full contents of the splunk_httpinput app directory
  to this directory for the configuration to work.
* When disabled, the input writes its configuration to
  $SPLUNK_HOME/etc/apps by default.
* Defaults to 0 (disabled).

index = <string>
* The default index to use.
* Defaults to the "default" index.

sourcetype = <string>
* The default source type for the events.
* If you do not specify a sourcetype, the input does not set a
sourcetype
  for events it generates.

enableSSL = [0|1]
* Whether or not to use SSL for the event collector endpoint server.
* HEC shares SSL settings with the Splunk management server and cannot
have
  'enableSSL' set to true when the Splunk management server has SSL
disabled.
* Defaults to 0 (enabled).

dedicatedIoThreads = <number>
* Defines the number of dedicated input/output threads in the event
collector
  input.
* Defaults to 0 (The input uses a single thread).

maxSockets = <int>
* The number of simultaneous HTTP connections that the event collector
input
  accepts simultaneously.
* Set this setting to constrain resource usage.
* If you set this setting to 0, the input automatically sets it to
  one third of the maximum allowable open files on the host.
* If this number is less than 50, the input sets it to 50. If this
number
  is  greater than 400000, the input sets it to 400000.
* If this number is negative, the input does not enforce a limit on
  connections.
* Defaults to 0.

maxThreads = <int>
* The number of threads that can be used by active HTTP transactions.
* Set this to constrain resource usage.
```

```
* If you set this setting to 0, the input automatically sets the limit
to
  one third of the maximum allowable threads on the host.
* If this number is less than 20, the input sets it to 20. If this
number is
  greater than 150000, the input sets it to 150000.
* If the 'maxSockets' setting has a positive value and 'maxThreads'
  is greater than 'maxSockets', then the input sets 'maxThreads' to be
equal
  to 'maxSockets'.
* If set to a negative number, the input does not enforce a limit on
threads.
* Defaults to 0.

serverCert = <path>
* The full path to the server certificate PEM format file.
* The same file may also contain a private key.
* Default is $SPLUNK_HOME/etc/auth/server.pem.
* The Splunk software automatically generates certificates when it first
  starts.
* You may replace the auto-generated certificate with your own
certificate.

sslKeysfile = <filename>
* This setting is DEPRECATED.
* Use the 'serverCert' setting instead.
* File is in the directory specified by 'caPath' (see below).
* Defaults to server.pem.

sslPassword = <password>
* The server certificate password.
* Initially set to plain-text password.
* Upon first use, it will be encrypted and rewritten.
* Defaults to "password".

sslKeysfilePassword = <password>
* This setting is DEPRECATED.
* Use the 'sslPassword' setting instead.

caCertFile = <filename>
* This setting is DEPRECATED.
* Use the 'server.conf/[sslConfig]/sslRootCAPath' setting instead.
* Used only if you do not set the 'sslRootCAPath' setting.
* Specifies the file name (relative to 'caPath') of the CA
  (Certificate Authority) certificate PEM format file containing one or
  more certificates concatenated together.
* Defaults to cacert.pem.

caPath = <path>
* This setting is DEPRECATED.
* Use absolute paths for all certificate files.
* If certificate files given by other settings in this stanza are not
```

```
absolute
  paths, then they will be relative to this path.
* Defaults to $SPLUNK_HOME/etc/auth.

sslVersions = <versions_list>
* A comma-separated list of SSL versions to support.
* The versions available are "ssl3", "tls1.0", "tls1.1", and "tls1.2"
* The special version "*" selects all supported versions. The version
"tls"
  selects all versions "tls1.0" or newer.
* To remove a version from the list, prefix it with "-".
* SSLv2 is always disabled. You can specify "-ssl2" in the version list,
but
  doing so has no effect.
* When configured in Federal Information Processing Standard (FIPS)
mode, the
  "ssl3" version is always disabled, regardless of this configuration.
* Defaults to "*,-ssl2".  (anything newer than SSLv2)

cipherSuite = <cipher suite string>
* The cipher string to use for the HTTP server.
* Use this setting to ensure that the server does not accept
connections using
  weak encryption protocols.
* If you set this setting, the input uses the specified cipher string
for
  the HTTP server.
* If you do not set the setting, the input uses the default cipher
  string that OpenSSL provides.

listenOnIPv6 = no|yes|only
* Select whether this input listens on IPv4, IPv6, or both.
* Set this to 'yes' to listen on both IPv4 and IPv6 protocols.
* Set to 'only' to listen on only the IPv6 protocol.
* If not present, the input uses the setting in the [general] stanza
  of server.conf.

acceptFrom = <network_acl> ...
* Lists a set of networks or IP addresses from which to accept
connections.
* Specify multiple rules with commas or spaces.
* Each rule can be in the following forms:
    1. A single IPv4 or IPv6 address (examples: "10.1.2.3", "fe80::4a3")
    2. A CIDR block of addresses (examples: "10/8", "fe80:1234/32")
    3. A DNS name, possibly with a '*' used as a wildcard (examples:
       "myhost.example.com", "*.splunk.com")
    4. A single '*', which matches anything.
* You can also prefix an entry with '!' to cause the rule to reject the
  connection. The input applies rules in order, and uses the first one
that
  matches. For example, "!10.1/16, *" allows connections from everywhere
except
```

```
  the 10.1.*.* network.
* Defaults to "*" (accept from anywhere)

requireClientCert = <bool>
* Requires that any client connecting to the HEC port has a certificate
that
  can be validated by the certificate authority specified in the
  'caCertFile' setting.
* Defaults to false.

ecdhCurveName = <string>
* This setting is DEPRECATED.
* Use the 'ecdhCurves' setting instead.
* This setting specifies the ECDH curve to use for ECDH key negotiation.
* Splunk only supports named curves that have been specified by their
SHORT name.
* The list of valid named curves by their short/long names
  can be obtained by executing this command:
  $SPLUNK_HOME/bin/splunk cmd openssl ecparam -list_curves
* Default is empty string.

ecdhCurves = <comma separated list of ec curves>
* ECDH curves to use for ECDH key negotiation.
* The curves should be specified in the order of preference.
* The client sends these curves as a part of Client Hello.
* The server supports only the curves specified in the list.
* Splunk only supports named curves that have been specified by their
SHORT names.
  (see struct ASN1_OBJECT in asn1.h)
* The list of valid named curves by their short/long names can be
obtained
  by executing this command:
  $SPLUNK_HOME/bin/splunk cmd openssl ecparam -list_curves
* Default is empty string.
* Example setting: ecdhCurves = prime256v1,secp384r1,secp521r1

crossOriginSharingPolicy = <origin_acl> ...
* List of the HTTP Origins for which to return Access-Control-Allow-*
(CORS)
  headers.
* These headers tell browsers that we trust web applications at those
sites
  to make requests to the REST interface.
* The origin is passed as a URL without a path component (for example
  "https://app.example.com:8000").
* This setting can take a list of acceptable origins, separated
  by spaces and/or commas.
* Each origin can also contain wildcards for any part.  Examples:
    *://app.example.com:*  (either HTTP or HTTPS on any port)
    https://*.example.com  (any host under example.com, including
example.com itself).
* An address can be prefixed with a '!' to negate the match, with
```

```
  the first matching origin taking precedence.  For example,
  "!*://evil.example.com:* *://*.example.com:*" to not avoid
  matching one host in a domain.
* A single "*" can also be used to match all origins.
* By default, the list is empty.

forceHttp10 = auto|never|always
* Whether or not the REST HTTP server forces clients that connect
  to it to use the HTTP 1.0 specification for web communications.
* When set to "always", the REST HTTP server does not use some
  HTTP 1.1 features such as persistent connections or chunked
  transfer encoding.
* When set to "auto" it does this only if the client did not send
  a User-Agent header, or if the user agent is known to have bugs
  in its support of HTTP/1.1.
* When set to "never" it always allows HTTP 1.1, even to
  clients it suspects may be buggy.
* Defaults to "auto".

sslCommonNameToCheck = <commonName1>, <commonName2>, ...
* If you set this setting and also set 'requireClientCert' to true,
  splunkd limits most inbound HTTPS connections to hosts that use
  a cert with one of the listed common names.
* The most important scenario is distributed search.
* This feature does not work with the deployment server and client
  communication over SSL.
* This setting is optional.
* Defaults to no common name checking.

sslAltNameToCheck = <alternateName1>, <alternateName2>, ...
* If you set this setting and also set 'requireClientCert' to true,
  splunkd can verify certificates that have a so-called
  "Subject Alternate Name" that matches any of the alternate
  names in this list.
  * Subject Alternate Names are effectively extended descriptive
    fields in SSL certs beyond the commonName. A common practice for
    HTTPS certs is to use these values to store additional valid
    hostnames or domains where the cert should be considered valid.
* Accepts a comma-separated list of Subject Alternate Names to consider
  valid.
* Items in this list are never validated against the SSL Common Name.
* This feature does not work with the deployment server and client
  communication over SSL.
* Optional.  Defaults to no alternate name checking

sendStrictTransportSecurityHeader = true|false
* If set to true, the REST interface sends a
"Strict-Transport-Security"
  header with all responses to requests made over SSL.
* This can help avoid a client being tricked later by a
Man-In-The-Middle
  attack to accept a non-SSL request. However, this requires a
```

```
commitment that
  no non-SSL web hosts will ever be run on this hostname on any port.
 For
  example, if Splunk Web is in default non-SSL mode this can break the
  ability of browser to connect to it.  Enable with caution.
* Defaults to false.

allowSslCompression = true|false
* If set to true, the server will allow clients to negotiate
  SSL-layer data compression.
* Defaults to true.

allowSslRenegotiation = true|false
* In the SSL protocol, a client may request renegotiation of the
connection
  settings from time to time.
* Setting this to false causes the server to reject all renegotiation
  attempts, which breaks the connection.
* This limits the amount of CPU a single TCP connection can use, but it
can
  cause connectivity problems, especially for long-lived connections.
* Defaults to true.

ackIdleCleanup = true|false
* If set to true, the server removes the ACK channels that are idle
  for 'maxIdleTime' seconds.
* Default to false.

maxIdleTime = <int>
* The maximum number of seconds the ACK channels are idle before they
are
  removed.
* Defaults to 600 seconds.

channel_cookie = <string>
* The name of the cookie to use when sending data with a specified
channel ID.
* The value of the cookie will be the channel sent. For example, if you
have
  set 'channel_cookie=foo' and sent a request with channel ID set to
'bar',
  then you will have a cookie in the response with the value 'foo=bar'.
* If no channel ID is present in the request, then no cookie will be
returned.
* This setting is to be used for load balancers (for example, AWS ELB)
that can
  only provide sticky sessions on cookie values and not general header
values.
* If no value is set (the default), then no cookie will be returned.
* Defaults to the empty string (no cookie).
```

HTTP Event Collector (HEC) - Local stanza for each token

```
#*******
# HTTP Event Collector (HEC) - Local stanza for each token
#*******HTTP Event Collector (HEC) - Local stanza for each token

[http://name]

token = <string>
* The value of the HEC token.

disabled = [0|1]
* Whether or not this token is active.
* Defaults to 0 (enabled).

description = <string>
* A human-readable description of this token.
* Defaults to empty string.

indexes = <string>
* The indexes the event for this token can go to.
* If you do not specify this value, the index list is empty, and any
index
  can be used.

index = <string>
* The default index to use for this token.
* Defaults to the default index.

sourcetype = <string>
* The default sourcetype to use if it is not specified in an event.
* Defaults to empty string.

outputgroup = <string>
* The name of the forwarding output group to send data to.
* Defaults to empty string.

queueSize = <integer>[KB|MB|GB]
* The maximum size of the in-memory input queue.
* Defaults to 500KB.

persistentQueueSize = <integer>[KB|MB|GB|TB]
* Maximum size of the persistent queue file.
* Defaults to 0 (no persistent queue).
* If set to some value other than 0, persistentQueueSize must be larger
than
  the in-memory queue size (as defined by the 'queueSize' setting in
  inputs.conf or 'maxSize' settings in [queue] stanzas in server.conf).
* Persistent queues can help prevent loss of transient data. For
information on
```

```
  persistent queues and how the 'queueSize' and 'persistentQueueSize'
settings
  interact, see the online documentation.

connection_host = [ip|dns|proxied_ip|none]
* Specify the host if an event doesn't have host set.
* "ip" sets the host to the IP address of the system sending the data.
* "dns" sets the host to the reverse DNS entry for IP address of the
system
  sending the data.
* "proxied_ip" checks whether an X-Forwarded-For header was sent
  (presumably by a proxy server) and if so, sets the host to that value.
  Otherwise, the IP address of the system sending the data is used.
* "none" leaves the host as specified in the HTTP header.

useACK = [true|false]
* When set to true, acknowledgment (ACK) is enabled. Events in a
request will
  be tracked until they are indexed. An events status (indexed or not)
can be
  queried from the ACK endpoint with the ID for the request.
* When set to false, acknowledgment is not enabled.
* This setting can be set at the stanza level.
* Defaults to false.
```

WINDOWS INPUTS:

```
#*******
# WINDOWS INPUTS:
#*******WINDOWS INPUTS:

* Windows platform specific input processor.
# ***********
# Splunk on Windows ships with several Windows-only inputs. They are
# defined in the default inputs.conf.

* Use the "disabled=" setting to enable/disable any of them.
* A short summary of the inputs follows:
  * Perfmon: Monitors Windows performance counters, objects, and
instances.
  * WinRegMon: Tracks and report any changes that occur in the
    local system Registry.
  * ADMon: Indexes existing Active Directory (AD) objects and listens
for AD
    changes.
  * WMI: Retrieves event logs remotely and locally through the Windows
    Management.  Instrumentation subsystem. It can also gather
performance
    data remotely, as well as receive various system notifications. See
```

```
    wmi.conf.spec for information on how to configure this input.

#*******
# The following Windows input specifications are for parsing on
non-Windows
# platforms.
#*******
```

Performance Monitor

```
#*******
# Performance Monitor
#*******Performance Monitor

[perfmon://<name>]

* This section explains possible settings for configuring
  the Windows Performance Monitor input.
* Each perfmon:// stanza represents an individually configured
performance
  monitoring input. If you configure the input through Splunk Web, then
the
  value of "<NAME>" matches what was specified there. While you can add
  performance monitor inputs manually, Splunk recommends that you use
Splunk
  Web to configure them, because it is easy to mistype the values for
  Performance Monitor objects, counters and instances.
* Note: The perfmon stanza is for local systems ONLY. To define
performance
  monitor inputs for remote machines, use wmi.conf.

object = <string>
* This is a valid Performance Monitor object as defined within
Performance
  Monitor (for example, "Process," "Server," "PhysicalDisk.")
* You can specify a single valid Performance Monitor object or use a
  regular expression (regex) to specify multiple objects.
* This setting is required, and the input will not run if the setting is
  not present.
* There is no default.

counters = <semicolon-separated strings>
* This can be a single counter, or multiple valid Performance Monitor
  counters.
* This setting is required, and the input will not run if the setting is
  not present.
* '*' is equivalent to all available counters for a given Performance
  Monitor object.
* There is no default.
```

```
instances = <semicolon-separated strings>
* This can be a single instance, or multiple valid Performance Monitor
  instances.
* '*' is  equivalent to all available instances for a given Performance
Monitor
  counter.
* If applicable instances are available for a counter and this setting
is not
  present, then the input logs data for all available instances (this is
the
  same as setting 'instances = *').
* If there are no applicable instances for a counter, then this setting
  can be safely omitted.
* There is no default.

interval = <integer>
* How often, in seconds, to poll for new data.
* This setting is required, and the input will not run if the setting is
  not present.
* The recommended setting depends on the Performance Monitor object,
  counter(s) and instance(s) that you define in the input, and how much
  performance data you require.
  * Objects with numerous instantaneous or per-second counters, such
    as "Memory," "Processor" and "PhysicalDisk" should have shorter
    interval times specified (anywhere from 1-3 seconds).
  * Less volatile counters such as "Terminal Services", "Paging File",
    and "Print Queue" can have longer times configured.
* Default is 300 seconds.

mode = [single|multikv]
* Specifies how the performance monitor input prints events.
* Set to 'single' to print each event individually, or 'multikv' to
  print events in multikv (formatted multiple key-value pair) format.
* Defaults to single.

samplingInterval = <sampling interval in ms>
* Advanced setting.
* How often, in milliseconds, to poll for new data.
* Enables high-frequency performance sampling. The input collects
  performance data every sampling interval. It then reports averaged
data
  and other statistics at every interval.
* The minimum legal value is 100, and the maximum legal value must be
less
  than what the 'interval' setting to.
* If not specified, high-frequency sampling does not take place.
* Defaults to not specified (disabled).

stats = <average;count;dev;min;max>
* Advanced setting.
* Reports statistics for high-frequency performance
```

```
  sampling.
* Acceptable values are: average, count, dev, min, max.
* You can specify multiple values by separating them with semicolons.
* If not specified, the input does not produce high-frequency sampling
  statistics.
* Defaults to not specified (disabled).

disabled = [0|1]
* Specifies whether or not the input is enabled.
* 1 to disable the input, 0 to enable it.
* Defaults to 0 (enabled).

index = <string>
* Specifies the index that this input should send the data to.
* This setting is optional.
* If no value is present, defaults to the default index.

showZeroValue = [0|1]
* Specfies whether or not zero value event data should be collected.
* Set to 1 to capture zero value event data, and 0 to ignore such data.
* Defaults to 0 (ignore zero value event data)

useEnglishOnly = [true|false]
* Controls which Windows Performance Monitor API the input uses.
* If true, the input uses PdhAddEnglishCounter() to add the counter
string.
  This ensures that counters display in English regardless of the
Windows
  host locale.
* If false, the input uses PdhAddCounter() to add the counter string.
* Note: if you set this setting to true, the 'object' setting does not
  accept a regular expression as a value on hosts that have a
non-English
  locale.
* Defaults to false.

formatString = <double format specifier>
* Controls the print format for double-precision statistic counters.
* Do not use quotes when specifying this string.
* Defaults to "%.20g" (without quotes).

###
# Direct Access File Monitor (does not use file handles)
# For Windows systems only.
###

[MonitorNoHandle://<path>]

* This input intercepts file writes to the specific file.
* <path> must be a fully qualified path name to a specific file.
Wildcards
  and directories are not accepted.
```

```
* You can specify more than one stanza of this type.

disabled = [0|1]
* Whether or not the input is enabled.
* Defaults to 0 (enabled).

index = <string>
* Specifies the index that this input should send the data to.
* This setting is optional.
* Defaults to the default index.
```

Windows Event Log Monitor

```
#*******
# Windows Event Log Monitor
#*******Windows Event Log Monitor

[WinEventLog://<name>]

* This section explains possible settings for configuring the
  Windows Event Log monitor.
* Each WinEventLog:// stanza represents an individually configured
WinEventLog
  monitoring input. If you you configure the input through Splunk Web,
the
  value of "<NAME>" matches what was specified there. While you can add
  event log monitor inputs manually, Splunk recommends that you use
Splunk
  Web to configure Windows event log monitor inputs because it is
  easy to mistype the values for event log channels.
* Note: The WinEventLog stanza is for local systems only. To define
event log
  monitor inputs for remote machines, use wmi.conf.

start_from = <string>
* How the input should chronologically read the Event Log channels.
* If you set this setting to 'oldest', the input reads Windows event
logs
  from oldest to newest.
* If you set this setting to 'newest' the input reads Windows event
logs
  in reverse, from newest to oldest. Once the input consumes the backlog
of
  events, it stops.
* Do not set this setting to 'newest' and at the same time set the
  'current_only' setting to 1. This results in the input not collecting
  any events because you instructed it to read existing events from
oldest
  to newest and read only incoming events concurrently (A logically
```

```
  impossible combination.)
* Defaults to oldest.

current_only = [0|1]
* Whether or not to acquire only events that arrive while the instance
is
  running.
* If you set this setting to 1, the input only acquires events that
arrive
  while the instance runs and the input is enabled. The input does not
read
  data which was stored in the Windows Event Log while the instance was
not
  running. This means that there will be gaps in the data if you restart
the
  instance or experiences downtime.
* If you set the setting to 0, the input first gets all existing events
  already stored in the log that have higher event IDs (have arrived
more
  recently) than the most recent events acquired. The input then
monitors
  events that arrive in real time.
* Do not set this setting to 1 and at the same time set the
  'start_from' setting to 'newest'. This results in the input not
collecting
  any events because you instructed it to read existing events from
oldest
  to newest and read only incoming events concurrently (A logically
  impossible combination.)
* Defaults to 0 (false), gathering stored events first before monitoring
  live events.

batch_size = <integer>
* How many Windows Event Log items to read per request.
* If troubleshooting identifies that the Event Log input is a
bottleneck in
  acquiring data, increasing this value can help.
  * NOTE: Splunk Support has seen cases where large values can result
in a
    stall in the Event Log subsystem.
    If you increase this value significantly, monitor closely for
trouble.
* In local testing and in customer acceptance testing, 10 worked well
  for both throughput and reliability.
* The default value is 10.

checkpointInterval = <integer>
* How often, in seconds, that the Windows Event Log input saves a
checkpoint.
* Checkpoints store the eventID of acquired events. This lets the input
  continue monitoring at the correct event after a shutdown or outage.
* The default value is 5.
```

```
disabled = [0|1]
* Whether or not the input is enabled.
* Set to 1 to disable the input, and 0 to enable it.
* The default is 0 (enabled).

evt_resolve_ad_obj = [1|0]
* How the input should interact with Active Directory while indexing
Windows
  Event Log events.
* If you set this setting to 1, the input resolves the Active
  Directory Security IDentifier (SID) objects to their canonical names
for
  a specific Windows Event Log channel.
* If you enable the setting, the rate at which the input reads events
  on high-traffic Event Log channels can decrease. Latency can also
increase
  during event acquisition. This is due to the overhead involved in
performing
  AD translations.
* When you set this setting to 1, you can optionally specify the domain
  controller name or dns name of the domain to bind to with the
'evt_dc_name'
  setting.  The input connects to that domain controller to resolve the
AD
  objects.
* If you set this setting to 0, the input does not attempt any
resolution.
* Defaults to 0 (disabled) for all channels.

evt_dc_name = <string>
* Which Active Directory domain controller to bind to for AD object
  resolution.
* If you prefix a dollar sign to a value (for example,
$my_domain_controller),
  the input interprets the value as an environment variable. If the
  environment variable has not been defined on the host, it is the same
  as if the value is blank.
* This setting is optional.
* This setting can be set to the NetBIOS name of the domain controller
  or the fully-qualified DNS name of the domain controller. Either name
  type can, optionally, be preceded by two backslash characters. The
following
  examples represent correctly formatted domain controller names:

    * "FTW-DC-01"
    * "\\FTW-DC-01"
    * "FTW-DC-01.splunk.com"
    * "\\FTW-DC-01.splunk.com"
    * $my_domain_controller

evt_dns_name = <string>
```

```
* The fully-qualified DNS name of the domain that the input should bind
to for
  AD object resolution.
* This setting is optional.

evt_resolve_ad_ds =[auto|PDC]
* How the input should choose the domain controller to bind for
  AD resolution.
* This setting is optional.
* If set to PDC, the input only contacts the primary domain controller
  to resolve AD objects.
* If set to auto, the input lets Windows chose the best domain
controller.
* If you set the 'evt_dc_name' setting, the input ignores this setting.
* Defaults to 'auto' (let Windows determine the domain controller to
use.)

evt_ad_cache_disabled = [0|1]
* Enables or disables the AD object cache.
* Defaults to 0.

evt_ad_cache_exp = <time in seconds>
* The expiration time, in seconds, for AD object cache entries.
* This setting is optional.
* The minimum allowed value is 10 and the maximum allowed value is
31536000.
* Defaults to 3600.

evt_ad_cache_exp_neg = <time in seconds>
* The expiration time, in seconds, for negative AD object cache entries.
* This setting is optional.
* The minimum allowed value is 10 and the maximum allowed value is
31536000.
* Defaults to 10.

evt_ad_cache_max_entries = <number of entries>
* The maximum number of AD object cache entries.
* This setting is optional.
* The minimum allowed value is 10 and the maximum allowed value is
40000.
* Defaults to 1000.

evt_sid_cache_disabled = [0|1]
* Enables or disables account Security IDentifier (SID) cache.
* This setting is global. It affects all Windows Event Log stanzas.
* Defaults to 0.

evt_sid_cache_exp = <time in seconds>
* The expiration time for account SID cache entries.
* This setting is optional.
* This setting is global. It affects all Windows Event Log stanzas.
* The minimum allowed value is 10 and the maximum allowed value is
```

```
31536000.
* Defaults to 3600.

evt_sid_cache_exp_neg = <time in seconds>
* The expiration time for negative account SID cache entries.
* This setting is optional.
* This setting is global. It affects all Windows Event Log stanzas.
* The minimum allowed value is 10 and the maximum allowed value is
31536000.
* Defaults to 10.

evt_sid_cache_max_entries = <number of entries>
* The maximum number of account SID cache entries.
* This setting is optional.
* This setting is global. It affects all Windows Event Log stanzas.
* The minimum allowed value is 10 and the maximum allowed value is
40000.
* Defaults to 10.

index = <string>
* Specifies the index that this input should send the data to.
* This setting is optional.
* If no value is present, defaults to the default index.

# Event Log filtering
#
# Filtering at the input layer is desirable to reduce the total
# processing load in network transfer and computation on the Splunk
# nodes that acquire and processing Event Log data.

whitelist = <list of eventIDs> | key=regex [key=regex]
blacklist = <list of eventIDs> | key=regex [key=regex]

whitelist1 = <list of eventIDs> | key=regex [key=regex]
whitelist2 = <list of eventIDs> | key=regex [key=regex]
whitelist3 = <list of eventIDs> | key=regex [key=regex]
whitelist4 = <list of eventIDs> | key=regex [key=regex]
whitelist5 = <list of eventIDs> | key=regex [key=regex]
whitelist6 = <list of eventIDs> | key=regex [key=regex]
whitelist7 = <list of eventIDs> | key=regex [key=regex]
whitelist8 = <list of eventIDs> | key=regex [key=regex]
whitelist9 = <list of eventIDs> | key=regex [key=regex]
blacklist1 = <list of eventIDs> | key=regex [key=regex]
blacklist2 = <list of eventIDs> | key=regex [key=regex]
blacklist3 = <list of eventIDs> | key=regex [key=regex]
blacklist4 = <list of eventIDs> | key=regex [key=regex]
blacklist5 = <list of eventIDs> | key=regex [key=regex]
blacklist6 = <list of eventIDs> | key=regex [key=regex]
blacklist7 = <list of eventIDs> | key=regex [key=regex]
blacklist8 = <list of eventIDs> | key=regex [key=regex]
blacklist9 = <list of eventIDs> | key=regex [key=regex]
```

```
* These settings are optional.
* Both numbered and unnumbered whitelists and blacklists support two
formats:
  * A comma-separated list of event IDs.
  * A list of key=regular expression pairs.
  * You cannot combine these formats. You can use either format on a
specific
    line.

* Numbered whitelist settings are permitted from 1 to 9, so whitelist1
through
  whitelist9 and blacklist1 through blacklist9 are supported.
* If no whitelist or blacklist rules are present, the input reads all
events.
```

Event Log whitelist and blacklist formats

```
###
# Event Log whitelist and blacklist formats
####Event Log whitelist and blacklist formats

* Event ID list format:
  * A comma-separated list of terms.
  * Terms may be a single event ID (e.g. 6) or range of event IDs (e.g.
100-200)
  * Example: 4,5,7,100-200
    * This applies to events with IDs 4, 5, 7, or any event ID between
100
      and 200, inclusive.
  * The event ID list format provides no additional functionality over
the
    key=regex format, but can be easier to understand:
    List format:      4,5,7,100-200
    Regex equivalent: EventCode=%^(4|5|7|1..|200)$%

* key=regex format:
  * A whitespace-separated list of Event Log components to match, and
    regular expressions to match against against them.
  * There can be one match expression or multiple expressions per line.
  * The key must belong to the set of valid keys provided below.
  * The regex consists of a leading delimiter, the regex expression, and
a
    trailing delimeter. Examples: %regex%, *regex*, "regex"
  * When multiple match expressions are present, they are treated as a
    logical AND.  In other words, all expressions must match for the
line to
    apply to the event.
  * If the value represented by the key does not exist, it is not
considered
```

```
    a match, regardless of the regex.
  * Example:
    whitelist = EventCode=%^200$% User=%jrodman%
    Include events only if they have EventCode 200 and relate to User
jrodman

# Valid keys for the key=regex format:

* The following keys are equivalent to the fields that appear in the
text of
  the acquired events:
  * Category, CategoryString, ComputerName, EventCode, EventType,
Keywords,
    LogName, Message, OpCode, RecordNumber, Sid, SidType, SourceName,
    TaskCategory, Type, User
* There are two special keys that do not appear literally in the event.
  * $TimeGenerated: The time that the computer generated the event
  * $Timestamp: The time that the event was received and recorded by the
               Event Log service.
* The 'EventType' key is only available on Windows Server 2003 /
  Windows XP and earlier.
* The 'Type' key is only available on Windows Server 2008 /
  Windows Vista and later.
* For a detailed definition of these keys, see the online documentation:
  http://docs.splunk.com/Documentation/Splunk/latest/Data/MonitorWindowsdata#Create_adv

suppress_text = [0|1]
* Whether or not to include the description of the event text for a
  given Event Log event.
* This setting is optional.
* Set this setting to 1 to suppress the inclusion of the event
  text description.
* Set this value to 0 to include the event text description.
* Defaults to 0.

renderXml= [true|false]
* Whether or not the input returns the event data in XML (eXtensible
Markup
  Language) format or in plain text.
* Set this to true to render events in XML.
* Set this to false to output events in plain text.
* Defaults to false.
```

Active Directory Monitor

```
#*******
# Active Directory Monitor
#*******Active Directory Monitor
```

```
[admon://<name>]

* This section explains possible settings for configuring the Active
Directory
  monitor input.
* Each admon:// stanza represents an individually configured Active
  Directory monitoring input. If you configure the input with Splunk
Web,
  then the value of "<NAME>" matches what was specified there. While
  you can add Active Directory monitor inputs manually, Splunk
recommends
  that you use Splunk Web to configure Active Directory monitor
  inputs because it is easy to mistype the values for Active Directory
  monitor objects.

targetDc = <string>
* The fully qualified domain name of a valid, network-accessible
  Active Directory domain controller.
* Defaults to the DC that the local host used to connect to AD. The
  input binds to its root Distinguished Name (DN).

startingNode = <string>
* Where in the Active Directory directory tree to start monitoring.
* The user that you configure the Splunk software to run as at
  installation determines where the input starts monitoring.
* If not specified, the input attempts to start at the root of
  the directory tree.

monitorSubtree = [0|1]
* Whether or not to monitor the subtree(s) of a given Active
  Directory tree path.
* Set this to 1 to monitor subtrees of a given directory tree
  path and 0 to monitor only the path itself.
* Defaults to 1 (monitor subtrees of a given directory tree path).

disabled = [0|1]
* Whether or not the input is enabled.
* Set this to 1 to disable the input and 0 to enable it.
* Defaults to 0 (enabled.)

index = <string>
* The index to store incoming data into for this input.
* This setting is optional.
* Defaults to the default index.

printSchema = [0|1]
* Whether or not to print the Active Directory schema.
* Set this to 1 to print the schema and 0 to not print
  the schema.
* Defaults to 1 (print the Active Directory schema).

baseline = [0|1]
```

```
* Whether or not to query baseline objects.
* Baseline objects are objects which currently reside in Active
Directory.
* Baseline objects also include previously deleted objects.
* Set this to 1 to query baseline objects, and 0 to not query
  baseline objects.
* Defaults to 0 (do not query baseline objects).
```

Windows Registry Monitor

```
###
# Windows Registry Monitor
###Windows Registry Monitor

[WinRegMon://<name>]

* This section explains possible settings for configuring the Windows
Registry
  Monitor input.
* Each WinRegMon:// stanza represents an individually configured
  WinRegMon monitoring input.
* If you configure the inputs with Splunk Web, the value of "<NAME>"
matches
  what was specified there. While you can add event log monitor inputs
  manually, recommends that you use Splunk Web to configure
  Windows registry monitor inputs because it is easy to mistype the
values
  for Registry hives and keys.
* The WinRegMon input is for local systems only.

proc = <string>
* Which processes this input should monitor for Registry access.
* If set, matches against the process name which performed the Registry
  access.
* The input includes events from processes that match the regular
expression
  that you specify here.
* The input filters out events for processes that do not match the
  regular expression.
* There is no default.

hive = <string>
* The Registry hive(s) that this input should monitor for Registry
access.
* If set, matches against the Registry key that was accessed.
* The input includes events from Registry hives that match the
  regular expression that you specify here.
* The input filters out events for Registry hives that do not match the
  regular expression.
```

```
* There is no default.

type = <string>
* A regular expression that specifies the type(s) of Registry event(s)
  that you want the input to monitor.
* There is no default.

baseline = [0|1]
* Whether or not the input should get a baseline of Registry events
  when it starts.
* If you set this to 1, the input captures a baseline for
  the specified hive when it starts for the first time. It then
  monitors live events.
* Defaults to 0 (do not capture a baseline for the specified hive
  first before monitoring live events).

baseline_interval = <integer>
* Selects how much downtime in continuous registry monitoring should
trigger
  a new baseline for the monitored hive and/or key.
* In detail:
  * Sets the minimum time interval, in seconds, between baselines.
  * At startup, a WinRegMon input will not generate a baseline if less
time
    has passed since the last checkpoint than baseline_interval
chooses.
  * In normal operation, checkpoints are updated frequently as data is
    acquired, so this will cause baselines to occur only when monitoring
was
    not operating for a period of time.
* If baseline is set to 0 (disabled), has no effect.
* Defaults to 0 (always baseline on startup, if baseline is 1)

disabled = [0|1]
* Whether or not the input is enabled.
* Set this to 1 to disable the input, or 0 to enable it.
* Defaults to 0 (enabled).

index = <string>
* The index that this input should send the data to.
* This setting is optional.
* Defaults to the default index.
```

Windows Host Monitoring

```
###
# Windows Host Monitoring
###Windows Host Monitoring
```

```
[WinHostMon://<name>]

* This section explains possible settings for configuring the Windows
host
  monitor input.
* Gathers status information from the local Windows system components as
  per the type field below.
* Each WinHostMon:// stanza represents an WinHostMon monitoring input.
* The "<name>" component of the stanza name will be used as the source
field
  on generated events, unless an explicit source setting is added to the
  stanza.  It does not affect what data is collected (see type setting
for
  that).
* If you configure the input in Splunk web, the value of "<name>"
matches
  what was specified there.
* Note: The WinHostMon input is for local Windows systems only. You
  can not monitor Windows host information remotely.

type = <semicolon-separated strings>
* An expression that specifies the type(s) of host inputs
  that you want the input to monitor.
* Type can be (case insensitive)
  Computer;Process;Processor;NetworkAdapter;Service;OperatingSystem;Disk;Driver;Roles

interval = <integer>
* The interval, in seconds, between when the input runs to gather
  Windows host information and generate events.
* See interval in the Scripted input section for more information.

disabled = [0|1]
* Whether or not the input is enabled.
* Set this to 1 to disable the input, or 0 to enable it.
* Defaults to 0 (enabled).

index = <string>
* The index that this input should send the data to.
* This setting is optional.
* Defaults to the default index.

[WinPrintMon://<name>]

* This section explains possible settings for configuring the Windows
print
  monitor input.
* Each WinPrintMon:// stanza represents an WinPrintMon monitoring input.
  The value of "<name>" matches what was specified in Splunk Web.
* Note: The WinPrintMon input is for local Windows systems only.
* The "<name>" component of the stanza name will be used as the source
field
  on generated events, unless an explicit source setting is added to the
```

```
  stanza.  It does not affect what data is collected (see type setting
for
  that).

type = <semicolon-separated strings>
* An expression that specifies the type(s) of print inputs
  that you want the input to monitor.
* Type can be (case insensitive)
  Printer;Job;Driver;Port

baseline = [0|1]
* Whether or not to capture a baseline of print objects when the
  input starts for the first time.
* If you set this to 1, the input captures a baseline of
  the current print objects when the input starts for the first time.
* Defaults to 0 (do not capture a baseline.)

disabled = [0|1]
* Whether or not the input is enabled.
* Set to 1 to disable the input, or 0 to enable it.
* Defaults to 0 (enabled).

index = <string>
* The index that this input should send the data to.
* This setting is optional.
* Defaults to the default index.

[WinNetMon://<name>]

* This section explains possible settings for configuring
  a Network Monitor input.
* Each WinNetMon:// stanza represents an individually configured network
  monitoring input.  The value of "<name>" matches what was specified
  in Splunk Web. Splunk recommends that you use Splunk Web to
  configure Network Monitor inputs because it is easy to mistype
  the values for Network Monitor objects.

remoteAddress = <regular expression>
* A regular expression that represents the remote IP address of a
  host that is involved in network communication.
* This setting accepts a regular expression that matches against
  IP addresses only, not host names. For example: 192\.163\..*
* The input includes events for remote IP addresses that match
  the regular expression that you specify here.
* The input filters out events for remote IP addresses that do not
  match the regular expression.
* Defaults to unset (including all remote address events).

process = <regular expression>
* A regular expression that represents the process or application that
  performed a network access.
* The input includes events for processes that match the
```

```
  regular expression that you specify here.
* The input filters out events for processes that do not match the
  regular expression.
* Defaults to unset (including all processes and application events).

user = <regular expression>
* A regular expression that represents the Windows user name that
  performed a network access.
* The input includes events for user names that match the
  regular expression that you specify here.
* The input filters out events for user names that do not match the
  regular expression.
* Defaults to unset (including all user name events).

addressFamily = ipv4;ipv6
* Determines the events to include by network address family.
* Setting ipv4 alone will include only TCP/IP v4 packets, while ipv6
alone
  will include only TCP/IP v6 packets.
* To specify both families, separate them with a semicolon.
  For example: ipv4;ipv6
* Defaults to unset (including events with both address families).

packetType = connect;accept;transport.
* Determines the events to include by network packet type.
* To specify multiple packet types, separate them with a semicolon.
  For example: connect;transport
* Defaults to unset (including events with any packet type).

direction = inbound;outbound
* Determines the events to include by network transport direction.
* To specify multiple directions, separate them with a semicolon.
  For example: inbound;outbound
* Defaults to unset (including events with any direction).

protocol = tcp;udp
* Determines the events to include by network protocol.
* To specify multiple protocols, separate them with a semicolon.
  For example: tcp;udp
* For more information about protocols, see
  http://www.ietf.org/rfc/rfc1700.txt
* Defaults to unset (including events with all protocols).

readInterval = <integer>
* How often, in milliseconds, that the input should read the network
  kernel driver for events.
* Advanced option. Use the default value unless there is a problem
  with input performance.
* Set this to adjust the frequency of calls into the network kernel
driver.
* Choosing lower values (higher frequencies) can reduce network
  performance, while higher numbers (lower frequencies) can cause event
```

```
  loss.
* The minimum allowed value is 10 and the maximum allowed value is 1000.
* Defaults to unset, handled as 100 (msec).

driverBufferSize = <integer>
* The maximum number of packets that the network kernel driver retains
  for retrieval by the input.
* Set to adjust the maximum number of network packets retained in
  the network driver buffer.
* Advanced option. Use the default value unless there is a problem
  with input performance.
* Configuring this setting to lower values can result in event loss,
while
  higher values can increase the size of non-paged memory on the host.
* The minimum allowed value is 128 and the maximum allowed value is
32768.
* Defaults to unset, handled as 32768 (packets).

userBufferSize = <integer>
* The maximum size, in megabytes, of the user mode event buffer.
* Controls amount of packets cached in the the user mode.
* Advanced option. Use the default value unless there is a problem
  with input performance.
* Configuring this setting to lower values can result in event loss,
while
  higher values can increase the amount of memory that the network
  monitor uses.
* The minimum allowed value is 20 and the maximum allowed value is 500.
* Defaults to unset, handled as 20 (megabytes).

mode = single|multikv
* Specifies how the network monitor input generates events.
* Set to 'single' to generate one event per packet, or 'multikv' to
  generate combined events of many packets in multikv format (many
packets
  described in a single table as one event).
* Defaults to single.

multikvMaxEventCount = <integer>
* The maximum number of packets to combine in multikv format when you
set
  the 'mode' setting to 'multikv'.
* Has no effect when 'mode' is set to 'single'.
* Advanced option.
* The minimum allowed value is 10 and the maximum allowed value is 500.
* Defaults to 100.

multikvMaxTimeMs = <integer>
* The maximum amount of time, in milliseconds, to accumulate packet data
to
  combine into a large tabular event in multikv format.
* Has no effect when 'mode' is set to 'single'.
```

```
* Advanced option.
* The minimum allowed value is 100 and the maximum allowed value is
5000.
* Defaults to 1000.

sid_cache_disabled = 0|1
* Enables or disables account Security IDentifier (SID) cache.
* This setting is global. It affects all Windows Network Monitor
stanzas.
* Defaults to 0.

sid_cache_exp = <time in seconds>
* The expiration time for account SID cache entries.
* This setting is optional.
* This setting is global. It affects all Windows Network Monitor
stanzas.
* The minimum allowed value is 10 and the maximum allowed value is
31536000.
* Defaults to 3600.

sid_cache_exp_neg = <time in seconds>
* The expiration time for negative account SID cache entries.
* This setting is optional.
* This setting is global. It affects all Windows Network Monitor
stanzas.
* The minimum allowed value is 10 and the maximum allowed value is
31536000.
* Defaults to 10.

sid_cache_max_entries = <number of entries>
* The maximum number of account SID cache entries.
* This setting is optional.
* This setting is global. It affects all Windows Network Monitor
stanzas.
* The minimum allowed value is 10 and the maximum allowed value is
40000.
* Defaults to 10.

disabled = 0|1
* Whether or not the input is enabled.
* Defaults to 0 (enabled.)

index = <string>
* The index that this input should send the data to.
* This setting is optional.
* Defaults to the default index.

[powershell://<name>]
* Runs Windows PowerShell version 3 commands or scripts.

script = <command>
* A PowerShell command-line script or .ps1 script file that the input
```

```
  should run.
* There is no default.

schedule = [<number>|<cron schedule>]
* How often to run the specified PowerShell command or script.
* You can specify a number in seconds, or provide a valid cron
  schedule.
* Defaults to running the command or script once, at startup.

[powershell2://<name>]
* Runs Windows PowerShell version 2 commands or scripts.

script = <command>
* A PowerShell command-line script or .ps1 script file that the input
  should run.

schedule = <schedule>
* How often to run the specified PowerShell command or script.
* You can provide a valid cron schedule.
* Defaults to running the command or script once, at startup.
```

inputs.conf.example

```
#   Version 6.5.0
#
# This is an example inputs.conf. Use this file to configure data
inputs.
#
# To use one or more of these configurations, copy the configuration
block into
# inputs.conf in $SPLUNK_HOME/etc/system/local/. You must restart
Splunk to
# enable configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles


# The following configuration reads all the files in the directory
/var/log.

[monitor:///var/log]


# The following configuration reads all the files under /var/log/httpd
and
```

```
# classifies them as sourcetype::access_common.
#
# When checking a file for new data, if the file's modification time is
from
# before seven days ago, the file will no longer be checked for changes
# until you restart the software.

[monitor:///var/log/httpd]
sourcetype = access_common
ignoreOlderThan = 7d


# The following configuration reads all the
# files under /mnt/logs. When the path is /mnt/logs/<host>/... it
# sets the hostname (by file) to <host>.

[monitor:///mnt/logs]
host_segment = 3


# The following configuration listens on TCP port 9997 for raw
# data from ANY remote server (not just a Splunk instance). The host of
the
# data is set to the IP address of the remote server.

[tcp://:9997]


# The following configuration listens on TCP port 9995 for raw
# data from ANY remote server. The host of the data is set as the host
name of
# the remote server.  All data will also be assigned the sourcetype
"log4j" and
# the source "tcp:9995".

[tcp://:9995]
connection_host = dns
sourcetype = log4j
source = tcp:9995


# The following configuration listens on TCP port 9995 for raw
# data from 10.1.1.10.
# All data is assigned the host "webhead-1", the sourcetype
"access_common" and
# the the source "//10.1.1.10/var/log/apache/access.log".

[tcp://10.1.1.10:9995]
host = webhead-1
sourcetype = access_common
source = //10.1.1.10/var/log/apache/access.log
```

```
# The following configuration listens on TCP port 9996 for
# Splunk cooked event data from ANY splunk forwarder.
# The host of the data is set to the host name of the remote server
ONLY IF the
# remote data has no host set, or if it is set to "localhost".

[splunktcp://:9996]
connection_host = dns


# The following configuration listens on TCP port 9996 for
# distributed search data from 10.1.1.100. The data is processed the
same as
# locally indexed data.

[splunktcp://10.1.1.100:9996]


# The following configuration listens on TCP port 514 for data
# from syslog.corp.company.net. The data is assigned the sourcetype
"syslog"
# and the host is set to the host name of the remote server.

[tcp://syslog.corp.company.net:514]
sourcetype = syslog
connection_host = dns

# Following configuration limits the acceptance of data to forwarders
# that have been configured with the token value specified in 'token'
field.
# NOTE: The token value is encrypted. The REST endpoint encrypts the
token
# while saving it.

[splunktcptoken://tok1]
token = $1$ifQTPTzHD/BA8VgKvVcgO1KQAtr3N1C8S/1uK3nAKIE9dd9e9g==

# Set up Secure Sockets Layer (SSL):

[SSL]
serverCert=$SPLUNK_HOME/etc/auth/server.pem
password=password
rootCA=$SPLUNK_HOME/etc/auth/cacert.pem
requireClientCert=false

[splunktcp-ssl:9996]

# Use file system change monitor:

[fschange:/etc/]
fullEvent=true
```

```
pollPeriod=60
recurse=true
sendEventMaxSize=100000
index=main

# Monitor the Security Windows Event Log channel, getting the most
recent
# events first, then older, and finally continuing to gather newly
arriving events

[WinEventLog://Security]
disabled = 0
start_from = newest
evt_dc_name =
evt_dns_name =
evt_resolve_ad_ds =
evt_resolve_ad_obj = 1
checkpointInterval = 5

# Monitor the ForwardedEvents Windows Event Log channel, only gathering
the
# events that arrive after monitoring starts, going forward in time.

[WinEventLog://ForwardedEvents]
disabled = 0
start_from = oldest
current_only = 1
batch_size = 10
checkpointInterval = 5

[tcp://9994]
queueSize=50KB
persistentQueueSize=100MB

# Perfmon: Windows performance monitoring examples

# You must specify the names of objects, counters and instances
# exactly as they are shown in the Performance Monitor application.
Splunk Web
# is the recommended interface to use to configure performance monitor
inputs.

# These stanzas gather performance data from the local system only.
# Use wmi.conf for performance monitor metrics on remote systems.

# Query the PhysicalDisk performance object and gather disk access data
for
# all physical drives installed in the system. Store this data in the
# "perfmon" index.
# Note: If the interval attribute is set to 0, Splunk will reset the
interval
# to 1.
```

```
[perfmon://LocalPhysicalDisk]
interval = 0
object = PhysicalDisk
counters = Disk Bytes/sec; % Disk Read Time; % Disk Write Time; % Disk
Time
instances = *
disabled = 0
index = PerfMon

# Gather common memory statistics using the Memory performance object,
every
# 5 seconds. Store the data in the "main" index. Since none of the
counters
# specified have applicable instances, the instances attribute is not
required.

[perfmon://LocalMainMemory]
interval = 5
object = Memory
counters = Committed Bytes; Available Bytes; % Committed Bytes In Use
disabled = 0
index = main

# Gather data on USB activity levels every 10 seconds. Store this data
in the
# default index.

[perfmon://USBChanges]
interval = 10
object = USB
counters = Usb Control Data Bytes/Sec
instances = *
disabled = 0

# Admon: Windows Active Directory monitoring examples

# Monitor the default domain controller (DC) for the domain that the
computer
# running Splunk belongs to. Start monitoring at the root node of Active
# Directory.
[admon://NearestDC]
targetDc =
startingNode =

# Monitor a specific DC, with a specific starting node. Store the events
in
# the "admon" Splunk index. Do not print Active Directory schema. Do not
# index baseline events.

[admon://DefaultTargetDC]
targetDc = pri01.eng.ad.splunk.com
```

```
startingNode = OU=Computers,DC=eng,DC=ad,DC=splunk,DC=com
index = admon
printSchema = 0
baseline = 0

# Monitor two different DCs with different starting nodes.
[admon://DefaultTargetDC]
targetDc = pri01.eng.ad.splunk.com
startingNode = OU=Computers,DC=eng,DC=ad,DC=splunk,DC=com

[admon://SecondTargetDC]
targetDc = pri02.eng.ad.splunk.com
startingNode = OU=Computers,DC=hr,DC=ad,DC=splunk,DC=com
```

instance.cfg.conf

The following are the spec and example files for instance.cfg.conf.

instance.cfg.conf.spec

```
#   Version 6.5.0
#
# This file contains the set of attributes and values you can expect to
find in
# the SPLUNK_HOME/etc/instance.cfg file; the instance.cfg file is not
to be
# modified or removed by user.  LEAVE THE instance.cfg FILE ALONE.
#

#
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# The [general] stanza defines global settings.
#
```

[general]

```
[general]
guid = <GUID in all-uppercase>
* This setting formerly (before 5.0) belonged in the [general] stanza of
  server.conf file.
```

```
* Splunk expects that every Splunk instance will have a unique string
for this
  value, independent of all other Splunk instances.  By default, Splunk
will
  arrange for this without user intervention.

* Currently used by (not exhaustive):
  * Clustering environments, to identify participating nodes.
  * Splunk introspective searches (Splunk on Splunk, Deployment
Monitor,
    etc.), to identify forwarders.

* At startup, the following happens:

  * If server.conf has a value of 'guid' AND instance.cfg has no value
of
    'guid', then the value will be erased from server.conf and moved to
    instance.cfg file.

  * If server.conf has a value of 'guid' AND instance.cfg has a value of
    'guid' AND these values are the same, the value is erased from
    server.conf file.

  * If server.conf has a value of 'guid' AND instance.cfg has a value of
'guid'
    AND these values are different, startup halts and error is shown.
 Operator
    must resolve this error.  We recommend erasing the value from
server.conf
    file, and then restarting.

  * If you are hitting this error while trying to mass-clone Splunk
installs,
    please look into the command 'splunk clone-prep-clear-config';
    'splunk help' has help.

* See http://www.ietf.org/rfc/rfc4122.txt for how a GUID (a.k.a. UUID)
is
  constructed.

* The standard regexp to match an all-uppercase GUID is
  "[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}".
```

instance.cfg.conf.example

```
#   Version 6.5.0
#
# This file contains an example SPLUNK_HOME/etc/instance.cfg file; the
```

```
# instance.cfg file is not to be modified or removed by user.  LEAVE THE
# instance.cfg FILE ALONE.
#

[general]
guid = B58A86D9-DF3D-4BF8-A426-DB85C231B699
```

limits.conf

The following are the spec and example files for limits.conf.

limits.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attribute/value pairs for configuring
limits for
# search commands.
#
# There is a limits.conf in $SPLUNK_HOME/etc/system/default/.  To set
custom
# configurations, place a limits.conf in
$SPLUNK_HOME/etc/system/local/. For
# examples, see limits.conf.example. You must restart Splunk to enable
# configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
#

# limits.conf settings and DISTRIBUTED SEARCH
#   Unlike most settings which affect searches, limits.conf settings are
not
#   provided by the search head to be used by the search peers.  This
means
#   that if you need to alter search-affecting limits in a distributed
#   environment, typically you will need to modify these settings on the
#   relevant peers and search head for consistent results.
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
top of
#     the file.
#   * Each conf file should have at most one default stanza. If there
are
#     multiple default stanzas, attributes are combined. In the case of
#     multiple definitions of the same attribute, the last definition in
the
#     file wins.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.

# CAUTION: Do not alter the settings in limits.conf unless you know what
you
#          are doing.  Improperly configured limits may result in
splunkd
#          crashes and/or memory overuse.

* Each stanza controls different parameters of search commands.
```

[default]

```
[default]
max_mem_usage_mb = <non-negative integer>
* Provides a limitation to the amount of RAM a batch of events or
results will
  use in the memory of a search process.
* Operates on an estimation of memory use which is not exact.
* The limitation is applied in an unusual way; if the number of results
or
  events exceeds maxresults, AND the estimated memory exceeds this
limit, the
  data is spilled to disk.
* This means, as a general rule, lower limits will cause a search to use
more
  disk I/O and less RAM, and be somewhat slower, but should cause the
same
  results to typically come out of the search in the end.
* This limit is applied currently to a number, but not all search
processors.
  However, more will likely be added as it proves necessary.
* The number is thus effectively a ceiling on batch size for many
components of
  search for all searches run on this system.
```

```
* 0 will specify the size to be unbounded.  In this case searches may
be
  allowed to grow to arbitrary sizes.

* The 'mvexpand' command uses this value in a different way.
  * mvexpand has no combined logic with maxresults
  * If the memory limit is exceeded, output is truncated, not spilled
to disk.

* The 'stats' processor uses this value in the following way.
* If the estimated memory usage exceeds the specified limit, the
results are spilled to disk
* If '0' is specified, the results are spilled to the disk when the
number of results
    exceed the maxresultrows setting.

* This value is not exact. The estimation can deviate by an order of
magnitude
  or so to both the smaller and larger sides.
* Defaults to 200 (MB)

min_batch_size_bytes = <integer>
* Specifies the size of the file/tar after which the file is handled by
the
  batch reader instead of the trailing processor.
* Global parameter, cannot be configured per input.
* Note configuring this to a very small value could lead to backing up
of jobs
  at the tailing processor.
* Defaults to 20 MB.

DelayArchiveProcessorShutdown = <bool>
* Specifies whether during splunk shutdown archive processor should
finish processing archive file under process.
* If set to false archive processor abandons further processing of
archive file and will process again from start again.
* If set to true archive processor will complete processing of archive
file. Shutdown will be delayed.
* defaults to false
```

[searchresults]

```
[searchresults]
* This stanza controls search results for a variety of Splunk search
commands.

maxresultrows = <integer>
* Configures the maximum number of events are generated by search
commands
  which grow the size of your result set (such as multikv) or that
```

```
create
  events. Other search commands are explicitly controlled in specific
stanzas
  below.
* This limit should not exceed 50000. Setting this limit higher than
50000
  causes instability.
* Defaults to 50000.

tocsv_maxretry = <integer>
* Maximum number of times to retry the atomic write operation.
* 1 = no retries.
* Defaults to 5.

tocsv_retryperiod_ms = <integer>
* Period of time to wait before each retry.
* Defaults to 500.

* These setting control logging of error messages to info.csv
  All messages will be logged to search.log regardless of these
settings.

compression_level = <integer>
* Compression level to use when writing search results to .csv.gz files
* Defaults to 1
```

[search_info]

```
[search_info]
* This stanza controls logging of messages to the info.csv file
* Messages logged to info.csv are available to REST API clients
  and the Splunk UI, so limiting the messages
  added to info.csv will mean that these messages will not be
  available in the UI and/or the REST API.

max_infocsv_messages  = <positive integer>
* If more than max_infocsv_messages log entries are generated,
additional
  entries will not be logged in info.csv. All entries will still be
logged in
  search.log.

infocsv_log_level = [DEBUG|INFO|WARN|ERROR]
* Limits the messages which are added to info.csv to the stated level
  and above.
* For example, if log_level is WARN, messages of type WARN and higher
  will be added to info.csv

show_warn_on_filtered_indexes = <boolean>
* Log warnings if search returns no results because user has
```

```
  no permissions to search on queried indexes

filteredindexes_log_level = [DEBUG|INFO|WARN|ERROR]
* Log level of messages when search results no results because
  user has no permissions to search on queries indexes
```

[subsearch]

```
[subsearch]
* This stanza controls subsearch results.
* NOTE: This stanza DOES NOT control subsearch results when a subsearch
is
  called by commands such as join, append, or appendcols.
* Read more about subsearches in the online documentation:
  http://docs.splunk.com/Documentation/Splunk/latest/Search/Aboutsubsearches

maxout = <integer>
* Maximum number of results to return from a subsearch.
* This value cannot be greater than or equal to 10500.
* Defaults to 10000.

maxtime = <integer>
* Maximum number of seconds to run a subsearch before finalizing
* Defaults to 60.

ttl = <integer>
* Time to cache a given subsearch's results, in seconds.
* Do not set this below 120 seconds.
* See definition in [search] ttl for more details on how the ttl is
computed
* Defaults to 300.
```

[anomalousvalue]

```
[anomalousvalue]
maxresultrows = <integer>
* Configures the maximum number of events that can be present in memory
at one
  time.
* Defaults to searchresults::maxresultsrows (which is by default
50000).

maxvalues = <integer>
* Maximum number of distinct values for a field.
* Defaults to 100000.

maxvaluesize = <integer>
* Maximum size in bytes of any single value (truncated to this size if
```

```
larger).
* Defaults to 1000.
```

[associate]

```
[associate]
maxfields = <integer>
* Maximum number of fields to analyze.
* Defaults to 10000.

maxvalues = <integer>
* Maximum number of values for any field to keep track of.
* Defaults to 10000.

maxvaluesize = <integer>
* Maximum length of a single value to consider.
* Defaults to 1000.
```

[autoregress]

```
[autoregress]
maxp = <integer>
* Maximum valid period for auto regression
* Defaults to 10000.

maxrange = <integer>
* Maximum magnitude of range for p values when given a range.
* Defaults to 1000.
```

[concurrency]

```
[concurrency]
max_count = <integer>
* Maximum number of detected concurrencies.
* Defaults to 10000000
```

[ctable]

```
[ctable]
* This stanza controls the contingency, ctable, and counttable commands.

maxvalues = <integer>
* Maximum number of columns/rows to generate (the maximum number of
distinct
  values for the row field and column field).
```

```
* Defaults to 1000.
```

[correlate]

```
[correlate]
maxfields = <integer>
* Maximum number of fields to correlate.
* Defaults to 1000.
```

[discretize]

```
[discretize]
* This stanza set attributes for bin/bucket/discretize.

default_time_bins = <integer>
* When discretizing time for timechart or explicitly via bin, the
default bins
  to use if no span or bins is specified.
* Defaults to 100

maxbins = <integer>
* Maximum number of buckets to discretize into.
* If maxbins is not specified or = 0, it defaults to
  searchresults::maxresultrows
* Defaults to 50000.
```

[export]

```
[export]
add_timestamp = <bool>
* Add a epoch time timestamp to JSON streaming output that reflects the
time
  the results were generated/retrieved
* Defaults to false

add_offset = <bool>
* Add an offset/row number to JSON streaming output
* Defaults to true
```

[extern]

```
[extern]
perf_warn_limit = <integer>
* Warn when external scripted command is applied to more than this many
events
```

```
* set to 0 for no message (message is always INFO level)
* Defaults to 10000
```

[inputcsv]

```
[inputcsv]
mkdir_max_retries = <integer>
* Maximum number of retries for creating a tmp directory (with random
name as
  subdir of SPLUNK_HOME/var/run/splunk)
* Defaults to 100.
```

[indexpreview]

```
[indexpreview]
max_preview_bytes = <integer>
* Maximum number of bytes to read from each file during preview
* Defaults to 2000000 (2 MB)

max_results_perchunk = <integer>
* Maximum number of results to emit per call to preview data generator
* Defaults to 2500.

soft_preview_queue_size = <integer>
* Loosely-applied maximum on number of preview data objects held in
memory
* Defaults to 100.
```

[join]

```
[join]
subsearch_maxout = <integer>
* Maximum result rows in output from subsearch to join against.
* Defaults to 50000.

subsearch_maxtime = <integer>
* Maximum search time (in seconds) before auto-finalization of
subsearch.
* Defaults to 60

subsearch_timeout = <integer>
* Maximum time to wait for subsearch to fully finish (in seconds).
* Defaults to 120.
```

[kmeans]

```
[kmeans]
maxdatapoints = <integer>
* Maximum data points to do kmeans clusterings for.
* Defaults to 100000000.

maxkvalue = <integer>
* Maximum number of clusters to attempt to solve for.
* Defaults to 1000.

maxkrange = <integer>
* Maximum number of k values to iterate over when specifying a range.
* Defaults to 100.
```

[kv]

```
[kv]
maxcols = <integer>
* When non-zero, the point at which kv should stop creating new fields.
* Defaults to 512.

limit = <integer>
* Maximum number of keys auto kv can generate.
* Defaults to 100.

maxchars = <integer>
* Truncate _raw to this size and then do auto KV.
* Defaults to 10240 characters.

max_extractor_time = <integer>
* Maximum amount of CPU time, in milliseconds, that a key-value pair
extractor
  will be allowed to take before warning. If the extractor exceeds this
  execution time on any event a warning will be issued
* Defaults to 1000.

avg_extractor_time = <integer>
* Maximum amount of CPU time, in milliseconds, that the average (over
search
  results) execution time of a key-value pair extractor will be allowed
to take
  before warning. Once the average becomes larger than this amount of
time a
  warning will be issued
* Defaults to 500
```

[lookup]

```
[lookup]
max_memtable_bytes = <integer>
* Maximum size of static lookup file to use an in-memory index for.
* Defaults to 10000000 in bytes (10MB)
* Lookup files with size above max_memtable_bytes will be indexed on
disk
* A large value results in loading large lookup files in memory leading
to bigger process memory footprint.
* Caution must be exercised when setting this parameter to arbitrarily
high values!

max_matches = <integer>
* maximum matches for a lookup
* range 1 - 1000
* Defaults to 1000

max_reverse_matches = <integer>
* maximum reverse lookup matches (for search expansion)
* Defaults to 50

batch_index_query = <bool>
* Should non-memory file lookups (files that are too large) use batched
queries
  to possibly improve performance?
* Defaults to true

batch_response_limit = <integer>
* When doing batch requests, the maximum number of matches to retrieve
  if more than this limit of matches would otherwise be retrieve, we
will fall
  back to non-batch mode matching
* Defaults to 5000000

max_lookup_messages = <positive integer>
* If more than "max_lookup_messages" log entries are generated,
additional
  entries will not be logged in info.csv. All entries will still be
logged in
  search.log.
```

[metrics]

```
[metrics]
maxseries = <integer>
* The number of series to include in the per_x_thruput reports in
metrics.log.
* Defaults to 10.
```

```
interval = <integer>
* Number of seconds between logging splunkd metrics to metrics.log.
* Minimum of 10.
* Defaults to 30.
```

[metrics:tcpin_connections]

```
[metrics:tcpin_connections]
aggregate_metrics = [true|false]
* For each splunktcp connection from forwarder, splunk logs metrics
information
  every metrics interval.
* When there are large number of forwarders connected to indexer, the
amount of
  information logged can take lot of space in metrics.log. When set to
true, it
  will aggregate information across each connection and report only once
per
  metrics interval.
* Defaults to false

suppress_derived_info = [true|false]
* For each forwarder connection, _tcp_Bps, _tcp_KBps, _tcp_avg_thruput,
  _tcp_Kprocessed is logged in metrics.log.
* This can be derived from kb. When set to true, the above derived info
will
  not be emitted.
* Defaults to true
```

[rare]

```
[rare]
maxresultrows = <integer>
* Maximum number of result rows to create.
* If not specified, defaults to searchresults::maxresultrows
* Defaults to 50000.

maxvalues = <integer>
* Maximum number of distinct field vector values to keep track of.
* Defaults 100000.

maxvaluesize = <integer>
* Maximum length of a single value to consider.
* Defaults to 1000.
```

[restapi]

```
[restapi]
maxresultrows = <integer>
* Maximum result rows to be returned by /events or /results getters from
REST
  API.
* Defaults to 50000.

time_format_reject = <regular expression>
* HTTP parameters for time_format and output_time_format which match
  this regex will be rejected (blacklisted).
* The regex will be satisfied by a substring match anywhere in the
parameter.
* Intended as defense-in-depth against XSS style attacks against
browser users
  by crafting specially encoded URLS for them to access splunkd.
* If unset, all parameter strings will be accepted.
* To disable this check entirely, set the value to empty.
  # Example of disabling: time_format_reject =
* Defaults to [<>!] , which means that the less-than '<', greater-than
'>', and
  exclamation point '!' are not allowed.

jobscontentmaxcount = <integer>
* Maximum length of a property in the contents dictionary of an entry
from
  /jobs getter from REST API
* Value of 0 disables truncation
* Defaults to 0
```

[search_metrics]

```
[search_metrics]
debug_metrics = <bool>
* This indicates whether we should output more detailed search metrics
for
  debugging.
* This will do things like break out where the time was spent by peer,
and may
  add additional deeper levels of metrics.
* This is NOT related to "metrics.log" but to the "Execution Costs" and
  "Performance" fields in the Search inspector, or the count_map in the
info.csv file.
* Defaults to false
```

[search]

```
[search]
summary_mode = [all|only|none]
* Controls if precomputed summary are to be used if possible?
* all: use summary if possible, otherwise use raw data
* only: use summary if possible, otherwise do not use any data
* none: never use precomputed summary data
* Defaults to 'all'

result_queue_max_size = <integer>
* Controls the size of the search results queue in dispatch
* Default size is set to 100MB
* Use caution while playing with this parameter

use_bloomfilter = <bool>
* Control whether to use bloom filters to rule out buckets
* Default value set to true

max_id_length = <integer>
* Maximum length of custom search job id when spawned via REST API arg
id=

ttl = <integer>
* How long search artifacts should be stored on disk once completed, in
  seconds. The ttl is computed relative to the modtime of status.csv of
the job
  if such file exists or the modtime of the search job's artifact
directory. If
  a job is being actively viewed in the Splunk UI then the modtime of
  status.csv is constantly updated such that the reaper does not remove
the job
  from underneath.
* Defaults to 600, which is equivalent to 10 minutes.

failed_job_ttl = <integer>
* How long search artifacts should be stored on disk once failed, in
seconds. The ttl is computed
* relative to the modtime of status.csv of the job if such file exists
or the modtime of the search
* job's artifact directory. If a job is being actively viewed in the
Splunk UI then the modtime of
* The status.csv file is constantly updated such that the reaper does
not remove the job from underneath.
* Defaults to 86400, which is equivalent to 24 hours.

default_save_ttl = <integer>
* How long the ttl for a search artifact should be extended in response
to the
  save control action, in second.  0 = indefinitely.
```

```
* Defaults to 604800 (1 week)

remote_ttl = <integer>
* How long artifacts from searches run in behalf of a search head
should be
  stored on the indexer after completion, in seconds.
* Defaults to 600 (10 minutes)

status_buckets = <integer>
* The approximate maximum number buckets to generate and maintain in the
  timeline.
* Defaults to 0, which means do not generate timeline information.

max_bucket_bytes = <integer>
* This setting has been deprecated and has no effect

max_count = <integer>
* The number of events that can be accessible in any given status bucket
(when status_buckets = 0).
* The last accessible event in a call that takes a base and bounds.
* Defaults to 500000.
* Note: This value does not reflect the number of events displayed on
the UI after the search is evaluated/computed.

max_events_per_bucket = <integer>
* For searches with status_buckets>0 this will limit the number of
events
  retrieved per timeline bucket.
* Defaults to 1000 in code.

truncate_report = [1|0]
* Specifies whether or not to apply the max_count limit to report
output.
* Defaults to false (0).

min_prefix_len = <integer>
* The minimum length of a prefix before a * to ask the index about.
* Defaults to 1.

cache_ttl = <integer>
* The length of time to persist search cache entries (in seconds).
* Defaults to 300.

max_results_perchunk = <integer>
* Maximum results per call to search (in dispatch), must be less than or
equal
  to maxresultrows.
* Defaults to 2500

min_results_perchunk = <integer>
* Minimum results per call to search (in dispatch), must be less than or
equal
```

```
  to max_results_perchunk.
* Defaults to 100

max_rawsize_perchunk = <integer>
* Maximum raw size of results per call to search (in dispatch).
* 0 = no limit.
* Defaults to 100000000 (100MB)
* Not affected by chunk_multiplier

target_time_perchunk = <integer>
* Target duration of a particular call to fetch search results in ms.
* Defaults to 2000

long_search_threshold = <integer>
* Time in seconds until a search is considered "long running".
* Defaults to 2

chunk_multiplier = <integer>
* max_results_perchunk, min_results_perchunk, and target_time_perchunk
are
  multiplied by this for a long running search.
* Defaults to 5

min_freq = <number>
* Minimum frequency of a field required for including in the /summary
endpoint
  as a fraction (>=0 and <=1).
* Defaults is 0.01 (1%)

reduce_freq = <integer>
* Attempt to reduce intermediate results every how many chunks (0 =
never).
* Defaults to 10

reduce_duty_cycle = <number>
* The maximum time to spend doing reduce, as a fraction of total search
time
* Must be > 0.0 and < 1.0
* Defaults to 0.25

preview_duty_cycle = <number>
* The maximum time to spend generating previews, as a fraction of total
search time
* Must be > 0.0 and < 1.0
* Defaults to 0.25

min_preview_period = <integer>
* This is the minimum time in seconds required between previews, used to
limit cases where
  the interval calculated using the preview_duty_cycle parameter is
very small, indicating
  that previews should be run frequently.
```

```
* Defaults to 1.

max_preview_period = <integer>
* This is the maximum time, in seconds, between previews. Used with the
preview interval that
  is calculated with the preview_duty_cycle parameter. '0' indicates
unlimited.
* Defaults to 0.

results_queue_min_size = <integer>
* The minimum size for the queue of results that will be kept from peers
for
  processing on the search head.
* The queue will be the max of this and the number of peers providing
results.
* Defaults to 10

dispatch_quota_retry = <integer>
* The maximum number of times to retry to dispatch a search when the
quota has
  been reached.
* Defaults to 4

dispatch_quota_sleep_ms = <integer>
* Milliseconds between retrying to dispatch a search if a quota has been
  reached.
* Retries the given number of times, with each successive wait 2x longer
than
  the previous.
* Defaults to 100

base_max_searches = <int>
* A constant to add to the maximum number of searches, computed as a
multiplier
  of the CPUs.
* Defaults to 6

max_searches_per_cpu = <int>
* The maximum number of concurrent historical searches per CPU. The
system-wide
  limit of historical searches is computed as:
  max_hist_searches =  max_searches_per_cpu x number_of_cpus +
base_max_searches
* Note: the maximum number of real-time searches is computed as:
  max_rt_searches = max_rt_search_multiplier x max_hist_searches
* Defaults to 1

max_rt_search_multiplier = <decimal number>
* A number by which the maximum number of historical searches is
multiplied to
  determine the maximum number of concurrent real-time searches
* Note: the maximum number of real-time searches is computed as:
```

```
  max_rt_searches = max_rt_search_multiplier x max_hist_searches
* Defaults to 1

max_macro_depth = <int>
* Max recursion depth for macros.
* Considered a search exception if macro expansion doesn't stop after
this many
  levels.
* Must be greater than or equal to 1.
* Default is 100

max_subsearch_depth = <int>
* max recursion depth for subsearch
* considered a search exception if subsearch doesn't stop after this
many levels

realtime_buffer = <int>
* Maximum number of accessible events to keep for real-time searches
from
  Splunk Web.
* Acts as circular buffer once this limit is reached
* Must be greater than or equal to 1
* Default is 10000

stack_size = <int>
* The stack size (in bytes) of the thread executing the search.
* Defaults to 4194304  (4 MB)

status_cache_size = <int>
* The number of search job status data splunkd can cache in RAM. This
cache
  improves performance of the jobs endpoint
* Defaults to 10000

timeline_freq = <timespan> or <ratio>
* Minimum amount of time between timeline commits.
* If specified as a number < 1 (and > 0), minimum time between commits
is
  computed as a ratio of the amount of time that the search has been
running.
* defaults to 0 seconds

preview_freq = <timespan> or <ratio>
* Minimum amount of time between results preview updates.
* If specified as a number < 1 (and > 0), minimum time between previews
is
  computed as a ratio of the amount of time that the search has been
running,
  or as a ratio of the length of the time window for real-time windowed
  searches.
* Defaults to ratio of 0.05
```

```
max_combiner_memevents = <int>
* Maximum size of in-memory buffer for search results combiner, in terms
of
  number of events.
* Defaults to 50000 events.

replication_period_sec  = <int>
* The minimum amount of time in seconds between two successive bundle
  replications.
* Defaults to 60

replication_file_ttl = <int>
* The TTL (in seconds) of bundle replication tarballs, i.e. *.bundle
files.
* Defaults to 600 (10m)

sync_bundle_replication = [0|1|auto]
* Flag indicating whether configuration file replication blocks searches
or is
  run asynchronously
* When setting this flag to auto Splunk will choose to use asynchronous
  replication if and only if all the peers support async bundle
replication,
  otherwise it will fall back into sync replication.
* Defaults to auto

rr_min_sleep_ms = <int>
* Minimum time to sleep when reading results in round-robin mode when no
data
  is available.
* Defaults to 10.

rr_max_sleep_ms = <int>
* Maximum time to sleep when reading results in round-robin mode when no
data
  is available.
* Defaults to 1000

rr_sleep_factor = <int>
* If no data is available even after sleeping, increase the next sleep
interval
  by this factor.
* defaults to 2

fieldstats_update_freq = <number>
* How often to update the field summary statistics, as a ratio to the
elapsed
  run time so far.
* Smaller values means update more frequently.  0 means as frequently as
  possible.
* Defaults to 0
```

```
fieldstats_update_maxperiod = <number>
* Maximum period for updating field summary statistics in seconds
* 0 means no maximum, completely dictated by current_run_time *
  fieldstats_update_freq
* Fractional seconds are allowed.
* defaults to 60

timeline_events_preview = <bool>
* Set timeline_events_preview to "true" to display events in the Search
app as
  the events are scanned, including events that are in-memory and not
yet committed,
  instead of waiting until all of the events are scanned to see the
search results.
* When set to "true", you will not be able to expand the event
information in the
  event viewer until events are committed.
* When set to "false", events are displayed only after the events are
committed
  (the events are written to the disk).
* This setting might increase disk usage to temporarily save uncommitted
events while
  the search is running. Additionally, search performance might be
impacted.
* Defaults to false.

remote_timeline = [0|1]
* If true, allows the timeline to be computed remotely to enable better
  map/reduce scalability.
* defaults to true (1).

remote_timeline_prefetch = <int>
* Each peer should proactively send at most this many full events at
the
  beginning
* Defaults to 100.

remote_timeline_parallel_fetch = <bool>
* Connect to multiple peers at the same time when fetching remote
events?
* Defaults to true

remote_timeline_min_peers = <int>
* Minimum search peers for enabling remote computation of timelines.
* Defaults to 1 (1).

remote_timeline_fetchall = [0|1]
* If set to true (1), Splunk fetches all events accessible through the
timeline from the remote
  peers before the job is considered done.
  * Fetching of all events may delay the finalization of some searches,
typically those running in
```

```
    verbose mode from the main Search view in Splunk Web.
  * This potential performance impact can be mitigated by lowering the
max_events_per_bucket
    settings.
* If set to false (0), the search peers may not ship all matching
events to the search-head,
  particularly if there is a very large number of them.
   * Skipping the complete fetching of events back to the search head
will result in prompt search
     finalization.
   * Some events may not be available to browse in the UI.
* This setting does *not* affect the accuracy of search results computed
by reporting searches.
* Defaults to true (1).

remote_timeline_thread = [0|1]
* If true, uses a separate thread to read the full events from remote
peers if
  remote_timeline is used and remote_timeline_fetchall is set to true.
(Has no
  effect if remote_timeline or remote_timeline_fetchall is false).
* Defaults to true (1).

remote_timeline_max_count = <int>
* Maximum number of events to be stored per timeline bucket on each
search
  peer.
* Defaults to 10000

remote_timeline_max_size_mb = <int>
* Maximum size of disk that remote timeline events should take on each
peer
* If limit is reached, a DEBUG message is emitted (and should be visible
from
  job inspector/messages
* Defaults to 100

remote_timeline_touchperiod = <number>
* How often to touch remote timeline artifacts to keep them from being
deleted
  by the remote peer, while a search is running.
* In seconds, 0 means never.  Fractional seconds are allowed.
* Defaults to 300.

remote_timeline_connection_timeout = <int>
* Connection timeout in seconds for fetching events processed by remote
peer
  timeliner.
* Defaults to 5.

remote_timeline_send_timeout = <int>
* Send timeout in seconds for fetching events processed by remote peer
```

```
  timeliner.
* Defaults to 10.

remote_timeline_receive_timeout = <int>
* Receive timeout in seconds for fetching events processed by remote
peer
  timeliner.
* Defaults to 10.

remote_event_download_initialize_pool = <int>
* Size of thread pool responsible for initiating the remote event fetch.
* Defaults to 5.

remote_event_download_finalize_pool = <int>
* Size of thread pool responsible for writing out the full remote
events.
* Defaults to 5.

remote_event_download_local_pool = <int>
* Size of thread pool responsible for reading full local events.
* Defaults to 5.

default_allow_queue = [0|1]
* Unless otherwise specified via REST API argument should an async job
spawning
  request be queued on quota violation (if not, an http error of server
too
  busy is returned)
* Defaults to true (1).

queued_job_check_freq = <number>
* Frequency with which to check queued jobs to see if they can be
started, in
  seconds
* Fractional seconds are allowed.
* Defaults to 1.

enable_history = <bool>
* Enable keeping track of searches?
* Defaults to true

max_history_length = <int>
* Max number of searches to store in history (per user/app)
* Defaults to 1000

allow_inexact_metasearch = <bool>
* Should a metasearch that is inexact be allow.  If so, an INFO message
will be
  added to the inexact metasearches.  If not, a fatal exception will
occur at
  search parsing time.
* Defaults to false
```

```
indexed_as_exact_metasearch = <bool>
* Should we allow a metasearch to treat <field>=<value> the same as
  <field>::<value> if <field> is an indexed field.  Allowing this will
allow a
  larger set of metasearches when allow_inexact_metasearch is set to
false.
  However, some of these searches may be inconsistent with the results
of doing
  a normal search.
* Defaults to false

dispatch_dir_warning_size = <int>
* The number of jobs in the dispatch directory when to issue a bulletin
message
  warning that performance could be impacted
* Defaults to 5000

allow_reuse = <bool>
* Allow normally executed historical searches to be implicitly re-used
for
  newer requests if the newer request allows it?
* Defaults to true

track_indextime_range = <bool>
* Track the _indextime range of returned search results?
* Defaults to true

reuse_map_maxsize = <int>
* Maximum number of jobs to store in the reuse map
* Defaults to 1000

status_period_ms = <int>
* The minimum amount of time, in milliseconds, between successive
  status/info.csv file updates
* This ensures search does not spend significant time just updating
these
  files.
  * This is typically important for very large number of search peers.
  * It could also be important for extremely rapid responses from search
peers,
    when the search peers have very little work to do.
* Defaults to 1000 (1 second)

search_process_mode = auto | traditional | debug <debugging-command>
[debugging-args ...]
* Control how search processes are started
* When set to "traditional", Splunk initializes each search process
completely from scratch
* When set to a string beginning with "debug", Splunk routes searches
through
  the given command, allowing the user the to "plug in" debugging tools
```

```
    * The <debugging-command> must reside in one of
        * $SPLUNK_HOME/etc/system/bin/
        * $SPLUNK_HOME/etc/apps/$YOUR_APP/bin/
        * $SPLUNK_HOME/bin/scripts/
    * Splunk will pass <debugging-args>, followed by the search command
it
      would normally run, to <debugging-command>
    * For example, given:
        search_process_mode = debug
$SPLUNK_HOME/bin/scripts/search-debugger.sh 5
      Splunk will run a command that looks generally like:
        $SPLUNK_HOME/bin/scripts/search-debugger.sh 5 splunkd search
--id=... --maxbuckets=... --ttl=... [...]
* Defaults to "auto"

max_searches_per_process = <int>
* On UNIX we can run more that one search per process; after a search
  completes its process can wait for another search to be started and
  let itself be reused
* When set to 1 (or 0), we'll never reuse a process
* When set to a negative value, we won't limit the number of searches a
  process can run
* When set to a number larger than one, we will let the process run
  up to that many searches before exiting
* Defaults to 500
* Has no effect on Windows, or if search_process_mode is not "auto"

max_time_per_process = <number>
* When running more than one search per process, this limits how much
  time a process can accumulate running searches before it must exit
* When set to a negative value, we won't limit the amount of time a
  search process can spend running
* Defaults to 300.0 (seconds)
* Has no effect on Windows, if search_process_mode is not "auto", or
  if max_searches_per_process is set to 0 or 1
* NOTE: a search can run longer than this without being terminated, this
  ONLY prevents that process from being used to run more searches
afterwards.

process_max_age = <number>
* When running more than one search per process, don't reuse a process
  if it is older than this number of seconds
* When set to a negative value, we won't limit the age of a search
process
* This is different than "max_time_per_process" because it includes
time
  the process spent idle
* Defaults to 7200.0 (seconds)
* Has no effect on Windows, if search_process_mode is not "auto", or
  if max_searches_per_process is set to 0 or 1
* NOTE: a search can run longer than this without being terminated, this
  ONLY prevents that process from being used to run more searches
```

```
afterwards.

idle_process_reaper_period = <number>
* When allowing more than one search to run per process, we'll
periodically
  check if we have too many idle search processes
* Defaults to 30.0 (seconds)
* Has no effect on Windows, if search_process_mode is not "auto", or
  if max_searches_per_process is set to 0 or 1

process_min_age_before_user_change = <number>
* When allowing more than one search to run per process, we'll try to
reuse
  an idle process that last ran a search by the same Splunk user
* If no such idle process exists, we'll try using a process from a
  different user, but only if it has been idle for at least this long
* When set to zero, we'll always allow an idle process to be reused by
  any Splunk user
* When set to a negative value, we'll only allow a search process to be
  used by same Splunk user each time
* Defaults to 4.0 (seconds)
* Has no effect on Windows, if search_process_mode is not "auto", or
  if max_searches_per_process is set to 0 or 1

launcher_threads = <int>
* When allowing more than one search to run per process, we'll run this
many
  server threads to manage those processes
* Defaults to -1 (meaning pick a value automatically)
* Has no effect on Windows, if search_process_mode is not "auto", or
  if max_searches_per_process is set to 0 or 1

launcher_max_idle_checks = <int>
* When allowing more than one search to run per process, we'll try to
find
  an appropriate idle process to use
* This controls how many idle processes we will inspect before giving up
  and starting a new one
* When set to a negative value, we'll inspect every eligible idle
process
* Defaults to 5
* Has no effect on Windows, if search_process_mode is not "auto", or
  if max_searches_per_process is set to 0 or 1

max_old_bundle_idle_time = <number>
* When reaping idle search processes, allow one to be reaped if it is
not
  configured with the most recent configuration bundle, and its bundle
  hasn't been used in at least this long
* When set to a negative value, we won't reap idle processes sooner than
  normal if they might be using an older configuration bundle
* Defaults to 5.0 (seconds)
```

```
* Has no effect on Windows, if search_process_mode is not "auto", or
  if max_searches_per_process is set to 0 or 1

idle_process_cache_timeout = <number>
* When a search process is allowed to run more than one search, it can
  cache some data between searches
* If a search process is idle for this long, take the opportunity to
purge
  some older data from these caches
* When set to a negative value, we won't do any purging based on how
long
  the search process is idle
* When set to zero, we'll always purge no matter if we're kept idle or
not
* Defaults to 0.5 (seconds)
* Has no effect on Windows, if search_process_mode is not "auto", or
  if max_searches_per_process is set to 0 or 1

idle_process_cache_search_count = <int>
* When a search process is allowed to run more than one search, it can
  cache some data between searches
* If a search process has run this many searches without purging older
  data from the cache, do it even if the "idle_process_cache_timeout"
has
  not been hit
* When set to a negative value, we won't purge no matter how many
  searches are run
* Defaults to 8
* Has no effect on Windows, if search_process_mode is not "auto", or
  if max_searches_per_process is set to 0 or 1

idle_process_regex_cache_hiwater = <int>
* When a search process is allowed to run more than one search, it can
  cache compiled regex artifacts
* If that cache grows to larger than this number of entries we'll try
  purging some older ones
* Normally the above "idle_process_cache_*" settings will take care of
  keeping the cache a reasonable size.  This setting is to prevent the
  cache from growing extremely large during a single large search
* When set to a negative value, we won't purge this cache based on its
size
* Defaults to 2500
* Has no effect on Windows, if search_process_mode is not "auto", or
  if max_searches_per_process is set to 0 or 1

fetch_remote_search_log = [enabled|disabledSavedSearches|disabled]
* enabled: all remote search logs will be downloaded barring the oneshot
search
* disabledSavedSearches: download all remote logs other than saved
search logs
  and oneshot search logs
* disabled: irrespective of the search type all remote search log
```

```
download
  functionality will be disabled
* Defaults to disabledSavedSearches
* The previous values:[true|false] are still supported but not
recommended for use
* The previous value of true maps to the current value of enabled
* The previous value of false maps to the current value of disabled

load_remote_bundles = <bool>
* On a search peer, allow remote (search head) bundles to be loaded in
splunkd.
* Defaults to false.

use_dispatchtmp_dir = <bool>
* Whether to use the dispatchtmp directory for temporary search time
files
  (write temporary files to a different directory from a job's dispatch
  directory).
* Temp files would be written to
$SPLUNK_HOME/var/run/splunk/dispatchtmp/<sid>/
* In search head pooling performance can be improved by mounting
disaptchtmp to
  the local file system.
* Defaults to true if search head pooling is enabled, false otherwise

check_splunkd_period = <number>
* Amount of time, in seconds, that determines how frequently the search
process
* (when running a real-time search) checks whether it's parent process
  (splunkd) is running or not.
* Fractional seconds are allowed.
* Defaults to 60

allow_batch_mode = <bool>
* Whether or not to allow the use of batch mode which searches in disk
based
  batches in a time insensitive manner.
* In distributed search environments, this setting is used on the search
head.
* Defaults to true

batch_search_max_index_values = <int>
* When using batch mode this limits the number of event entries read
from the
  index file. These entries are small approximately 72 bytes. However
batch
  mode is more efficient when it can read more entries at once.
* Setting this value to a smaller number can lead to slower search
performance.
* A balance needs to be struck between more efficient searching in batch
mode
* and running out of memory on the system with concurrently running
```

```
searches.
* Defaults to 10000000


* These settings control the periodicity of retries to search peers in
the
  event of failure. (Connection errors, and others.) The interval exists
  between failure and first retry, as well as successive retries in the
event
  of further failures.

batch_retry_min_interval = <int>
* When batch mode attempts to retry the search on a peer that failed
wait at
  least this many seconds
* Default to 5

batch_retry_max_interval = <int>
* When batch mode attempts to retry the search on a peer that failed
wait at
  most this many seconds
* Default to 300

batch_retry_scaling = <double>
* After a retry attempt fails increase the time to wait before trying
again by
  this scaling factor (Value should be > 1.0)
* Default 1.5

batch_wait_after_end = <int>
* Batch mode considers the search ended(finished) when all peers without
  communication failure have explicitly indicated that they are
complete; eg
  have delivered the complete answer.  After the search is at an end,
batch
  mode will continue to retry with lost-connection peers for this many
seconds.
* Default 900

batch_search_max_pipeline = <int>
* Controls the number of search pipelines launched at the indexer during
batch search.
* Default value is set to one pipeline.
* Increasing the number of search pipelines should help improve search
performance
* but there will be an increase in thread and memory usage.

batch_search_max_results_aggregator_queue_size = <int>
* Controls the size of the search results queue to which all the search
pipelines dump the processed search results.
* Default size is set to 100MB.
* Increasing the size can lead to performance gain where as decreasing
```

```
can reduce search performance.
* Do not set this parameter to zero.

batch_search_max_serialized_results_queue_size = <int>
* Controls the size of the serialized results queue from which the
serialized search results are transmitted.
* Default size is set to 100MB.
* Increasing the size can lead to performance gain where as decreasing
can reduce search performance.
* Do not set this parameter to zero.

write_multifile_results_out = <bool>
* At the end of the search, if results are in multiple files, write out
the
  multiple files to results_dir directory, under the search results
directory.
* This will speed up post-processing search, since the results will
already be
  split into appropriate size files.
* Default true

enable_cumulative_quota = <bool>
* Whether to enforce cumulative role based quotas
* Default false

remote_reduce_limit = <unsigned long>
* The number of results processed by a streaming search before we force
a reduce
* Note: this option applies only if the search is run with
--runReduce=true
  (currently on Hunk does this)
* Note: a value of 0 is interpreted as unlimited
* Defaults to: 1000000

max_workers_searchparser = <int>
* The number of worker threads in processing search result when using
round
  robin policy.
* default 5

max_chunk_queue_size = <int>
* The maximum size of the chunk queue
* default 10000000

max_tolerable_skew = <positive integer>
* Absolute value of the largest timeskew in seconds that we will
tolerate
  between the native clock on the searchhead and the native clock on the
peer
  (independent of time-zone).
* If this timeskew is exceeded we will log a warning. This estimate is
  approximate and tries to account for network delays.
```

```
addpeer_skew_limit = <positive integer>
* Absolute value of the largest time skew in seconds that is allowed
when configuring
  a search peer from a search head, independent of time.
* If the difference in time (skew) between the search head and the peer
is greater
  than this limit, the search peer will not be added.
* This is only relevant to manually added peers; currently this setting
has no effect
  upon index cluster search peers.

unified_search = <bool>
* Turns on/off unified search for hunk archiving, defaults to false if
not
  specified.

enable_memory_tracker = <bool>
* If memory tracker is disabled, search won't be terminated even if it
exceeds the memory limit.
* Must be set to <true> if you want to enable
search_process_memory_usage_threshold or
* search_process_memory_usage_percentage_threshold
* By default false.

search_process_memory_usage_threshold = <double>
* To be active, this setting requires setting: enable_memory_tracker =
true
* Signifies the maximum memory in MB the search process can consume in
RAM.
* Search processes violating the threshold will be terminated.
* If the value is set to zero, then splunk search processes are allowed
  to grow unbounded in terms of in memory usage.
* The default value is set to 4000MB or 4GB.

search_process_memory_usage_percentage_threshold = <float>
* To be active, this setting requires setting: enable_memory_tracker =
true
* Signifies the percentage of the total memory the search process is
entitled to consume.
* Any time the search process violates the threshold percentage the
process will be brought down.
* If the value is set to zero, then splunk search processes are allowed
to grow unbounded
  in terms of percentage memory usage.
* The default value is set to 25%.
* Any number set larger than 100 or less than 0 will be discarded and
the default value will be used.

enable_datamodel_meval = <bool>
* Enable concatenation of successively occurring evals into a single
  comma separated eval during generation of datamodel searches.
```

```
* default true

do_not_use_summaries = <bool>
* Do not use this setting without working in tandem with Splunk
support.
* This setting is a very narrow subset of summary_mode=none. When set
to true, this
  setting disables some functionality that is necessary for report
acceleration.
  In particular, when set to true, search processes will no longer query
the main
  splunkd's /admin/summarization endpoint for report acceleration
summary ids.
* In certain narrow use-cases this may improve performance if report
acceleration
  (savedsearches.conf:auto_summarize) is not in use by lowering the
main splunkd's
  process overhead.
* Defaults to false.

unified_search = <bool>
* Enables the unified search feature.
* Defaults to false.

force_saved_search_dispatch_as_user = <bool>
* Specifies whether to overwrite the 'dispatchAs' value.
* If set to 'true', the 'dispatchAs' value is overwritten by 'user'
regardless
  of the 'user | owner' value in the savedsearches.conf file.
* If set to 'false', the value in the savedsearches.conf file is used.
* User may want to set this to effectively disable dispatchAs = owner
for
  the entire install, if that more closely aligns with security goals.
* Defaults to false.

-- Unsupported [search] settings: --

enable_status_cache = <bool>
* This is not a user tunable setting.  Do not use this setting without
  working in tandem with Splunk personnel.  This setting is not tested
at
  non-default.
* This controls whether the status cache is used, which caches
information
  about search jobs (and job artifacts) in memory in main splunkd.
* Normally this cacheing is enabled and assists performance. However,
when
  using Search Head Pooling, artifacts in the shared storage location
will be
  changed by other search heads, so this cacheing is disabled.
* Explicit requests to jobs endpoints , eg /services/search/jobs/<sid>
are
```

```
  always satisfied from disk, regardless of this setting.
* Defaults to true; except in Search Head Pooling environments where it
  defaults to false.

status_cache_in_memory_ttl = <positive integer>
* This setting has no effect unless search head pooling is enabled, AND
  enable_status_cache has been set to true.
* This is not a user tunable setting.  Do not use this setting without
working
  in tandem with Splunk personnel. This setting is not tested at
non-default.
* If set, controls the number of milliseconds which a status cache entry
may be
  used before it expires.
* Defaults to 60000, or 60 seconds.
```

[realtime]

```
[realtime]
# Default options for indexer support of real-time searches
# These can all be overridden for a single search via REST API
arguments

local_connect_timeout = <int>
* Connection timeout for an indexer's search process when connecting to
that
  indexer's splunkd (in seconds)
* Defaults to 5

local_send_timeout = <int>
* Send timeout for an indexer's search process when connecting to that
  indexer's splunkd (in seconds)
* Defaults to 5

local_receive_timeout = <int>
* Receive timeout for an indexer's search process when connecting to
that
  indexer's splunkd (in seconds)
* Defaults to 5

queue_size = <int>
* Size of queue for each real-time search (must be >0).
* Defaults to 10000

blocking = [0|1]
* Specifies whether the indexer should block if a queue is full.
* Defaults to false

max_blocking_secs = <int>
* Maximum time to block if the queue is full (meaningless if blocking =
```

```
false)
* 0 means no limit
* Default to 60

indexfilter = [0|1]
* Specifies whether the indexer should prefilter events for efficiency.
* Defaults to true (1).

default_backfill = <bool>
* Specifies if windowed real-time searches should backfill events
* Defaults to true

enforce_time_order = <bool>
* Specifies if real-time searches should ensure that events are sorted
in
  ascending time order (the UI will automatically reverse the order that
it
  display events for real-time searches so in effect the latest events
will be
  first)
* Defaults to true

disk_usage_update_period = <number>
* Specifies how frequently (in seconds) should the search process
estimate the
  artifact disk usage.
* Fractional seconds are allowed.
* Defaults to 10

indexed_realtime_use_by_default = <bool>
* Should we use the indexedRealtime mode by default
* Precedence: SearchHead
* Defaults to false

indexed_realtime_disk_sync_delay = <int>
* After indexing there is a non-deterministic period where the files on
disk
  when opened by other programs might not reflect the latest flush to
disk,
  particularly when a system is under heavy load.
* This settings controls the number of seconds to wait for disk flushes
to
  finish when using indexed/continuous/psuedo realtime search so that
we see
  all of the data.
* Precedence: SearchHead overrides Indexers
* Defaults to 60

indexed_realtime_default_span = <int>
* An indexed realtime search is made up of many component historical
searches
  that by default will span this many seconds. If a component search is
```

```
not
  completed in this many seconds the next historical search will span
the extra
  seconds. To reduce the overhead of running an indexed realtime search
you can
  change this span to delay longer before starting the next component
  historical search.
* Precedence: Indexers
* Defaults to 1

indexed_realtime_maximum_span = <int>
* While running an indexed realtime search, if the component searches
regularly
  take longer than indexed_realtime_default_span seconds, then indexed
realtime
  search can fall more than indexed_realtime_disk_sync_delay seconds
behind
  realtime. Use this setting to set a limit after which we will drop
data to
  return back to catch back up to the specified delay from realtime, and
only
  search the default span of seconds.
* Precedence: API overrides SearchHead overrides Indexers
* Defaults to 0 (unlimited)

indexed_realtime_cluster_update_interval = <int>
* While running an indexed realtime search, if we are on a cluster we
need to
  update the list of allowed primary buckets. This controls the interval
that
  we do this. And it must be less than the
indexed_realtime_disk_sync_delay. If
  your buckets transition from Brand New to warm in less than this time
indexed
  realtime will lose data in a clustered environment.
* Precedence: Indexers
* Default: 30

alerting_period_ms = <int>
* This limits the frequency that we will trigger alerts during a
realtime search
* A value of 0 means unlimited and we will trigger an alert for every
batch of
  events we read in dense realtime searches with expensive alerts this
can
  overwhelm the alerting system.
* Precedence: Searchhead
* Default: 0
```

[slc]

```
[slc]
maxclusters = <integer>
* Maximum number of clusters to create.
* Defaults to 10000.
```

[findkeywords]

```
[findkeywords]
maxevents = <integer>
* Maximum number of events used by the findkeywords command and the
Patterns tab.
* Defaults to 50000.
```

[sort]

```
[sort]
maxfiles = <integer>
* Maximum files to open at once.  Multiple passes are made if the number
of
  result chunks exceeds this threshold.
* Defaults to 64.
```

[stats|sistats]

```
[stats|sistats]
maxmem_check_freq = <integer>
* How frequently to check to see if we are exceeding the in memory data
  structure size limit as specified by max_mem_usage_mb, in rows
* Defaults to 50000 rows

maxresultrows = <integer>
* Maximum number of rows allowed in the process memory.
* When the search process exceeds max_mem_usage_mb and maxresultrows,
data is
  spilled out to the disk
* If not specified, defaults to searchresults::maxresultrows (which is
by default 50000).

maxvalues = <integer>
* Maximum number of values for any field to keep track of.
* Defaults to 0 (unlimited).

maxvaluesize = <integer>
```

```
* Maximum length of a single value to consider.
* Defaults to 0 (unlimited).

# rdigest is a data structure used to compute approximate order
statistics
# (such as median and percentiles) using sublinear space.

rdigest_k = <integer>
* rdigest compression factor
* Lower values mean more compression
* After compression, number of nodes guaranteed to be greater than or
equal to
  11 times k.
* Defaults to 100, must be greater than or equal to 2

rdigest_maxnodes = <integer>
* Maximum rdigest nodes before automatic compression is triggered.
* Defaults to 1, meaning automatically configure based on k value

max_stream_window = <integer>
* For the streamstats command, the maximum allow window size
* Defaults to 10000.

max_valuemap_bytes = <integer>
* For sistats command, the maximum encoded length of the valuemap, per
result
  written out
* If limit is exceeded, extra result rows are written out as needed.  (0
= no
  limit per row)
* Defaults to 100000.

perc_method = nearest-rank|interpolated
* Which method to use for computing percentiles (and medians=50
percentile).
  * nearest-rank picks the number with 0-based rank R =
    floor((percentile/100)*count)
  * interpolated means given F = (percentile/100)*(count-1),
    pick ranks R1 = floor(F) and R2 = ceiling(F).
    Answer = (R2 * (F - R1)) + (R1 * (1 - (F - R1)))
* See wikipedia percentile entries on nearest rank and "alternative
methods"
* Defaults to interpolated

approx_dc_threshold = <integer>
* When using approximate distinct count (i.e. estdc(<field>) in
  stats/chart/timechart), do not use approximated results if the actual
number
  of distinct values is less than this number
* Defaults to 1000

dc_digest_bits = <integer>
```

```
* 2^<integer> bytes will be size of digest used for approximating
distinct count.
* Defaults to 10 (equivalent to 1KB)
* Must be >= 8 (128B) and <= 16 (64KB)

natural_sort_output = <bool>
* Do a natural sort on the output of stats if output size is <=
maxresultrows
* Natural sort means that we sort numbers numerically and non-numbers
  lexicographically
* Defaults to true

list_maxsize = <int>
* Maximum number of list items to emit when using the list() function
  stats/sistats
* Defaults to 100

sparkline_maxsize = <int>
* Maximum number of elements to emit for a sparkline
* Defaults to value of the list_maxsize setting

sparkline_time_steps = <time-step-string>
* Specify a set of time steps in order of decreasing granularity. Use an
integer and
* one of the following time units to indicate each step.
** s = seconds
** m = minutes
** h = hours
** d = days
** month
* Defaults to: 1s,5s,10s,30s,1m,5m,10m,30m,1h,1d,1month
* A time step from this list is selected based on the
<sparkline_maxsize> setting.
* The lowest <sparkline_time_steps> value that does not exceed the
maximum number
* of bins is used.
* Example:
** If you have the following configurations:
** <sparkline_time_steps> = 1s,5s,10s,30s,1m,5m,10m,30m,1h,1d,1month
** <sparkline_maxsize> = 100
** The timespan for 7 days of data is 604,800 seconds.
** Span = 604,800/<sparkline_maxsize>.
** If sparkline_maxsize = 100, then span = (604,800 / 100) = 60,480 sec
== 1.68 hours.
** The "1d" time step is used because it is the lowest value that does
not exceed
** the maximum number of bins.

default_partitions = <int>
* Number of partitions to split incoming data into for
parallel/multithreaded reduce
* Defaults to 1
```

```
partitions_limit = <int>
* Maximum number of partitions to split into that can be specified via
the
  'partitions' option.
* When exceeded, the number of partitions is reduced to this limit.
* Defaults to 100
```

[thruput]

```
[thruput]
maxKBps = <integer>
* If specified and not zero, this limits the speed through the thruput
processor
  in the ingestion pipeline to the specified rate in kilobytes per
second.
* To control the CPU load while indexing, use this to throttle the
number of
  events this indexer processes to the rate (in KBps) you specify.
* Note that this limit will be applied per ingestion pipeline. For more
information
  about multiple ingestion pipelines see parallelIngestionPipelines in
the
  server.conf.spec file.
* With N parallel ingestion pipelines the thruput limit across all of
the ingestion
  pipelines will be N * maxKBps.
```

[journal_compression]

```
[journal_compression]
threads = <integer>
* Specifies the maximum number of indexer threads which will be work on
  compressing hot bucket journal data.
* Defaults to the number of CPU threads of the host machine
* This setting does not typically need to be modified.
```

[top]

```
[top]
maxresultrows = <integer>
* Maximum number of result rows to create.
* If not specified, defaults to searchresults::maxresultrows (usually
50000).

maxvalues = <integer>
* Maximum number of distinct field vector values to keep track of.
```

```
* Defaults to 100000.

maxvaluesize = <integer>
* Maximum length of a single value to consider.
* Defaults to 1000.
```

[summarize]

```
[summarize]
hot_bucket_min_new_events = <integer>
* The minimum number of new events that need to be added to the hot
bucket
  (since last summarization)  before a new summarization can take place.
To
  disable hot bucket summarization set this value to a * large positive
number.
* Defaults to 100000

max_hot_bucket_summarization_idle_time = <unsigned int>
* Maximum amount of time, in seconds, a hot bucket can be idle after
which we summarize all the
  events even if there are not enough events (determined by
hot_bucket_min_new_events)
* Defaults to 900 seconds (or 15 minutes)

sleep_seconds = <integer>
* The amount of time to sleep between polling of summarization complete
status.
* Default to 5

stale_lock_seconds = <integer>
* The amount of time to have elapse since the mod time of a .lock file
before
  summarization considers * that lock file stale and removes it
* Default to 600

max_summary_ratio = <float>
* A number in the [0-1] range that indicates the maximum ratio of
  summary data / bucket size at which point the summarization of that
bucket,
  for the particular search, will be disabled. Use 0 to disable.
* Defaults to 0

max_summary_size = <int>
* Size of summary, in bytes, at which point we'll start applying the
  max_summary_ratio. Use 0 to disable.
* Defaults to 0

max_time = <int>
* The maximum amount of time, seconds, that a summary search process is
```

```
allowed
  to run. Use 0 to disable.
* Defaults to 0

indextime_lag = <unsigned int>
* The amount of lag time to give indexing to ensure that it has synced
any
  received events to disk. Effectively, the data that has been received
in the
  past indextime_lag will NOT be summarized.
* Do not change this value unless directed by Splunk support.
* Defaults to 90

max_replicated_hot_bucket_idle_time = <unsigned int>

* Maximum amount of time, in seconds, a replicated hot bucket can be
idle after which we won't
  apply indextime_lag.
* This applies to only idle replicated hot buckets. As soon as new
events start flowing
  in we will revert to the default behavior of applying indextime_lag
* Defaults to 3600 seconds
```

[transactions]

```
[transactions]
maxopentxn = <integer>
* Specifies the maximum number of not yet closed transactions to keep in
the
  open pool before starting to evict transactions.
* Defaults to 5000.

maxopenevents = <integer>
* Specifies the maximum number of events (which are) part of open
transactions
  before transaction eviction starts happening, using LRU policy.
* Defaults to 100000.
```

[inputproc]

```
[inputproc]
max_fd = <integer>
* Maximum number of file descriptors that a ingestion pipeline in Splunk
will keep open,
  to capture any trailing data from files that are written to very
slowly.
* Note that this limit will be applied per ingestion pipeline. For more
information
```

```
  about multiple ingestion pipelines see parallelIngestionPipelines in
the
  server.conf.spec file.
* With N parallel ingestion pipelines the maximum number of file
descriptors that
  can be open across all of the ingestion pipelines will be N * max_fd.
* Defaults to 100.

monitornohandle_max_heap_mb = <integer>
* Controls the maximum memory used by the Windows-specific modular input
  MonitorNoHandle.
* The memory of this input grows in size when the data being produced
  by applications writing to monitored files comes in faster than the
Splunk
  system can accept it.
* When set to 0, the heap size (memory allocated in the modular input)
can grow
  without limit.
* If this size is limited, and the limit is encountered, the input will
drop
  some data to stay within the limit.
* Defaults to 0.

time_before_close = <integer>
* MOVED.  This setting is now configured per-input in inputs.conf.
* Specifying this setting in limits.conf is DEPRECATED, but for now will
  override the setting for all monitor inputs.

tailing_proc_speed = <integer>
* REMOVED.  This setting is no longer used.

file_tracking_db_threshold_mb = <integer>
* This setting controls the trigger point at which the file tracking db
(also
  commonly known as the "fishbucket" or btree) rolls over.  A new
database is
  created in its place.  Writes are targeted at new db.  Reads are first
  targeted at new db, and we fall back to old db for read failures.  Any
reads
  served from old db successfully will be written back into new db.
* MIGRATION NOTE: if this setting doesn't exist, the initialization code
in
  splunkd triggers an automatic migration step that reads in the current
value
  for "maxDataSize" under the "_thefishbucket" stanza in indexes.conf
and
  writes this value into etc/system/local/limits.conf.

learned_sourcetypes_limit = <0 or positive integer>
* Limits the number of entries added to the learned app for performance
  reasons.
* If nonzero, limits two properties of data added to the learned app by
```

```
the
  file classifier. (Code specific to monitor:: stanzas that
auto-determines
  sourcetypes from content.)
  * The number of sourcetypes added to the learned app's props.conf file
will
    be limited to approximately this number.
  * The number of file-content fingerprints added to the learned app's
    sourcetypes.conf file will be limited to approximately this number.
* The tracking for uncompressed and compressed files is done separately,
so in
  some cases this value may be exceeded.
* This limit is not the recommended solution for auto-identifying
sourcetypes.
  The usual  best practices are to set sourcetypes in input stanzas, or
  alternatively to apply them based on filename pattern in props.conf
  [source::<pattern>] stanzas.
* Defaults to 1000.
```

[scheduler]

```
[scheduler]
saved_searches_disabled = <bool>
* Whether saved search jobs are disabled by the scheduler.
* Defaults to false.

max_searches_perc = <integer>
* The maximum number of searches the scheduler can run, as a percentage
of the
  maximum number of concurrent searches, see [search]
max_searches_per_cpu for
  how to set the system wide maximum number of searches.
* Defaults to 50.

max_searches_perc.<n> = <integer>
max_searches_perc.<n>.when = <cron string>
* The same as max_searches_perc but the value is applied only when the
cron
  string matches the current time.  This allows max_searches_perc to
have
  different values at different times of day, week, month, etc.
* There may be any number of non-negative <n> that progress from least
specific
  to most specific with increasing <n>.
* The scheduler looks in reverse-<n> order looking for the first match.
* If either these settings aren't provided at all or no "when" matches
the
  current time, the value falls back to the non-<n> value of
max_searches_perc.
```

```
auto_summary_perc = <integer>
* The maximum number of concurrent searches to be allocated for auto
  summarization, as a percentage of the concurrent searches that the
scheduler
  can run.
* Auto summary searches include:
  * Searches which generate the data for the Report Acceleration
feature.
  * Searches which generate the data for Data Model acceleration.
* Note: user scheduled searches take precedence over auto summary
searches.
* Defaults to 50.

auto_summary_perc.<n> = <integer>
auto_summary_perc.<n>.when = <cron string>
* The same as auto_summary_perc but the value is applied only when the
cron
  string matches the current time.  This allows auto_summary_perc to
have
  different values at different times of day, week, month, etc.
* There may be any number of non-negative <n> that progress from least
specific
  to most specific with increasing <n>.
* The scheduler looks in reverse-<n> order looking for the first match.
* If either these settings aren't provided at all or no "when" matches
the
  current time, the value falls back to the non-<n> value of
auto_summary_perc.

priority_runtime_factor = <double>
* The amount to scale the priority runtime adjustment by.
* Every search's priority is made higher (worse) by its typical running
time.
  Since many searches run in fractions of a second and the priority is
  integral, adjusting by a raw runtime wouldn't change the result;
therefore,
  it's scaled by this value.
* Defaults to 10.

priority_skipped_factor = <double>
* The amount to scale the skipped adjustment by.
* A potential issue with the priority_runtime_factor is that now
longer-running
  searches may get starved.  To balance this out, make a search's
priority
  lower (better) the more times it's been skipped.  Eventually, this
adjustment
  will outweigh any worse priority due to a long runtime. This value
controls
  how quickly this happens.
* Defaults to 1.
```

```
search_history_max_runtimes = <unsigned int>
* The number of runtimes kept for each search.
* Used to calculate historical typical runtime during search
prioritization.
* Defaults to 10.

search_history_load_timeout = <duration-specifier>
* The maximum amount of time to defer running continuous scheduled
searches
  while waiting for the KV Store to come up in order to load historical
data.
  This is used to prevent gaps in continuous scheduled searches when
splunkd
  was down.
* Use [<int>]<unit> to specify a duration; a missing <int> defaults to
1.
* Relevant units are: s, sec, second, secs, seconds, m, min, minute,
mins,
  minutes.
* For example: "60s" = 60 seconds, "5m" = 5 minutes.
* Defaults to 2m.

max_continuous_scheduled_search_lookback = <duration-specifier>
* The maximum amount of time to run missed continuous scheduled searches
for
  once Splunk comes back up in the event it was down.
* Use [<int>]<unit> to specify a duration; a missing <int> defaults to
1.
* Relevant units are: m, min, minute, mins, minutes, h, hr, hour, hrs,
hours,
  d, day, days, w, week, weeks, mon, month, months.
* For example: "5m" = 5 minutes, "1h" = 1 hour.
* A value of 0 means no lookback.
* Defaults to 24 hours.

introspection_lookback = <duration-specifier>
* The amount of time to "look back" when reporting introspection
statistics.
* For example: what is the number of dispatched searches in the last 60
minutes?
* Use [<int>]<unit> to specify a duration; a missing <int> defaults to
1.
* Relevant units are: m, min, minute, mins, minutes, h, hr, hour, hrs,
hours,
  d, day, days, w, week, weeks.
* For example: "5m" = 5 minutes, "1h" = 1 hour.
* Defaults to 1 hour.

max_action_results = <integer>
* The maximum number of results to load when triggering an alert action.
* Defaults to 50000
```

```
action_execution_threads = <integer>
* Number of threads to use to execute alert actions, change this number
if your
  alert actions take a long time to execute.
* This number is capped at 10.
* Defaults to 2

actions_queue_size = <integer>
* The number of alert notifications to queue before the scheduler
starts
  blocking, set to 0 for infinite size.
* Defaults to 100

actions_queue_timeout = <integer>
* The maximum amount of time, in seconds to block when the action queue
size is
  full.
* Defaults to 30

alerts_max_count = <integer>
* Maximum number of unexpired alerts information to keep for the alerts
  manager, when this number is reached Splunk will start discarding the
oldest
  alerts.
* Defaults to 50000

alerts_max_history = <integer>[s|m|h|d]
* Maximum time to search in the past for previously triggered alerts.
* splunkd uses this property to populate the Activity -> Triggered
Alerts page at startup.
* Defaults to 7 days.
* Values greater than the default may cause slowdown.

alerts_scoping = host|splunk_server|all
* Determines the scoping to use on the search to populate the triggered
alerts
  page. Choosing splunk_server will result in the search query
  using splunk_server=local, host will result in the search query using
  host=<search-head-host-name>, and all will have no scoping added to
the
  search query.
* Defaults to splunk_server.

alerts_expire_period = <integer>
* The amount of time between expired alert removal
* This period controls how frequently the alerts list is scanned, the
only
  benefit from reducing this is better resolution in the number of
alerts fired
  at the savedsearch level.
* Change not recommended.
* Defaults to 120.
```

```
persistance_period = <integer>
* The period (in seconds) between scheduler state persistance to disk.
The
  scheduler currently persists the suppression and fired-unexpired
alerts to
  disk.
* This is relevant only in search head pooling mode.
* Defaults to 30.

max_lock_files = <int>
* The number of most recent lock files to keep around.
* This setting only applies in search head pooling.

max_lock_file_ttl = <int>
* Time (in seconds) that must pass before reaping a stale lock file.
* Only applies in search head pooling.

max_per_result_alerts = <int>
* Maximum number of alerts to trigger for each saved search instance (or
  real-time results preview for RT alerts)
* Only applies in non-digest mode alerting. Use 0 to disable this limit
* Defaults to 500

max_per_result_alerts_time = <int>
* Maximum number of time to spend triggering alerts for each saved
search
  instance (or real-time results preview for RT alerts)
* Only applies in non-digest mode alerting. Use 0 to disable this limit.
* Defaults to 300

scheduled_view_timeout = <int>[s|m|h|d]
* The maximum amount of time that a scheduled view (pdf delivery) would
be
  allowed to render
* Defaults to 60m

concurrency_message_throttle_time = <int>[s|m|h|d]
* Amount of time controlling throttling between messages warning about
scheduler concurrency limits
*Defaults to 10m

shp_dispatch_to_slave = <bool>
* By default the scheduler should distribute jobs throughout the pool.
* Defaults to true

shc_role_quota_enforcement = <bool>
* When this is enabled, the following limits are enforced by the captain
for scheduled searches:
  - User role quotas are enforced globally.
    A given role can have (n *number_of_peers) searches running
cluster-wide,
```

```
    where n is the quota for that role as defined by srchJobsQuota and
    rtSrchJobsQuota on the captain
  - Maximum number of concurrent searches is enforced globally.
    This is (n * number_of_peers) where n is the max concurrent
searches on the captain
    (see max_searches_per_cpu for a description of how this is
computed).
    Concurrent searches include both scheduled searches and ad hoc
searches.
* Scheduled searches will therefore not have an enforcement of either of
the above
   on a per-member basis.
* Note that this doesn't control the enforcement of the scheduler quota.
  For a search head cluster, that is defined as (max_searches_perc *
number_of_peers)
  and is always enforced globally on the captain.
* Quota information is conveyed from the members to the captain. Network
delays
  can cause the quota calculation on the captain to vary from the actual
values
  in the members and may cause search limit warnings. This should clear
up as
  the information is synced.
* Defaults to false.

shc_local_quota_check = <bool>
* Enabling this enforces user role quota and maximum number of
  concurrent searches on a per-member basis.
* Cluster-wide scheduler quota is still enforced globally on the
captain.
* See shc_role_quota_enforcement for more details.
* Disabling this requires shc_role_quota_enforcement=true. Otherwise,
all
  quota checks will be skipped.
* Note that disabling this will also disable disk quota checks.
* Defaults to true.
```

[auto_summarizer]

```
[auto_summarizer]
cache_timeout = <integer>
* The amount of time, in seconds, to cache auto summary details and
search hash
  codes
* Defaults to 600 - 10 minutes

search_2_hash_cache_timeout = <integer>
* The amount of time, in seconds, to cache search hash codes
* Defaults to the value of cache_timeout i.e. 600 - 10 minutes
```

```
maintenance_period = <integer>
* The period of time, in seconds, that the auto summarization
maintenance
  happens
* Defaults to 1800 (30 minutes)

allow_event_summarization = <bool>
* Whether auto summarization of searches whose remote part returns
events
  rather than results will be allowed.
* Defaults to false

max_verify_buckets = <int>
* When verifying buckets, stop after verifying this many buckets if no
failures
  have been found
* 0 means never
* Defaults to 100

max_verify_ratio = <number>
* Maximum fraction of data in each bucket to verify
* Defaults to 0.1 (10%)

max_verify_bucket_time = <int>
* Maximum time to spend verifying each bucket, in seconds
* Defaults to 15 (seconds)

verify_delete = <bool>
* Should summaries that fail verification be automatically deleted?
* Defaults to false

max_verify_total_time = <int>
* Maximum total time in seconds to spend doing verification, regardless
if any
  buckets have failed or not
* Defaults to 0 (no limit)

max_run_stats = <int>
* Maximum number of summarization run statistics to keep track and
expose via
  REST.
* Defaults to 48

return_actions_with_normalized_ids = [yes|no|fromcontext]
* Report acceleration summaries are stored under a signature/hash which
can be
  regular or normalized.
    * Normalization improves the re-use of pre-built summaries but is
not
      supported before 5.0. This config will determine the default
value of how
       normalization works (regular/normalized)
```

```
    * Default value is "fromcontext", which would mean the end points
and
      summaries would be operating based on context.
* normalization strategy can also be changed via admin/summarization
REST calls
  with the "use_normalization"  parameter which can take the values
  "yes"/"no"/"fromcontext"

normalized_summaries = <bool>
* Turn on/off normalization of report acceleration summaries.
* Default = false and will become true in 6.0

detailed_dashboard = <bool>
* Turn on/off the display of both normalized and regular summaries in
the
  Report Acceleration summary dashboard and details.
* Default = false

shc_accurate_access_counts = <bool>
* Only relevant if you are using search head clustering
* Turn on/off to make acceleration summary access counts accurate on
the
  captain.
* by centralizing the access requests on the captain.
* Default = false
```

[show_source]

```
[show_source]
max_count = <integer>
* Maximum number of events accessible by show_source.
* The show source command will fail when more than this many events are
in the
  same second as the requested event.
* Defaults to 10000

max_timebefore = <timespan>
* Maximum time before requested event to show.
* Defaults to '1day' (86400 seconds)

max_timeafter = <timespan>
* Maximum time after requested event to show.
* Defaults to '1day' (86400 seconds)

distributed = <bool>
* Controls whether we will do a distributed search for show source to
get
  events from all servers and indexes
* Turning this off results in better performance for show source, but
events
```

```
  will only come from the initial server and index
* NOTE: event signing and verification is not supported in distributed
mode
* Defaults to true

distributed_search_limit = <unsigned int>
* Sets a limit on the maximum events we will request when doing the
search for
  distributed show source
* As this is used for a larger search than the initial non-distributed
show
  source, it is larger than max_count
* Splunk will rarely return anywhere near this amount of results, as we
will
  prune the excess results
* The point is to ensure the distributed search captures the target
event in an
  environment with many events
* Defaults to 30000
```

[typeahead]

```
[typeahead]
maxcount = <integer>
* Maximum number of typeahead results to find.
* Defaults to 1000

use_cache = [0|1]
* Specifies whether the typeahead cache will be used if use_cache is not
  specified in the command line or endpoint.
* Defaults to true.

fetch_multiplier = <integer>
* A multiplying factor that determines the number of terms to fetch from
the
  index, fetch = fetch_multiplier x count.
* Defaults to 50

cache_ttl_sec = <integer>
* How long the typeahead cached results are valid, in seconds.
* Defaults to 300.

min_prefix_length = <integer>
* The minimum string prefix after which to provide typeahead.
* Defaults to 1.

max_concurrent_per_user = <integer>
* The maximum number of concurrent typeahead searches per user. Once
this
  maximum is reached only cached typeahead results might be available
```

```
* Defaults to 3.
```

[typer]

```
[typer]
maxlen = <int>
* In eventtyping, pay attention to first <int> characters of any
attribute
  (such as _raw), including individual tokens. Can be overridden by
supplying
  the typer operator with the argument maxlen (for example,
  "|typer maxlen=300").
* Defaults to 10000.
```

[authtokens]

```
[authtokens]
expiration_time = <integer>
* Expiration time of auth tokens in seconds.
* Defaults to 3600
```

[sample]

```
[sample]
maxsamples = <integer>
* Defaults to 10000

maxtotalsamples = <integer>
* Defaults to 100000
```

[metadata]

```
[metadata]
maxresultrows = <integer>
* The maximum number of results in a single chunk fetched by the
metadata
  command
* A smaller value will require less memory on the search head in setups
with
  large number of peers and many metadata results, though, setting this
too
  small will decrease the search performance
* Default is 10000
* Do not change unless instructed to do so by Splunk Support
```

```
maxcount = <integer>
* The total number of metadata search results returned by the search
head;
  after the maxcount is reached, any additional metadata results
received from
  the search peers will be ignored (not returned)
* A larger number incurs additional memory usage on the search head
* Default is 100000
```

[set]

```
[set]
maxresultrows = <integer>
* The maximum number of results the set command will use from each
resultset
  to compute the required set operation
```

[input_channels]

```
[input_channels]
max_inactive = <integer>
* Internal setting, do not change unless instructed to do so by Splunk
Support

lowater_inactive = <integer>
* Internal setting, do not change unless instructed to do so by Splunk
Support

inactive_eligibility_age_seconds = <integer>
* Internal setting, do not change unless instructed to do so by Splunk
Support
```

[ldap]

```
[ldap]
max_users_to_precache = <unsigned integer>
* The maximum number of users we will attempt to pre-cache from LDAP
after reloading auth
* Set this to 0 to turn off pre-caching

allow_multiple_matching_users = <bool>
* This controls whether we allow login when we find multiple entries
with the
  same value for the username attribute
* When multiple entries are found, we choose the first user DN
  lexicographically
* Setting this to false is more secure as it does not allow any
```

```
ambiguous
  login, but users with duplicate entries will not be able to login.
* Defaults to true
```

[spath]

```
[spath]
extraction_cutoff = <integer>
* For extract-all spath extraction mode, only apply extraction to the
first
  <integer> number of bytes
* Defaults to 5000

extract_all = <boolean>
* Controls whether we respect automatic field extraction when spath is
invoked
  manually.
* If true, we extract all fields regardless of settings.  If false, we
only
  extract fields used by later search commands.
```

[reversedns]

```
[reversedns]
rdnsMaxDutyCycle = <integer>
* Generate diagnostic WARN in splunkd.log if reverse dns lookups are
taking
  more than this percent of time
* Range 0-100
* Defaults to 10
```

[viewstates]

```
[viewstates]
enable_reaper = <boolean>
* Controls whether the viewstate reaper runs
* Defaults to true

reaper_freq = <integer>
* Controls how often the viewstate reaper runs
* Defaults to 86400 (1 day)

reaper_soft_warn_level = <integer>
* Controls what the reaper considers an acceptable number of viewstates
* Defaults to 1000

ttl = <integer>
```

```
* Controls the age at which a viewstate is considered eligible for
reaping
* Defaults to 86400 (1 day)
```

[geostats]

```
[geostats]
maxzoomlevel = <integer>
* Controls the number of zoom levels that geostats will cluster events
on

zl_0_gridcell_latspan = <float>
* Controls what is the grid spacing in terms of latitude degrees at the
lowest zoom
  level, which is zoom-level 0.
* Grid-spacing at other zoom levels are auto created from this value by
reducing by a
  factor of 2 at each zoom-level.

zl_0_gridcell_longspan = <float>
* Controls what is the grid spacing in terms of longitude degrees at the
lowest zoom
  level, which is zoom-level 0
* Grid-spacing at other zoom levels are auto created from this value by
reducing by a
  factor of 2 at each zoom-level.

filterstrategy = <integer>
* Controls the selection strategy on the geoviz map. Allowed values are
1 and 2.
```

[iplocation]

```
[iplocation]
db_path = <path>
* Absolute path to GeoIP database in MMDB format
* If not set, defaults to database included with splunk
```

[tscollect]

```
[tscollect]
squashcase = <boolean>
* The default value of the 'squashcase' argument if not specified by the
command
* Defaults to false

keepresults = <boolean>
```

```
* The default value of the 'keepresults' argument if not specified by
the command
* Defaults to false

optimize_max_size_mb = <unsigned int>
* The maximum size in megabytes of files to create with optimize
* Specify 0 for no limit (may create very large tsidx files)
* Defaults to 1024
```

[tstats]

```
[tstats]
apply_search_filter = <boolean>
* Controls whether we apply role-based search filters when users run
tstats on
  normal index data
* Note: we never apply search filters to data collected with tscollect
or datamodel acceleration
* Defaults to true

summariesonly = <boolean>
* The default value of 'summariesonly' arg if not specified by the
command
* When running tstats on an accelerated datamodel, summariesonly=false
implies
  a mixed mode where we will fall back to search for missing TSIDX data
* summariesonly=true overrides this mixed mode to only generate results
from
  TSIDX data, which may be incomplete
* Defaults to false

allow_old_summaries = <boolean>
* The default value of 'allow_old_summaries' arg if not specified by
the
  command
* When running tstats on an accelerated datamodel,
allow_old_summaries=false
  ensures we check that the datamodel search in each bucket's summary
metadata
  is considered up to date with the current datamodel search. Only
summaries
  that are considered up to date will be used to deliver results.
* The allow_old_summaries=true attribute overrides this behavior and
will deliver results
  even from bucket summaries that are considered out of date with the
current
  datamodel.
* Defaults to false

chunk_size = <unsigned int>
```

```
* ADVANCED: The default value of 'chunk_size' arg if not specified by
the command
* This argument controls how many events are retrieved at a time within
a
  single TSIDX file when answering queries
* Consider lowering this value if tstats queries are using too much
memory
  (cannot be set lower than 10000)
* Larger values will tend to cause more memory to be used (per search)
and
  might have performance benefits.
* Smaller values will tend to reduce performance and might reduce memory
used
  (per search).
* Altering this value without careful measurement is not advised.
* Defaults to 10000000

warn_on_missing_summaries = <boolean>
* ADVANCED: Only meant for debugging summariesonly=true searches on
accelerated datamodels.
* When true, search will issue a warning for a tstats
summariesonly=true search for
  the following scenarios:
    a) If there is a non-hot bucket that has no corresponding datamodel
acceleration
       summary whatsoever.
    b) If the bucket's summary does not match with the current
datamodel acceleration search.
* Defaults to false
```

[pdf]

```
[pdf]
max_rows_per_table = <unsigned int>
* The maximum number of rows that will be rendered for a table within
  integrated PDF rendering
* Defaults to 1000

render_endpoint_timeout = <unsigned int>
* The number of seconds after which the pdfgen render endpoint will
timeout if
  it has not yet finished rendering the PDF output
* Defaults to 3600
```

[kvstore]

```
[kvstore]
max_accelerations_per_collection = <unsigned int>
```

```
* The maximum number of accelerations that can be assigned to a single
  collection
* Valid values range from 0 to 50
* Defaults to 10

max_fields_per_acceleration = <unsigned int>
* The maximum number of fields that can be part of a compound
acceleration
  (i.e. an acceleration with multiple keys)
* Valid values range from 0 to 50
* Defaults to 10

max_rows_per_query = <unsigned int>
* The maximum number of rows that will be returned for a single query to
a collection.
* If the query returns more rows than the specified value, then returned
result
  set will contain the number of rows specified in this value.
* Defaults to 50000

max_queries_per_batch = <unsigned int>
* The maximum number of queries that can be run in a single batch
* Defaults to 1000

max_size_per_result_mb = <unsigned int>
* The maximum size of the result that will be returned for a single
query to a
  collection in MB.
* Defaults to 50 MB

max_size_per_batch_save_mb = <unsigned int>
* The maximum size of a batch save query in MB
* Defaults to 50 MB

max_documents_per_batch_save = <unsigned int>
* The maximum number of documents that can be saved in a single batch
* Defaults to 1000

max_size_per_batch_result_mb = <unsigned int>
* The maximum size of the result set from a set of batched queries
* Defaults to 100 MB

max_rows_in_memory_per_dump = <unsigned int>
* The maximum number of rows in memory before flushing it to the CSV
projection
  of KVStore collection.
* Defaults to 200

max_threads_per_outputlookup = <unsigned int>
* The maximum number of threads to use during outputlookup commands on
KVStore
* If the value is 0 the thread count will be determined by CPU count
```

```
* Defaults to 1
```

[http_input]

```
[http_input]
max_number_of_tokens = <unsigned int>
* The maximum number of tokens reported by logging input metrics.
* Default to 10000.

metrics_report_interval = 60
* The interval (in seconds) of logging input metrics report.
* Default to 60 (one minute).

max_content_length = 1000000
* The maximum length of http request content accepted by HTTP Input
server.
* Default to 1000000 (~ 1MB).

max_number_of_ack_channel = 1000000
* The maximum number of ACK channels accepted by HTTP Event Collector
server.
* Default to 1000000 (~ 1M).

max_number_of_acked_requests_pending_query = 10000000
* The maximum number of ACKed requests pending query on HTTP Event
Collector server.
* Default to 10000000 (~ 10M).

max_number_of_acked_requests_pending_query_per_ack_channel = 1000000
* The maximum number of ACKed requested pending query per ACK channel on
HTTP
  Event Collector server..
* Default to 1000000 (~ 1M).
```

[slow_peer_disconnect]

```
[slow_peer_disconnect]
* Settings for the heuristic that will detect and disconnect slow peers
towards
  the end of a search that has returned a large volume of data

disabled = <boolean>
* is this feature enabled.
* Defaults to true

batch_search_activation_fraction = <double>
* The fraction of peers that must have completed before we start
disconnecting
```

```
* This is only applicable to batch search because the slow peers will
not hold
  back the fast peers.
* Defaults to 0.9

packets_per_data_point = <unsigned int>
* Rate statistics will be sampled once every packets_per_data_point
packets.
* Defaults to 500

sensitivity = <double>
* Sensitivity of the heuristic to newer values. For larger values of
sensitivity
  the heuristic will give more weight to newer statistic.
* Defaults to 0.3

grace_period_before_disconnect = <double>
* If the heuristic consistently claims that the peer is slow for at
least
  <grace_period_before_disconnect>*life_time_of_collector seconds then
only
  will we disconnect the peer
* Defaults to 0.1

threshold_data_volume = <unsigned int>
* The volume of uncompressed data that must have accumulated in KB from
  a peer before we consider them in the heuristic.
* Defaults to 1024

threshold_connection_life_time = <unsigned int>
* All peers will be given an initial grace period of at least these
many
  seconds before we consider them in the heuristic.
* Defaults to 60

bound_on_disconnect_threshold_as_fraction_of_mean = <double>
* The maximum value of the threshold data rate we will use to determine
if
  a peer is slow. The actual threshold will be computed dynamically at
search
  time but will never exceed
(100*maximum_threshold_as_fraction_of_mean)% on
  either side of the mean.
* Defaults to 0.2
```

[geomfilter]

```
[geomfilter]
enable_generalization = <boolean>
* Whether or not generalization is applied to polygon boundaries to
```

```
reduce
  point count for rendering
* Defaults to true

enable_clipping = <boolean>
* Whether or not polygons are clipped to the viewport provided by the
render client
* Defaults to true
```

[system_checks]

```
[system_checks]
insufficient_search_capabilities = enabled | disabled
* Enables/disables automatic daily logging of scheduled searches by
users who
  have insufficient capabilities to run them as configured.
* Such searches are those that:
  + Have schedule_priority set to a value other than "default" but the
owner
    does not have the edit_search_schedule_priority capability.
  + Have schedule_window set to a value other than "auto" but the owner
does
    not have the edit_search_schedule_window capability.
* This check and any resulting logging occur on system startup and
every 24
  hours thereafter.
* Defaults to enabled.

orphan_searches = enabled|disabled
* Enables/disables automatic UI message notifications to admins for
  scheduled saved searches with invalid owners.
  * Scheduled saved searches with invalid owners are considered
"orphaned".  They
    cannot be run because Splunk cannot determine the roles to use for
the search
    context.
  * Typically, this situation occurs when a user creates scheduled
searches
    then departs the organization or company, causing their account to
be
    deactivated.
* Currently this check and any resulting notifications occur on system
startup
  and every 24 hours thereafter.
* Defaults to enabled.

installed_files_integrity = enabled | log_only | disabled
* Enables/disables automatic verification on every startup that all the
files
  that were installed with the running Splunk version are still the
```

```
files that
  should be present.
  * Effectively this finds cases where files were removed or changed
that
    should not be removed or changed, whether by accident or intent.
  * The source of truth for the files that should be present is the
manifest
    file in the $SPLUNK_HOME directory that comes with the release, so
if this
    file is removed or altered, the check cannot work correctly.
  * Reading of all the files provided with the install has some I/O
cost,
    though it is paid out over many seconds and should not be severe.
* When "enabled", detected problems will cause a message to be posted to
the
  bulletin board (system UI status message).
* When "enabled" or "log_only", detected problems will cause details to
be
  written out to splunkd.log
* When "disabled", no check will be attempted or reported.
* Defaults to enabled.

###############################################################################
# Global Optimization Settings
###############################################################################
```

[search_optimization]

```
[search_optimization]
enabled = <bool>
* Enables search optimizations
* Defaults to true


###############################################################################
# Individual optimizers
###############################################################################

#Configuration options for predicate_push optimizations
```

[search_optimization::predicate_push]

```
[search_optimization::predicate_push]
enabled = <bool>
* Enables predicate push optimization
* Defaults to true
```

```
#Configuration options for predicate_merge optimizations
```

[search_optimization::predicate_merge]

```
[search_optimization::predicate_merge]
enabled = <bool>
* Enables predicate merge optimization
* Defaults to true
```

[mvexpand]

```
[mvexpand]
* This stanza allows for fine tuning of mvexpand search command.

max_mem_usage_mb = <non-negative integer>
* Overrides the default value for max_mem_usage_mb
* See definition in [default] max_mem_usage_mb for more details
* Defaults to 500 (MB)
```

[mvcombine]

```
[mvcombine]
* This stanza allows for fine tuning of mvcombine search command.

max_mem_usage_mb = <non-negative integer>
* overrides the default value for max_mem_usage_mb
* See definition in [default] max_mem_usage_mb for more details
* defaults to 500 (MB)
```

[xyseries]

```
[xyseries]
* This stanza allows for fine tuning of xyseries search command.

max_mem_usage_mb = <non-negative integer>
* overrides the default value for max_mem_usage_mb
* See definition in [default] max_mem_usage_mb for more details
```

limits.conf.example

```
#   Version 6.5.0
# CAUTION: Do not alter the settings in limits.conf unless you know what
```

```
you are doing.
# Improperly configured limits may result in splunkd crashes and/or
memory overuse.


[searchresults]
maxresultrows = 50000
# maximum number of times to try in the atomic write operation (1 = no
retries)
tocsv_maxretry = 5
# retry period is 1/2 second (500 milliseconds)
tocsv_retryperiod_ms = 500

[subsearch]
# maximum number of results to return from a subsearch
maxout = 100
# maximum number of seconds to run a subsearch before finalizing
maxtime = 10
# time to cache a given subsearch's results
ttl = 300

[anomalousvalue]
maxresultrows = 50000
# maximum number of distinct values for a field
maxvalues = 100000
# maximum size in bytes of any single value (truncated to this size if
larger)
maxvaluesize = 1000

[associate]
maxfields = 10000
maxvalues = 10000
maxvaluesize = 1000

# for the contingency, ctable, and counttable commands
[ctable]
maxvalues = 1000

[correlate]
maxfields = 1000

# for bin/bucket/discretize
[discretize]
maxbins = 50000
# if maxbins not specified or = 0, defaults to
searchresults::maxresultrows

[inputcsv]
# maximum number of retries for creating a tmp directory (with random
name in
# SPLUNK_HOME/var/run/splunk)
mkdir_max_retries = 100
```

```
[kmeans]
maxdatapoints = 100000000

[kv]
# when non-zero, the point at which kv should stop creating new columns
maxcols = 512

[rare]
maxresultrows = 50000
# maximum distinct value vectors to keep track of
maxvalues = 100000
maxvaluesize = 1000

[restapi]
# maximum result rows to be returned by /events or /results getters from
REST
# API
maxresultrows = 50000

[search]
# how long searches should be stored on disk once completed
ttl = 86400

# the approximate maximum number of timeline buckets to maintain
status_buckets = 300

# the last accessible event in a call that takes a base and bounds
max_count = 10000

# the minimum length of a prefix before a * to ask the index about
min_prefix_len = 1

# the length of time to persist search cache entries (in seconds)
cache_ttl = 300

[scheduler]

# User default value (needed only if different from system/default
value) when
# no max_searches_perc.<n>.when (if any) below matches.
max_searches_perc = 60

# Increase the value between midnight-5AM.
max_searches_perc.0 = 75
max_searches_perc.0.when = * 0-5 * * *

# More specifically, increase it even more on weekends.
max_searches_perc.1 = 85
max_searches_perc.1.when = * 0-5 * * 0,6

[slc]
```

```
# maximum number of clusters to create
maxclusters = 10000

[findkeywords]
#events to use in findkeywords command (and patterns UI)
maxevents = 50000

[stats]
maxresultrows = 50000
maxvalues = 10000
maxvaluesize = 1000

[top]
maxresultrows = 50000
# maximum distinct value vectors to keep track of
maxvalues = 100000
maxvaluesize = 1000


[search_optimization]
enabled = true

[search_optimization::predicate_push]
enabled = true

[search_optimization::predicate_merge]
enabled = true
```

literals.conf

The following are the spec and example files for literals.conf.

literals.conf.spec

```
#   Version 6.5.0
#
# This file contains attribute/value pairs for configuring externalized
strings
# in literals.conf.
#
# There is a literals.conf in $SPLUNK_HOME/etc/system/default/.  To set
custom
# configurations, place a literals.conf in
$SPLUNK_HOME/etc/system/local/. For
# examples, see literals.conf.example. You must restart Splunk to
enable
```

```
# configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
#
# For the full list of all literals that can be overridden, check out
# $SPLUNK_HOME/etc/system/default/literals.conf.

###############################################################################
#
# CAUTION:
#
#  - You can destroy Splunk's performance by editing literals.conf
incorrectly.
#
#  - Only edit the attribute values (on the right-hand side of the
'=').
#    DO NOT edit the attribute names (left-hand side of the '=').
#
#  - When strings contain "%s", do not add or remove any occurrences
of %s, or
#    reorder their positions.
#
#  - When strings contain HTML tags, take special care to make sure that
all
#    tags and quoted attributes are properly closed, and that all
entities such
#    as & are escaped.
#
```

literals.conf.example

```
#   Version 6.5.0
#
# This file contains an example literals.conf, which is used to
# configure the externalized strings in Splunk.
#
# For the full list of all literals that can be overwritten, consult
# the far longer list in $SPLUNK_HOME/etc/system/default/literals.conf
#

[ui]
```

```
PRO_SERVER_LOGIN_HEADER = Login to Splunk (guest/guest)
INSUFFICIENT_DISK_SPACE_ERROR = The server's free disk space is too
low.  Indexing will temporarily pause until more disk space becomes
available.
SERVER_RESTART_MESSAGE = This Splunk Server's configuration has been
changed.  The server needs to be restarted by an administrator.
UNABLE_TO_CONNECT_MESSAGE = Could not connect to splunkd at %s.
```

macros.conf

The following are the spec and example files for macros.conf.

macros.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attribute/value pairs for search language
macros.

# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

[<STANZA_NAME>]

```
[<STANZA_NAME>]
* Each stanza represents a search macro that can be referenced in any
search.
* The stanza name is the name of the macro if the macro takes no
arguments.
  Otherwise, the stanza name is the macro name appended with
"(<numargs>)",
  where <numargs> is the number of arguments that this macro takes.
* Macros can be overloaded. In other words, they can have the same name
but a
  different number of arguments. If you have [foobar], [foobar(1)],
  [foobar(2)], etc., they are not the same macro.
* Macros can be used in the search language by enclosing the macro name
and any
  argument list within tick marks, for example:`foobar(arg1,arg2)` or
`footer`.
* Splunk does not expand macros when they are inside of quoted values,
```

```
for
  example: "foo`bar`baz".

args = <string>,<string>,...
* A comma-delimited string of argument names.
* Argument names can only contain alphanumeric characters, underscores
'_', and
  hyphens '-'.
* If the stanza name indicates that this macro takes no arguments, this
  attribute will be ignored.
* This list cannot contain any repeated elements.

definition = <string>
* The string that the macro will expand to, with the argument
substitutions
  made. (The exception is when iseval = true, see below.)
* Arguments to be substituted must be wrapped by dollar signs ($), for
example:
  "the last part of this string will be replaced by the value of
argument foo $foo$".
* Splunk replaces the $<arg>$ pattern globally in the string, even
inside of
  quotes.

validation = <string>
* A validation string that is an 'eval' expression.  This expression
must
  evaluate to a boolean or a string.
* Use this to verify that the macro's argument values are acceptable.
* If the validation expression is boolean, validation succeeds when it
returns
  true.  If it returns false or is NULL, validation fails, and Splunk
returns
  the error message defined by the attribute, errormsg.
* If the validation expression is not boolean, Splunk expects it to
return a
  string or NULL.  If it returns NULL, validation is considered a
success.
  Otherwise, the string returned is the error string.

errormsg = <string>
* The error message to be displayed if validation is a boolean
expression and
  it does not evaluate to true.

iseval = <true/false>
* If true, the definition attribute is expected to be an eval expression
that
  returns a string that represents the expansion of this macro.
* Defaults to false.

description = <string>
```

```
* OPTIONAL. Simple english description of what the macro does.
```

macros.conf.example

```
#   Version 6.5.0
#
# Example macros.conf
#

# macro foobar that takes no arguments can be invoked via `foobar`
[foobar]
# the defintion of a macro can invoke another macro.  nesting can be
indefinite
# and cycles will be detected and result in an error
definition = `foobar(foo=defaultfoo)`


# macro foobar that takes one argument, invoked via `foobar(someval)`
[foobar(1)]
args = foo
# note this is definition will include the leading and trailing quotes,
i.e.
# something `foobar(someval)`
# would expand to
# something "foo = someval"
definition = "foo = $foo$"

# macro that takes two arguments
# note that macro arguments can be named so this particular macro could
be
# invoked equivalently as `foobar(1,2)` `foobar(foo=1,bar=2)` or
# `foobar(bar=2,foo=1)`
[foobar(2)]
args = foo, bar
definition = "foo = $foo$, bar = $bar$"

# macro that takes one argument that does validation
[foovalid(1)]
args = foo
definition = "foovalid = $foo$"
# the validation eval function takes any even number of arguments (>=2)
where
# the first argument is a boolean expression, the 2nd a string, the
third
# boolean, 4th a string, etc etc etc
validation = validate(foo>15,"foo must be greater than
15",foo<=100,"foo must be <= 100")

# macro showing simple boolean validation, where if foo > bar is not
```

```
true,
# errormsg is displayed
[foovalid(2)]
args = foo, bar
definition = "foo = $foo$ and bar = $bar$"
validation = foo > bar
errormsg = foo must be greater than bar

# example of an eval-based definition.  For example in this case
# `fooeval(10,20)` would get replaced by 10 + 20
[fooeval(2)]
args = foo, bar
definition = if (bar > 0, "$foo$ + $bar$", "$foo$ - $bar$")
iseval = true
```

multikv.conf

The following are the spec and example files for multikv.conf.

multikv.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attribute and value pairs for creating
multikv
# rules.  Multikv is the process of extracting events from table-like
events,
# such as the output of top, ps, ls, netstat, etc.
#
# There is NO DEFAULT multikv.conf.  To set custom configurations,
place a
# multikv.conf in $SPLUNK_HOME/etc/system/local/. For examples, see
# multikv.conf.example.  You must restart Splunk to enable
configurations.
#
# To learn more about configuration files (including precedence) see
the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
#
# NOTE: Only configure multikv.conf if Splunk's default multikv behavior
does
# not meet your needs.

# A table-like event includes a table consisting of four sections:
```

```
#
```

Section Name | Description

```
#-------------------------------------------------------------------------------
# Section Name | Description
#-------------------------------------------------------------------------------
Name | Description
# pre          | optional: info/description (for example: the system
summary output in top)
# header       | optional: if not defined, fields are named Column_N
# body         | required: the body of the table from which child events
are constructed
# post         | optional: info/description
#-------------------------------------------------------------------------------

# NOTE: Each section must have a definition and a processing component.
See
# below.

[<multikv_config_name>]
* Name of the stanza to use with the multikv search command, for
example:
       '| multikv conf=<multikv_config_name> rmorig=f | ....'
* Follow this stanza name with any number of the following
attribute/value pairs.
```

Section Definition

```
#####################
# Section Definition
#####################Section Definition
# Define where each section begins and ends.

<Section Name>.start = <regex>
* A line matching this regex denotes the start of this section
(inclusive).

OR

<Section Name>.start_offset = <int>
* Line offset from the start of an event or the end of the previous
section
  (inclusive).
* Use this if you cannot define a regex for the start of the section.

<Section Name>.member = <regex>
```

```
* A line membership test.
* Member if lines match the regex.

<Section Name>.end = <regex>
* A line matching this regex denotes the end of this section
(exclusive).

OR

<Section Name>.linecount = <int>
* Specify the number of lines in this section.
* Use this if you cannot specify a regex for the end of the section.
```

Section processing

```
#####################
# Section processing
#####################Section processing
# Set processing for each section.

<Section Name>.ignore = [_all_|_none_|_regex_ <regex-list>]
* Determines which member lines will be ignored and not processed
further.

<Section Name>.replace = <quoted-str> = <quoted-str>, <quoted-str> =
<quoted-str>,...
* List of the form: "toReplace" = "replaceWith".
* Can have any number of quoted string pairs.
* For example: "%" = "_", "#" = "_"

<Section Name>.tokens = [<chopper>|<tokenizer>|<aligner>|<token-list>]
* See below for definitions of each possible token: chopper, tokenizer,
aligner,
  token-list.

<chopper> = _chop_, <int-list>
* Transform each string into a list of tokens specified by <int-list>.
* <int-list> is a list of (offset, length) tuples.

<tokenizer> = _tokenize_ <max_tokens (int)> <delims>
(<consume-delims>)?
* Tokenize the string using the delim characters.
* This generates at most max_tokens number of tokens.
* Set max_tokens to:
  * -1 for complete tokenization.
  * 0 to inherit from previous section (usually header).
  * A non-zero number for a specific token count.
* If tokenization is limited by the max_tokens, the rest of the string
is added
```

```
  onto the last token.
* <delims> is a comma-separated list of delimiting chars.
* <consume-delims> - boolean, whether to consume consecutive delimiters.
Set to
                    false/0 if you want consecutive delimiters to be
treated
                    as empty values. Defaults to true.

<aligner> = _align_, <header_string>, <side>, <max_width>
* Generates tokens by extracting text aligned to the specified header
fields.
* header_string: a complete or partial header field value the columns
are aligned with.
* side: either L or R (for left or right align, respectively).
* max_width: the maximum width of the extracted field.
  * Set max_width to -1 for automatic width. This expands the field
until any
    of the following delimiters are found: " ", "\t"

<token_list> = _token_list_ <comma-separated list>
* Defines a list of static tokens in a section.
* This is useful for tables with no header, for example: the output of
'ls -lah'
  which misses a header altogether.
```

multikv.conf.example

```
#   Version 6.5.0
#
# This file contains example multi key/value extraction configurations.
#
# To use one or more of these configurations, copy the configuration
block into
# multikv.conf in $SPLUNK_HOME/etc/system/local/. You must restart
Splunk to
# enable configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles


# This example breaks up the output from top:

# Sample output:

# Processes: 56 total, 2 running, 54 sleeping... 221 threads 10:14:07
```

```
#.....
#
#   PID COMMAND  %CPU TIME     #TH #PRTS #MREGS RPRVT RSHRD RSIZE  VSIZE
# 29960 mdimport 0.0%  0:00.29  3    60    50  1.10M  2.55M 3.54M  38.7M
# 29905 pickup   0.0%  0:00.01  1    16    17   164K   832K  764K
 26.7M
#....

[top_mkv]
# pre table starts at "Process..." and ends at line containing "PID"
pre.start = "Process"
pre.end = "PID"
pre.ignore = _all_

# specify table header location and processing
header.start = "PID"
header.linecount = 1
header.replace = "%" = "_", "#" = "_"
header.tokens = _tokenize_, -1," "

# table body ends at the next "Process" line (ie start of another top)
tokenize
# and inherit the number of tokens from previous section (header)
body.end = "Process"
body.tokens  = _tokenize_,  0, " "



## This example handles the output of 'ls -lah' command:
#
# total 2150528
# drwxr-xr-x 88 john john 2K   Jan 30 07:56 .
# drwxr-xr-x 15 john john 510B Jan 30 07:49 ..
# -rw------- 1  john john 2K   Jan 28 11:25 .hiden_file
# drwxr-xr-x 20 john john 680B Jan 30 07:49 my_dir
# -r--r--r-- 1  john john 3K   Jan 11 09:00 my_file.txt


[ls-lah-cpp]
pre.start     = "total"
pre.linecount = 1

# the header is missing, so list the column names
header.tokens = _token_list_, mode, links, user, group, size, date, name

# The ends when we have a line starting with a space
body.end     = "^\s*$"
# This filters so that only lines that contain with .cpp are used
body.member  = "\.cpp"
# concatenates the date into a single unbreakable item
body.replace = "(\w{3})\s+(\d{1,2})\s+(\d{2}:\d{2})" ="\1_\2_\3"
```

```
# ignore dirs
body.ignore = _regex_ "^drwx.*",
body.tokens  = _tokenize_, 0, " "
```

outputs.conf

The following are the spec and example files for outputs.conf.

outputs.conf.spec

```
#   Version 6.5.0
#
# Forwarders require outputs.conf; non-forwarding Splunk instances do
not
# use it.  It determines how the forwarder sends data to receiving
Splunk
# instances, either indexers or other forwarders.
#
# To configure forwarding, create an outputs.conf file in
# $SPLUNK_HOME/etc/system/local/.  For examples of its use, see
# outputs.conf.example.
#
# You must restart Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
#
# NOTE: To learn more about forwarding, see the documentation at
#
http://docs.splunk.com/Documentation/Splunk/latest/Deploy/Aboutforwardingandreceivingda
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
top
#     of the file.
#   * Each conf file should have at most one default stanza. If there
are
```

```
#     multiple default stanzas, attributes are combined. In the case of
#     multiple definitions of the same attribute, the last definition in
the
#     file wins.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.

############
TCP Output stanzas
############
# There are three levels of TCP Output stanzas:
# * Global: [tcpout]
# * Target group: [tcpout:<target_group>]
# * Single server: [tcpout-server://<ip address>:<port>]
#
# Settings at more specific levels override settings at higher levels.
For
# example, an attribute set for a single server overrides the value of
that
# attribute, if any, set at that server's target group stanza. See the
# online documentation on configuring forwarders for details.
#
# This spec file first describes the three levels of stanzas (and any
# attributes unique to a particular level).  It then describes the
optional
# attributes, which can be set at any  of the three levels.


#----TCP Output Global Configuration -----
# The global configurations specified here in the [tcpout] stanza can
be
# overwritten in stanzas for specific target groups, as described later.
# Note that the defaultGroup and indexAndForward attributes can only be
set
# here, at the global level.
#
# Starting with 4.2, the [tcpout] stanza is no longer required.

[tcpout]

defaultGroup = <target_group>, <target_group>, ...
* Comma-separated list of one or more target group names, specified
later
  in [tcpout:<target_group>] stanzas.
* The forwarder sends all data to the specified groups.
* If you don't want to forward data automatically, don't set this
attribute.
* Can be overridden by an inputs.conf _TCP_ROUTING setting, which in
turn
  can be overridden by a props.conf/transforms.conf modifier.
* Starting with 4.2, this attribute is no longer required.
```

```
indexAndForward = [true|false]
* Index all data locally, in addition to forwarding it.
* This is known as an "index-and-forward" configuration.
* This attribute is only available for heavy forwarders.
* This attribute is available only at the top level [tcpout] stanza. It
  cannot be overridden in a target group.
* Defaults to false.

#----Target Group Configuration -----

# If multiple servers are specified in a target group, the forwarder
# performs auto load-balancing, sending data alternately to each
available
# server in the group. For example, assuming you have three servers
# (server1, server2, server3) and autoLBFrequency=30, the forwarder
sends
# all data to server1 for 30 seconds, then it sends all data to server2
for
# the next 30 seconds, then all data to server3 for the next 30 seconds,
# finally cycling back to server1.
#
# You can have as many target groups as you want.
# If more than one target group is specified, the forwarder sends all
data
# to each target group.
# This is known as "cloning" the data.


[tcpout:<target_group>]

server = [<ip>|<servername>]:<port>, [<ip>|<servername>]:<port>, ...
* Required if indexerDiscovery is not set.
* Takes a comma separated list of one or more systems to send data to
over a
  tcp socket.
* Typically used to specify receiving splunk systems, although it can be
  used to send data to non-splunk systems (see sendCookedData setting).
* For each mentioned system, the following are required:
  * IP or servername where one or system is listening.
  * Port on which syslog server is listening.

blockWarnThreshold = <integer>
* Optional
* Default value is 100
* Sets the output pipleline send failure count threshold after which a
  failure message will be displayed as banner on UI
* To disable any warnings to be sent to UI on blocked output queue
  condition, set this to a large value (2 million for example)

indexerDiscovery = <name>
* Instructs the forwarder to fetch the list of indexers from the master
```

```
node
  specified in the corresponding [indexer_discovery:<name>] stanza.

token = <string>
* Optional
* If an access token is configured for receiving Splunk system, that
token
  is populated here. Note that if receiver is configured with an access
token
  and that token is not specified here, then data sent to it will be
  rejected.
#----Single server configuration -----

# You can define specific configurations for individual indexers on a
# server-by-server basis.  However, each server must also be part of a
# target group.

[tcpout-server://<ip address>:<port>]
* Optional.  There is no requirement to have any tcpout-server stanzas.
```

TCPOUT ATTRIBUTES----

```
############
#----TCPOUT ATTRIBUTES----
############TCPOUT ATTRIBUTES----
# These attributes are optional and can appear in any of the three
stanza levels.

[tcpout<any of above>]

#----General Settings----

sendCookedData = [true|false]
* If true, events are cooked (have been processed by Splunk).
* If false, events are raw and untouched prior to sending.
* Set to false if you are sending to a third-party system.
* Defaults to true.

heartbeatFrequency = <integer>
* How often (in seconds) to send a heartbeat packet to the receiving
server.
* Heartbeats are only sent if sendCookedData=true.
* Defaults to 30 (seconds).

blockOnCloning = [true|false]
* If true, TcpOutputProcessor blocks till at least one of the cloned
group
  gets events. This will not drop events when all the cloned groups are
  down.
```

```
* If false, TcpOutputProcessor will drop events when all the cloned
groups
  are down and queues for the cloned groups are full. When at least one
of
  the cloned groups is up and queues are not full, the events are not
  dropped.
* Defaults to true.

# For the following setting see the [tcpout:<target_group>] stanza
blockWarnThreshold = <integer>

compressed = [true|false]
* Applies to non-SSL forwarding only. For SSL useClientSSLCompression
  setting is used.
* If true, forwarder sends compressed data.
* If set to true, the receiver port must also have compression turned
on (in
  its inputs.conf file).
* Defaults to false.

negotiateNewProtocol = [true|false]
* When setting up a connection to an indexer, try to negotiate the use
of
  the new forwarder protocol.
* If set to false, the forwarder will not query the indexer for support
for
  the new protocol, and the connection will fall back on the
traditional
  protocol.
* Defaults to true.

channelReapInterval = <integer>
* Controls how often, in milliseconds, channel codes are reaped, i.e.
made
  available for re-use.
* This value sets the minimum time between reapings; in practice,
  consecutive reapings may be separated by greater
  than <channelReapInterval> milliseconds.
* Defaults to 60000 (1 minute)

channelTTL = <integer>
* Controls how long, in milliseconds, a channel may remain "inactive"
before
  it is reaped, i.e. before its code is made available for re-use by a
  different channel.
* Defaults to 300000 (5 minutes)

channelReapLowater = <integer>
* If the number of active channels is above <channelReapLowater>, we
reap
  old channels in order to make their channel codes available for
re-use.
```

```
* If the number of active channels is below <channelReapLowater>, we do
not
  reap channels, no matter how old they are.
* This value essentially determines how many active-but-old channels we
keep
  "pinned" in memory on both sides of a splunk-to-splunk connection.
* A non-zero value helps ensure that we do not waste network resources
by
  "thrashing" channels in the case of a forwarder sending a trickle of
data.
* Defaults to 10.

socksServer = [<ip>|<servername>]:<port>
* IP or servername of Socks5 server.
* Port on which socks server is listening on. You must specify the
port.
* Note: Only Socks5 is supported.

socksUsername = <username>
* Optional
* Socks username to use when authenticating against socks server

socksPassword = <password>
* Optional
* Socks password to use when authenticating against socks server

socksResolveDNS = <bool>
* Optional
* If set to true, forwarder will not attempt to resolve indexer's DNS,
and
* will forward the indexer's DNS as is to let socks server resolve it.

#----Queue Settings----

maxQueueSize = [<integer>|<integer>[KB|MB|GB]|auto]
* This attribute sets the maximum size of the forwarder's output queue.
* The size can be limited based on the number of entries, or on the
total
  memory used by the items in the queue.
* If specified as a lone integer (for example, maxQueueSize=100),
  maxQueueSize indicates the maximum count of queued items.
* If specified as an integer followed by KB, MB, or GB
  (for example, maxQueueSize=100MB), maxQueueSize indicates the maximum
RAM
  size of all the items in the queue.
* If set to auto, chooses a value depending on whether useACK is
enabled.
  * If useACK=false, uses 500KB
  * If useACK=true, uses 7MB
* If the useACK setting is enabled, the maximum size of the wait queue
is
  set to to 3x this value.
```

```
  * Although the wait queue and the output queue sizes are both
controlled
    by this attribute, they are separate.
* Limiting the queue sizes by quantity is largely historical.  However,
  should you choose to configure queues based on quantity, keep the
  following in mind:
  * Queued items can be events or blocks of data.
    * Non-parsing forwarders, such as universal forwarders, will send
      blocks, which may be up to 64KB.
    * Parsing forwarders, such as heavy forwarders, will send events,
which
      will be the size of the events.  For some events these are as
small as
      a few hundred bytes.  In unusual cases (data dependent),
customers may
      arrange to produce events that are multiple megabytes.
* Defaults to auto
  * If useACK is enabled, effectively defaults the wait queue to 21MB

dropEventsOnQueueFull = <integer>
* If set to a positive number, wait <integer> seconds before throwing
out
  all new events until the output queue has space.
* Setting this to -1 or 0 will cause the output queue to block when it
gets
  full, causing further blocking up the processing chain.
* If any target group's queue is blocked, no more data will reach any
other
  target group.
* Using auto load-balancing is the best way to minimize this condition,
  because, in that case, multiple receivers must be down (or jammed up)
  before queue blocking can occur.
* Defaults to -1 (do not drop events).
* DO NOT SET THIS VALUE TO A POSITIVE INTEGER IF YOU ARE MONITORING
FILES!

dropClonedEventsOnQueueFull = <integer>
* If set to a positive number, do not block completely, but wait up to
  <integer> seconds to queue events to a group. If it cannot enqueue to
a
  group for more than <integer> seconds, begin dropping events for the
  group. It makes sure that at least one group in the cloning
configuration
  will get events. It blocks if event cannot be delivered to any of the
  cloned groups.
* If set to -1, the TcpOutputProcessor will make sure that each group
will
  get all of the events.  If one of the groups is down, then Splunk
will
  block everything.
* Defaults to 5.
```

```
#----Backoff Settings When Unable To Send Events to Indexer----
# The settings in this section determine forwarding behavior when there
are
# repeated failures in sending events to an indexer ("sending
failures").

maxFailuresPerInterval = <integer>
* Specifies the maximum number failures allowed per interval before
backoff
  takes place. The interval is defined below.
* Defaults to 2.

secsInFailureInterval = <integer>
* Number of seconds in an interval. If the number of write failures
exceeds
  maxFailuresPerInterval in the specified secsInFailureInterval
seconds, the
  forwarder applies backoff. The backoff time period range is
  1-10 * autoLBFrequency.
* Defaults to 1.

backoffOnFailure = <positive integer>
* Number of seconds a forwarder will wait before attempting another
  connection attempt.
* Defaults to 30

maxConnectionsPerIndexer = <integer>
* Maximum number of allowed connections per indexer. In presence of
  failures, the max number of connection attempt per indexer at any
point in
  time.
* Defaults to 2.

connectionTimeout = <integer>
* Time out period if connection establishment does not finish in
<integer>
  seconds.
* Defaults to 20 seconds.

readTimeout = <integer>
* Time out period if read from socket does not finish in <integer>
seconds.
* This timeout is used to read acknowledgment when indexer
acknowledgment is
  used (useACK=true).
* Defaults to 300 seconds.

writeTimeout = <integer>
* Time out period if write on socket does not finish in <integer>
seconds.
* Defaults to 300 seconds.
```

```
tcpSendBufSz = <integer>
* TCP send buffer size in <integer> bytes.
* Useful to improve thruput with small size events like windows events.
* Only set this value if you are a TCP/IP expert.
* Defaults to system default.

ackTimeoutOnShutdown = <integer>
* Time out period if ACKs not received in <integer> seconds during
forwarder shutdown.
* Defaults to 30 seconds.

dnsResolutionInterval = <integer>
* Specifies base time interval in seconds at which indexer dns names
will be
  resolved to ip address.  This is used to compute runtime
  dnsResolutionInterval as follows:
  runtime interval = dnsResolutionInterval + (number of indexers in
server settings - 1)*30.
  DNS resolution interval is extended by 30 second for each additional
  indexer in server setting.
* Defaults to 300 seconds.

forceTimebasedAutoLB = [true|false]
* Will force existing streams to switch to newly elected indexer every
  AutoLB cycle.
* Defaults to false

#----Index Filter Settings.
# These attributes are only applicable under the global [tcpout] stanza.
# This filter does not work if it is created under any other stanza.
forwardedindex.<n>.whitelist = <regex>
forwardedindex.<n>.blacklist = <regex>
* These filters determine which events get forwarded, based on the
indexes
  the events belong are targetting.
* This is an ordered list of whitelists and blacklists, which together
  decide if events should be forwarded to an index.
* The order is determined by <n>. <n> must start at 0 and continue with
  positive integers, in sequence. There cannot be any gaps in the
sequence.
  * For example:
    forwardedindex.0.whitelist, forwardedindex.1.blacklist,
forwardedindex.2.whitelist, ...
* The filters can start from either whitelist or blacklist. They are
tested
  from forwardedindex.0 to forwardedindex.<max>.
* If both forwardedindex.<n>.whitelist and forwardedindex.<n>.blacklist
are
  present for the same value of n, then forwardedindex.<n>.whitelist is
  honored. forwardedindex.<n>.blacklist is ignored in this case.
* You should not normally need to change these filters from their
default
```

```
  settings in $SPLUNK_HOME/system/default/outputs.conf.
* Filtered out events are not indexed if local indexing is not enabled.

forwardedindex.filter.disable = [true|false]
* If true, disables index filtering. Events for all indexes are then
  forwarded.
* Defaults to false.

#----Automatic Load-Balancing
autoLB = true
* Automatic load balancing is the only way to forward data. Round-robin
  method is not supported anymore.
* Defaults to true.

autoLBFrequency = <seconds>
* Every autoLBFrequency seconds, a new indexer is selected randomly from
the
  list of indexers provided in the server attribute of the target group
  stanza.
* Defaults to 30 (seconds).

#----SSL Settings----

# To set up SSL on the forwarder, set the following attribute/value
pairs.
# If you want to use SSL for authentication, add a stanza for each
receiver
# that must be certified.

sslPassword = <password>
* The password associated with the CAcert.
* The default Splunk CAcert uses the password "password".
* There is no default value.

clientCert = <path>
* The full path to the client SSL certificate in PEM format.
* If (and only if) specified, this connection will use SSL.
* There is no default value.

sslCertPath = <path>
* DEPRECATED; use 'clientCert' instead.

cipherSuite = <string>
* If set, uses the specified cipher string for the input processors.
* If not set, the default cipher string provided by OpenSSL is used.
* This is used to ensure that the server does not accept connections
using weak
  encryption protocols.

sslCipher = <string>
* DEPRECATED; use 'cipherSuite' instead.
```

```
ecdhCurves = <comma separated list of ec curves>
* ECDH curves to use for ECDH key negotiation.
* The curves should be specified in the order of preference.
* The client sends these curves as a part of Client Hello.
* The server supports only the curves specified in the list.
* We only support named curves specified by their SHORT names.
  (see struct ASN1_OBJECT in asn1.h)
* The list of valid named curves by their short/long names can be
obtained
  by executing this command:
  $SPLUNK_HOME/bin/splunk cmd openssl ecparam -list_curves
* Default is empty string.
* e.g. ecdhCurves = prime256v1,secp384r1,secp521r1

sslRootCAPath = <path>
* DEPRECATED; use 'server.conf/[sslConfig]/sslRootCAPath' instead.
* Used only if server.conf's 'sslRootCAPath' is unset.
* Full path to the root CA (Certificate Authority) certificate store.
* The <path> must refer to a PEM format file containing one or more
root CA
  certificates concatenated together.
* Default is unset.

sslVerifyServerCert = <bool>
* If true, you must make sure that the server you are connecting to is
a
  valid one (authenticated).
* Both the common name and the alternate name of the server are then
checked
  for a match.
* Defaults to false.

tlsHostname = <string>
* TLS extension that allows sending an identifier with SSL Client Hello
* Defaults to empty string

sslCommonNameToCheck = <commonName1>, <commonName2>, ...
* Optional. Defaults to no common name checking.
* Check the common name of the server's certificate against this name.
* If there is no match, assume that Splunk is not authenticated against
this
  server.
* 'sslVerifyServerCert' must be set to true for this setting to work.

sslAltNameToCheck = <alternateName1>, <alternateName2>, ...
* Optional. Defaults to no alternate name checking.
* Check the alternate name of the server's certificate against this list
of names.
* If there is no match, assume that Splunk is not authenticated against
this
  server.
* 'sslVerifyServerCert' must be set to true for this setting to work.
```

```
useClientSSLCompression = <bool>
* Enables compression on SSL.
* Defaults to value of
'server.conf/[sslConfig]/useClientSSLCompression'.

sslQuietShutdown = <bool>
* Enables quiet shutdown mode in SSL
* Defaults to false

sslVersions = <string>
* Comma-separated list of SSL versions to support
* The versions available are "ssl3", "tls1.0", "tls1.1", and "tls1.2"
* The special version "*" selects all supported versions.  The version
"tls"
  selects all versions tls1.0 or newer
* If a version is prefixed with "-" it is removed from the list
* SSLv2 is always disabled; "-ssl2" is accepted in the version list but
does nothing
* When configured in FIPS mode ssl3 is always disabled regardless of
  this configuration
* Defaults to "*,-ssl2".  (anything newer than SSLv2)

#----Indexer Acknowledgment ----
# Indexer acknowledgment ensures that forwarded data is reliably
delivered
# to the receiver.
# If the receiver is an indexer, it indicates that the indexer has
received
# the data, indexed it, and written it to the file system. If the
receiver
# is an intermediate forwarder, it indicates that the intermediate
forwarder
# has successfully forwarded the data to the terminating indexer and has
# received acknowledgment from  that indexer.

# Important: Indexer acknowledgment is a complex feature that requires
# careful planning. Before using it, read the online topic describing it
in
# the Distributed Deployment manual.

useACK = [true|false]
* When set to true, the forwarder will retain a copy of each sent
event,
  until the receiving system sends an acknowledgement.
  * The receiver will send an acknowledgement when it has fully handled
it
    (typically written it to disk in indexing)
  * In the event of receiver misbehavior (acknowledgement is not
received),
    the data will be re-sent to an alternate receiver.
  * Note: the maximum memory used for the outbound data queues will
```

```
increase
    significantly by default (500KB ->  28MB) when useACK is enabled.
This
    is intended for correctness and performance.
* When set to false, the forwarder will consider the data fully
processed
  when it finishes writing it to the network socket.
* This attribute can be set at the [tcpout] or [tcpout:<target_group>]
  stanza levels. You cannot set it for individual servers at the
  [tcpout-server: ...] stanza level.
* Defaults to false.
```

Syslog output----

```
############
#----Syslog output----
############Syslog output----
# The syslog output processor is not available for universal or light
# forwarders.

# The following configuration is used to send output using syslog:

[syslog]
defaultGroup = <target_group>, <target_group>, ...

# For the following settings see the [syslog:<target_group>] stanza
below
type = [tcp|udp]
priority = <priority_value> | NO_PRI
dropEventsOnQueueFull = <integer>
maxEventSize = <integer>

[syslog:<target_group>]

#----REQUIRED SETTINGS----
# Required settings for a syslog output group:

server = [<ip>|<servername>]:<port>
* IP or servername where syslog server is running.
* Port on which server is listening. You must specify the port. Syslog,
by
  default, uses 514.

#----OPTIONAL SETTINGS----

# Optional settings for syslog output:

type = [tcp|udp]
* Protocol used.
```

```
* Default is udp.

priority = <priority_value> | NO_PRI
* The priority_value should specified as "<integer>" (an integer
surrounded
  by angle brackets). For example, specify  a priority of 34 like this:
<34>
* The integer must be one to three digits in length.
* The value you enter will appear in the syslog header.
* Mimics the number passed via syslog interface call, documented via man
  syslog.
* The integer can be computed as (<facility> * 8) + <severity>. For
example,
  if <facility> is 4 (security/authorization messages) and <severity>
is 2
  (critical conditions), the priority will be 34 = (4 * 8) + 2. Set the
  attribute to: <34>
* The table of facility and severity (and their values) can be
referenced in
  RFC3164, eg http://www.ietf.org/rfc/rfc3164.txt section 4.1.1
* Defaults to <13>, or a facility of "user" or typically unspecified
  application, and severity of "Notice".
* If you do not wish to add priority, set 'NO_PRI' as priority value.
    * Example: priority = NO_PRI
* The table is reproduced briefly here, some of these are archaic.
  Facility:
     0 kernel messages
     1 user-level messages
     2 mail system
     3 system daemons
     4 security/authorization messages
     5 messages generated internally by syslogd
     6 line printer subsystem
     7 network news subsystem
     8 UUCP subsystem
     9 clock daemon
    10 security/authorization messages
    11 FTP daemon
    12 NTP subsystem
    13 log audit
    14 log alert
    15 clock daemon
    16 local use 0  (local0)
    17 local use 1  (local1)
    18 local use 2  (local2)
    19 local use 3  (local3)
    20 local use 4  (local4)
    21 local use 5  (local5)
    22 local use 6  (local6)
    23 local use 7  (local7)
  Severity:
    0  Emergency: system is unusable
```

```
    1  Alert: action must be taken immediately
    2  Critical: critical conditions
    3  Error: error conditions
    4  Warning: warning conditions
    5  Notice: normal but significant condition
    6  Informational: informational messages
    7  Debug: debug-level messages

syslogSourceType = <string>
* Specifies an additional rule for handling data, in addition to that
  provided by the 'syslog' source type.
* This string is used as a substring match against the sourcetype key.
 For
  example, if the string is set to 'syslog', then all source types
  containing the string 'syslog' will receive this special treatment.
* To match a source type explicitly, use the pattern
  "sourcetype::sourcetype_name".
    * Example: syslogSourceType = sourcetype::apache_common
* Data which is 'syslog' or matches this setting is assumed to already
be in
  syslog format.
* Data which does not match the rules has a header, optionally a
timestamp
  (if defined in 'timestampformat'), and a hostname added to the front
of
  the event. This is how Splunk causes arbitrary log data to match
syslog
  expectations.
* Defaults to unset.

timestampformat = <format>
* If specified, the formatted timestamps are added to the start of
events
  forwarded to syslog.
* As above, this logic is only applied when the data is not syslog, or
the
  syslogSourceType.
* If the data is not in syslog-compliant format and timestampformat is
  not specified, the output produced will not be RFC3164-compliant.
* The format is a strftime-style timestamp formatting string. This is
the
  same implementation used in the 'eval' search command, splunk
logging, and
  other places in splunkd.
  * For example: %b %e %H:%M:%S for RFC3164-compliant output
    * %b - Abbreviated month name (Jan, Feb, ...)
    * %e - Day of month
    * %H - Hour
    * %M - Minute
    * %s - Second
* For a more exhaustive list of the formatting specifiers, refer to the
  online documentation.
```

```
* Note that the string is not quoted.
* Defaults to unset, which means that no timestamp will be inserted into
the
  front of events.

dropEventsOnQueueFull = <integer>
* If set to a positive number, wait <integer> seconds before throwing
out
  all new events until the output queue has space.
* Setting this to -1 or 0 will cause the output queue to block when it
gets
  full, causing further blocking up the processing chain.
* If any target group's queue is blocked, no more data will reach any
other
  target group.
* Defaults to -1 (do not drop events).

maxEventSize = <integer>
* If specified, sets the maximum size of an event that splunk will
transmit.
* All events excedding this size will be truncated.
* Defaults to 1024 bytes.

#---- Routing Data to Syslog Server -----
# To route data to syslog server:
# 1) Decide which events to route to which servers.
# 2) Edit the props.conf, transforms.conf, and outputs.conf files on
the
#    forwarders.

# Edit $SPLUNK_HOME/etc/system/local/props.conf and set a
TRANSFORMS-routing
# attribute as shown here:
#
# [<spec>]
# TRANSFORMS-routing=<unique_stanza_name>

* <spec> can be:
  * <sourcetype>, the source type of an event
  * host::<host>, where <host> is the host for an event
  * source::<source>, where <source> is the source for an event

* Use the <unique_stanza_name> when creating your entry in
transforms.conf.

# Edit $SPLUNK_HOME/etc/system/local/transforms.conf and set rules to
match your props.conf stanza:
#
#  [<unique_stanza_name>]
#  REGEX=<your_regex>
#  DEST_KEY=_SYSLOG_ROUTING
#  FORMAT=<unique_group_name>
```

```
* <unique_stanza_name> must match the name you created in props.conf.
* Enter the regex rules in <your_regex> to determine which events get
  conditionally routed.
* DEST_KEY should be set to _SYSLOG_ROUTING to send events via SYSLOG.
* Set FORMAT to <unique_group_name>. This should match the syslog group
name
  you create in outputs.conf.
```

IndexAndForward Processor-----

```
############
#----IndexAndForward Processor-----
############IndexAndForward Processor-----
# The IndexAndForward processor determines the default behavior for
indexing
# data on full Splunk. It has the "index" property, which determines
whether
# indexing occurs.
#
# When Splunk is not configured as a forwarder, "index" is set to
"true".
# That is, the Splunk instance indexes data by default.
#
# When Splunk is configured as a forwarder, the processor turns "index"
to
# "false". That is, the Splunk instance does not index data by default.
#
# The IndexAndForward processor has no effect on the universal
forwarder,
# which can never index data.
#
# If the [tcpout] stanza configures the indexAndForward attribute, the
value
# of that attribute overrides the default value of "index". However, if
you
# set "index" in the [indexAndForward] stanza, described below, it
# supersedes any value set in [tcpout].

[indexAndForward]
index = [true|false]
* If set to true, data is indexed.
* If set to false, data is not indexed.
* Default depends on whether the Splunk instance is configured as a
  forwarder, modified by any value configured for the indexAndForward
  attribute in [tcpout].

selectiveIndexing = [true|false]
* When index is 'true', all events are indexed. Setting
```

```
selectiveIndexing to
  'true' allows you to index only specific events that has key
  '_INDEX_AND_FORWARD_ROUTING' set.
* '_INDEX_AND_FORWARD_ROUTING' can be set in inputs.conf as:
  [<input_stanza>]
  _INDEX_AND_FORWARD_ROUTING = local
* Defaults to false.

[indexer_discovery:<name>]

pass4SymmKey = <password>
* Security key shared between indexer_discovery and forwarders.
* If specified here, the same value must also be specified on the master
node identified by master_uri.

send_timeout = <seconds>
* Low-level timeout for sending messages to the master node.
* Fractional seconds are allowed.
* Default is 30.

rcv_timeout = <seconds>
* Low-level timeout for receiving messages from the master node.
* Fractional seconds are allowed.
* Default is 30.

cxn_timeout = <seconds>
* Low-level timeout for connecting to the master node.
* Fractional seconds are allowed.
* Default is 30.

master_uri = <uri>
* URI and management port of the cluster master used in indexer
discovery.
* Example: https://SplunkMaster01.example.com:8089
```

outputs.conf.example

```
#   Version 6.5.0
#
# This file contains an example outputs.conf.  Use this file to
configure
# forwarding in a distributed set up.
#
# To use one or more of these configurations, copy the configuration
block into
# outputs.conf in $SPLUNK_HOME/etc/system/local/. You must restart
Splunk to
# enable configurations.
#
```

```
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles


# Specify a target group for an IP:PORT which consists of a single
receiver.
# This is the simplest possible configuration; it sends data to the
host at
# 10.1.1.197 on port 9997.

[tcpout:group1]
server=10.1.1.197:9997


# Specify a target group for a hostname which consists of a single
receiver.

[tcpout:group2]
server=myhost.Splunk.com:9997


# Specify a target group made up of two receivers.  In this case, the
data will
# be distributed using AutoLB between these two receivers.  You can
specify as
# many receivers as you wish here. You can combine host name and IP if
you
# wish.
# NOTE: Do not use this configuration with SplunkLightForwarder.

[tcpout:group3]
server=myhost.Splunk.com:9997,10.1.1.197:6666


# You can override any of the global configuration values on a
per-target group
# basis.  All target groups that do not override a global config will
inherit
# the global config.

# Send every event to a receiver at foo.Splunk.com:9997 with a maximum
queue
# size of 100,500 events.

[tcpout:group4]
server=foo.Splunk.com:9997
heartbeatFrequency=45
maxQueueSize=100500
```

```
# Send data to a receiving system that controls access by tokens.
# NOTE: token value is encrypted. Encryption is done by REST endpoint
while saving.
[tcpout:group4]
server=foo.Splunk.com:9997
token=$1$/fRSBT+2APNAyCB7tlcgOyLnAtqAQFC8NI4TGA2wX4JHfN5d9g==

# Clone events to groups indexer1 and indexer2. Also, index all this
data
# locally as well.

[tcpout]
indexAndForward=true

[tcpout:indexer1]
server=Y.Y.Y.Y:9997

[tcpout:indexer2]
server=X.X.X.X:6666


# Clone events between two data balanced groups.

[tcpout:indexer1]
server=A.A.A.A:1111, B.B.B.B:2222

[tcpout:indexer2]
server=C.C.C.C:3333, D.D.D.D:4444

# Syslout output configuration
# This example sends only events generated by the splunk daemon to a
remote
# syslog host in syslog-compliant format:

[syslog:syslog-out1]
disabled = false
server = X.X.X.X:9099
type = tcp
priority = <34>
timestampformat = %b %e %H:%M:%S


# New in 4.0: Auto Load Balancing
#
# This example balances output between two indexers running on
# 1.2.3.4:4433 and 1.2.4.5:4433.
# To achieve this you'd create a DNS entry for splunkLB pointing
# to the two IP addresses of your indexers:
#
#   $ORIGIN example.com.
#   splunkLB A 1.2.3.4
#   splunkLB A 1.2.3.5
```

```
[tcpout]
defaultGroup = lb

[tcpout:lb]
server = splunkLB.example.com:4433
autoLB = true

# Alternatively, you can autoLB sans DNS:

[tcpout]
defaultGroup = lb

[tcpout:lb]
server = 1.2.3.4:4433, 1.2.3.5:4433
autoLB = true


# Compression
#
# This example sends compressed events to the remote indexer.
# NOTE: Compression can be enabled TCP or SSL outputs only.
# The receiver input port should also have compression enabled.

[tcpout]
server = splunkServer.example.com:4433
compressed = true


# SSL
#
# This example sends events to an indexer via SSL using splunk's
# self signed cert:

[tcpout]
server = splunkServer.example.com:4433
sslPassword = password
sslCertPath = $SPLUNK_HOME/etc/auth/server.pem
sslRootCAPath = $SPLUNK_HOME/etc/auth/ca.pem

#
# The following example shows how to route events to syslog server
# This is similar to tcpout routing, but DEST_KEY is set to
_SYSLOG_ROUTING
#

# 1. Edit $SPLUNK_HOME/etc/system/local/props.conf and set a
TRANSFORMS-routing
#    attribute:
[default]
TRANSFORMS-routing=errorRouting
```

```
[syslog]
TRANSFORMS-routing=syslogRouting

# 2. Edit $SPLUNK_HOME/etc/system/local/transforms.conf and set
errorRouting
#    and syslogRouting rules:
[errorRouting]
REGEX=error
DEST_KEY=_SYSLOG_ROUTING
FORMAT=errorGroup

[syslogRouting]
REGEX=.
DEST_KEY=_SYSLOG_ROUTING
FORMAT=syslogGroup

# 3. Edit $SPLUNK_HOME/etc/system/local/outputs.conf and set which
syslog
#    outputs go to with servers or groups:
[syslog]
defaultGroup=everythingElseGroup

[syslog:syslogGroup]
server = 10.1.1.197:9997

[syslog:errorGroup]
server=10.1.1.200:9999

[syslog:everythingElseGroup]
server=10.1.1.250:6666

#
# Perform selective indexing and forwarding
#
# With a heavy forwarder only, you can index and store data locally, as
well as
# forward the data onwards to a receiving indexer. There are two ways to
do
# this:

# 1. In outputs.conf:
[tcpout]
defaultGroup = indexers

[indexAndForward]
index=true
selectiveIndexing=true

[tcpout:indexers]
server = 10.1.1.197:9997, 10.1.1.200:9997

# 2. In inputs.conf, Add _INDEX_AND_FORWARD_ROUTING for any data that
```

```
you want
#    index locally, and
_TCP_ROUTING=<target_group> for data to be forwarded.

[monitor:///var/log/messages/]
_INDEX_AND_FORWARD_ROUTING=local

[monitor:///var/log/httpd/]
_TCP_ROUTING=indexers
```

passwords.conf

The following are the spec and example files for passwords.conf.

passwords.conf.spec

```
#   Version 6.5.0
#
# This file maintains the credential information for a given app in
Splunk Enterprise.
#
# There is no global, default passwords.conf. Instead, anytime a user
creates
# a new user or edit a user onwards hitting the storage endpoint
# will create this passwords.conf file which gets replicated
# in a search head clustering enviornment.
# Note that passwords.conf is only created from 6.3.0 release.
#
# You must restart Splunk Enterprise to reload manual changes to
passwords.conf.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
# More details for storage endpoint is at
# http://blogs.splunk.com/2011/03/15/storing-encrypted-credentials/
```

[credential:<realm>:<username>:]

```
[credential:<realm>:<username>:]
password = <password>
* Password that corresponds to the given username for the given realm.
```

```
  Note that realm is optional
* The password can be in clear text, however when saved from splunkd the
  password will always be encrypted
```

passwords.conf.example

```
#   Version 6.5.0
#
# The following are example passwords.conf configurations. Configure
properties for
# your custom application.
#
# There is NO DEFAULT passwords.conf. The file only gets created once
you add/edit
# a credential information via the storage endpoint as follows.
#
# The POST request to add user1 credentials to the storage/password
endpoint
# curl -k -u admin:changeme
https://localhost:8089/servicesNS/nobody/search/storage/passwords -d
name=user1 -d password=changeme2
#
# The GET request to list all the credentials stored at the
storage/passwords endpoint
# curl -k -u admin:changeme
https://localhost:8089/services/storage/passwords
#
# To use one or more of these configurations, copy the configuration
block into
# passwords.conf in $SPLUNK_HOME/etc/<apps>/local/. You must restart
Splunk to
# enable configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
#

[credential::testuser:]
password = changeme
```

pdf_server.conf

The following are the spec and example files for pdf_server.conf.

pdf_server.conf.spec

```
#   Version 6.1
#
# This file contains possible attributes and values you can use to
configure Splunk's pdf server.
#
# There is a pdf_server.conf in $SPLUNK_HOME/etc/system/default/.  To
set custom configurations,
# place a pdf_server.conf in $SPLUNK_HOME/etc/system/local/.  For
examples, see pdf_server.conf.example.
# You must restart the pdf server to enable configurations.
#
# To learn more about configuration files (including precedence) please
see the documentation
# located at
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#     * You can also define global settings outside of any stanza, at
the top of the file.
#     * Each conf file should have at most one default stanza. If there
are multiple default
#       stanzas, attributes are combined. In the case of multiple
definitions of the same
#       attribute, the last definition in the file wins.
#     * If an attribute is defined at both the global level and in a
specific stanza, the
#       value in the specific stanza takes precedence.

[settings]
        * Set general Splunk Web configuration options under this stanza
name.
        * Follow this stanza name with any number of the following
attribute/value pairs.
        * If you do not specify an entry for each attribute, Splunk will
use the default value.

startwebserver = [0|1]
           * Set whether or not to start the server.
           * 0 disables Splunk Web, 1 enables it.
           * Defaults to 1.

httpport = <port_number>
```

```
           * Must be present for the server to start.
           * If omitted or 0 the server will NOT start an http listener.
        * If using SSL, set to the HTTPS port number.
           * Defaults to 9000.

enableSplunkWebSSL = [True|False]
        * Toggle between http or https.
        * Set to true to enable https and SSL.
        * Defaults to False.

privKeyPath = /certs/privkey.pem
caCertPath = /certs/cert.pem
   * Specify paths and names for Web SSL certs.
   * Path is relative to $SPLUNK_HOME/share/splunk.

supportSSLV3Only = [True|False]
   * Allow only SSLv3 connections if true.
   * NOTE: Enabling this may cause some browsers problems.

root_endpoint = <URI_prefix_string>
   * Defines the root URI path on which the appserver will listen.
   * Default setting is '/'.
   * For example: if you want to proxy the splunk UI at
http://splunk:8000/splunkui, then set root_endpoint = /splunkui

static_endpoint = <URI_prefix_string>
   * Path to static content.
   * The path here is automatically appended to root_endpoint defined
above.
   * Default is /static.

static_dir = <relative_filesystem_path>
   * The directory that actually holds the static content.
   * This can be an absolute URL if you want to put it elsewhere.
   * Default is share/splunk/search_mrsparkle/exposed.

enable_gzip = [True|False]
   * Determines if web server applies gzip compression to responses.
   * Defaults to True.



#
# cherrypy HTTP server config
#

server.thread_pool = <integer>
   * Specifies the numbers of threads the app server is allowed to
maintain.
   * Defaults to 10.

server.socket_host = <ip_address>
```

```
   * Host values may be any IPv4 or IPv6 address, or any valid hostname.
   * The string 'localhost' is a synonym for '127.0.0.1' (or '::1', if
your hosts file prefers IPv6).
     The string '0.0.0.0' is a special IPv4 entry meaning "any active
interface" (INADDR_ANY), and
     '::' is the similar IN6ADDR_ANY for IPv6.
   * The empty string or None are not allowed.
   * Defaults to 0.0.0.0

log.access_file = <filename>
   * Specifies the HTTP access log filename.
   * Stored in default Splunk /var/log directory.
   * Defaults to pdf_access.log

log.error_file = <filename>
   * Specifies the HTTP error log filename.
   * Stored in default Splunk /var/log directory.
   * Defaults to pdf_service.log

log.screen = [True|False]
   * Indicates if runtime output is displayed inside an interactive tty.
   * Defaults to True

request.show_tracebacks = [True|False]
   * Indicates if an exception traceback is displayed to the user on
fatal exceptions.
   * Defaults to True

engine.autoreload_on = [True|False]
   * Indicates if the app server will auto-restart if it detects a
python file has changed.
   * Defaults to False

tools.sessions.on = True
    * Indicates if user session support is enabled.
    * Should always be True

tools.sessions.timeout = <integer>
   * Specifies the number of minutes of inactivity before a user session
expires.
   * Defaults to 60

response.timeout = <integer>
   * Specifies the number of seconds to wait for the server to complete
a response.
   * Some requests such as uploading large files can take a long time.
   * Defaults to 7200

tools.sessions.storage_type = [file]
tools.sessions.storage_path = <filepath>
   * Specifies the session information storage mechanisms.
   * Comment out these two lines to use RAM based sessions instead.
```

```
   * Use an absolute path to store sessions outside of the Splunk
directory tree.
   * Defaults to storage_type=file, storage_path=var/run/splunk

tools.decode.on = [True|False]
   * Indicates if all strings that come into CherryPy controller methods
are decoded as unicode (assumes UTF-8 encoding).
   * WARNING: Disabling this will likely break the application, as all
incoming strings are assumed
     to be unicode.
   * Defaults to True

tools.encode.on = [True|False]
   * Encodes all controller method response strings into UTF-8 str
objects in Python.
   * WARNING: Disabling this will likely cause high byte character
encoding to fail.
   * Defaults to True

tools.encode.encoding = <codec>
   * Force all outgoing characters to be encoded into UTF-8.
   * This only works with tools.encode.on set to True.
   * By setting this to utf-8, CherryPy's default behavior of observing
the Accept-Charset header
     is overwritten and forces utf-8 output. Only change this if you
know a particular browser
     installation must receive some other character encoding (Latin-1,
iso-8859-1, etc.).
   * WARNING: Change this at your own risk.
   * Defaults to utf-8

pid_path = <filepath>
   * Specifies the path to the PID file.
   * Defaults to var/run/splunk/splunkweb.pid.

firefox_cmdline = <cmdline>
   * Specifies additional arguments to pass to Firefox.
   * This should normally not be set.

max_queue = <integer>
   * Specifies the maximum size of the backlog of pending report
requests.
   * Once the backlog is reached the server will return an error on
receiving additional requests.
   * Defaults to 10.

max_concurrent = <integer>
   * Specifies the maximum number of copies of Firefox that the report
server will use concurrently to render reports.
   * Increase only if the host machine has multiple cores and plenty of
spare memory.
   * Defaults to 2.
```

```
Xvfb = <path>
  * Pathname to the Xvfb program.
  * Defaults to searching the PATH.

xauth = <path>
  * Pathname to the xauth program.
  * Defaults to searching the PATH.

mcookie = <path>
  * Pathname to the mcookie program.
  * Defaults to searching the PATH.

appserver_ipaddr = <ip_networks>
  * If set, the PDF server will only query Splunk app servers on IP
addresses within the IP networks
    specified here.
  * Networks can be specified as a prefix (10.1.0.0/16) or using a
netmask (10.1.0.0/255.255.0.0).
  * IPv6 addresses are also supported.
  * Individual IP addresses can also be listed (1.2.3.4).
  * Multiple networks should be comma separated.
  * Defaults to accepting any IP address.

client_ipaddr = <ip_networks>
  * If set, the PDF server will only accept requests from hosts whose
IP address falls within the IP
    networks specified here.
  * Generally this setting should match the appserver_ipaddr setting.
  * Format matches appserver_ipaddr.
  * Defaults to accepting any IP address.

screenshot_enabled = [True|False]
  * If enabled allows screenshots of the X server to be taken for
debugging purposes.
  * Enabling this is a potential security hole as anyone on an IP
address matching client_ipaddr will be
    able to see reports in progress.
  * Defaults to False.
```

pdf_server.conf.example

```
#   Version 6.1
#
# This is an example pdf_server.conf.  Use this file to configure pdf
server process settings.
#
# To use one or more of these configurations, copy the configuration
```

```
block into pdf_server.conf
# in $SPLUNK_HOME/etc/system/local/. You must restart the pdf server to
enable configurations.
#
# To learn more about configuration files (including precedence) please
see the documentation
# located at
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles



# This stanza heading must precede any changes.
[settings]

# Change the default port number:
httpport = 12900

# Lock down access to the IP address of specific appservers
# that will utilize the pdf server
appserver_ipaddr = 192.168.3.0/24,192.168.2.2
client_ipaddr = 192.168.3.0/24,192.168.2.2
```

procmon-filters.conf

The following are the spec and example files for procmon-filters.conf.

procmon-filters.conf.spec

```
#   Version 6.5.0
#
# *** DEPRECATED ***
#
#
# This file contains potential attribute/value pairs to use when
configuring
# Windows registry monitoring. The procmon-filters.conf file contains
the
# regular expressions you create to refine and filter the processes you
want
# Splunk to monitor. You must restart Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
```

```
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

#### find out if this file is still being used.
```

[<stanza name>]

```
[<stanza name>]
* Name of the filter being defined.

proc = <string>
* Regex specifying process image that you want Splunk to monitor.

type = <string>
* Regex specifying the type(s) of process event that you want Splunk to
  monitor.

hive = <string>
* Not used in this context, but should always have value ".*"
```

procmon-filters.conf.example

```
#   Version 6.5.0
#
# This file contains example registry monitor filters. To create your
own
# filter, use the information in procmon-filters.conf.spec.
#
# To use one or more of these configurations, copy the configuration
block into
# procmon-filters.conf in $SPLUNK_HOME/etc/system/local/. You must
restart
# Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

[default]
hive = .*

[not-splunk-optimize]
proc = (?<!splunk-optimize.exe)$
type = create|exit|image
```

props.conf

The following are the spec and example files for props.conf.

props.conf.spec

```
#
# This file contains possible attribute/value pairs for configuring
Splunk's
# processing properties via props.conf.
#
# Props.conf is commonly used for:
#
# * Configuring linebreaking for multiline events.
# * Setting up character set encoding.
# * Allowing processing of binary files.
# * Configuring timestamp recognition.
# * Configuring event segmentation.
# * Overriding Splunk's automated host and source type matching. You can
use
#   props.conf to:
#     * Configure advanced (regex-based) host and source type overrides.
#     * Override source type matching for data from a particular source.
#     * Set up rule-based source type recognition.
#     * Rename source types.
# * Anonymizing certain types of sensitive incoming data, such as credit
#   card or social security numbers, using sed scripts.
# * Routing specific events to a particular index, when you have
multiple
#   indexes.
# * Creating new index-time field extractions, including header-based
field
#   extractions.
#   NOTE: We do not recommend adding to the set of fields that are
extracted
#         at index time unless it is absolutely necessary because there
are
#         negative performance implications.
# * Defining new search-time field extractions. You can define basic
#   search-time field extractions entirely through props.conf. But a
#   transforms.conf component is required if you need to create
search-time
#   field extractions that involve one or more of the following:
#       * Reuse of the same field-extracting regular expression across
#         multiple sources, source types, or hosts.
#       * Application of more than one regex to the same source, source
type,
#         or host.
#       * Delimiter-based field extractions (they involve field-value
```

```
pairs
#         that are separated by commas, colons, semicolons, bars, or
#         something similar).
#       * Extraction of multiple values for the same field (multivalued
#         field extraction).
#       * Extraction of fields with names that begin with numbers or
#         underscores.
# * Setting up lookup tables that look up fields from external sources.
# * Creating field aliases.
#
# NOTE: Several of the above actions involve a corresponding
transforms.conf
# configuration.
#
# You can find more information on these topics by searching the Splunk
# documentation (http://docs.splunk.com/Documentation/Splunk).
#
# There is a props.conf in $SPLUNK_HOME/etc/system/default/.  To set
custom
# configurations, place a props.conf in $SPLUNK_HOME/etc/system/local/.
For
# help, see props.conf.example.
#
# You can enable configurations changes made to props.conf by typing
the
# following search string in Splunk Web:
#
# | extract reload=T
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
#
# For more information about using props.conf in conjunction with
# distributed Splunk deployments, see the Distributed Deployment Manual.
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
top
#     of the file.
#   * Each conf file should have at most one default stanza. If there
are
#     multiple default stanzas, attributes are combined. In the case of
#     multiple definitions of the same attribute, the last definition in
```

```
the
#     file wins.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.

[<spec>]
* This stanza enables properties for a given <spec>.
* A props.conf file can contain multiple stanzas for any number of
  different <spec>.
* Follow this stanza name with any number of the following
attribute/value
  pairs, as appropriate for what you want to do.
* If you do not set an attribute for a given <spec>, the default is
used.

<spec> can be:
1. <sourcetype>, the source type of an event.
2. host::<host>, where <host> is the host, or host-matching pattern,
for an
                  event.
3. source::<source>, where <source> is the source, or source-matching
                       pattern, for an event.
4. rule::<rulename>, where <rulename> is a unique name of a source type
                       classification rule.
5. delayedrule::<rulename>, where <rulename> is a unique name of a
delayed
                              source type classification rule.
                              These are only considered as a last resort
                              before generating a new source type based
on the
                              source seen.

**[<spec>] stanza precedence:**

For settings that are specified in multiple categories of matching
[<spec>]
stanzas, [host::<host>] settings override [<sourcetype>] settings.
Additionally, [source::<source>] settings override both [host::<host>]
and [<sourcetype>] settings.

**Considerations for Windows file paths:**

When you specify Windows-based file paths as part of a
[source::<source>]
stanza, you must escape any backslashes contained within the specified
file
path.

Example: [source::c:\\path_to\\file.txt]

**[<spec>] stanza patterns:**
```

When setting a [<spec>] stanza, you can use the following regex-type
syntax:
... recurses through directories until the match is met
    or equivalently, matches any number of characters.
*   matches anything but the path separator 0 or more times.
    The path separator is '/' on unix, or '\' on windows.
    Intended to match a partial or complete directory or filename.
| is equivalent to 'or'
( ) are used to limit scope of |.
\\ = matches a literal backslash '\'.

Example: [source::....(?<!tar.)(gz|bz2)]

 This matches any file ending with '.gz' or '.bz2', provided this is not
 preceded by 'tar.', so tar.bz2 and tar.gz would not be matched.

[source::<source>] and [host::<host>] stanza match language:

Match expressions must match the entire name, not just a substring. If
you
are familiar with regular expressions, match expressions are based on a
full
implementation of PCRE with the translation of ..., * and . Thus .
matches a
period, * matches non-directory separators, and ... matches any number
of
any characters.

For more information see the wildcards section at:
http://docs.splunk.com/Documentation/Splunk/latest/Data/Specifyinputpathswithwildcards

[<spec>] stanza pattern collisions:

Suppose the source of a given input matches multiple [source::<source>]
patterns. If the [<spec>] stanzas for these patterns each supply
distinct
settings, Splunk applies all of these settings.

However, suppose two [<spec>] stanzas supply the same setting. In this
case,
Splunk chooses the value to apply based on the ASCII order of the
patterns
in question.

For example, take this source:

    source::az

and the following colliding patterns:

    [source::...a...]

```
    sourcetype = a

    [source::...z...]
    sourcetype = z
```

In this case, the settings provided by the pattern [source::...a...] take
precedence over those provided by [source::...z...], and sourcetype ends up
with "a" as its value.

To override this default ASCII ordering, use the priority key:

```
    [source::...a...]
    sourcetype = a
    priority = 5

    [source::...z...]
    sourcetype = z
    priority = 10
```

Assigning a higher priority to the second stanza causes sourcetype to have
the value "z".

Case-sensitivity for [<spec>] stanza matching:

By default, [source::<source>] and [<sourcetype>] stanzas match in a
case-sensitive manner, while [host::<host>] stanzas match in a
case-insensitive manner. This is a convenient default, given that DNS names
are case-insensitive.

To force a [host::<host>] stanza to match in a case-sensitive manner use the
"(?-i)" option in its pattern.

For example:

```
    [host::foo]
    FIELDALIAS-a = a AS one

    [host::(?-i)bar]
    FIELDALIAS-b = b AS two
```

The first stanza will actually apply to events with host values of "FOO" or
"Foo" . The second stanza, on the other hand, will not apply to events with
host values of "BAR" or "Bar".

Building the final [<spec>] stanza:

```
The final [<spec>] stanza is built by layering together (1)
literal-matching
stanzas (stanzas which match the string literally) and (2) any
regex-matching stanzas, according to the value of the priority field.

If not specified, the default value of the priority key is:
* 0 for pattern-matching stanzas.
* 100 for literal-matching stanzas.

NOTE: Setting the priority key to a value greater than 100 causes the
pattern-matched [<spec>] stanzas to override the values of the
literal-matching [<spec>] stanzas.

The priority key can also be used to resolve collisions
between [<sourcetype>] patterns and [host::<host>] patterns. However,
be aware
that the priority key does *not* affect precedence across <spec> types.
For
example, [<spec>] stanzas with [source::<source>] patterns take
priority over
stanzas with [host::<host>] and [<sourcetype>] patterns, regardless of
their
respective priority key values.


#*******************************************************************************
# The possible attributes/value pairs for props.conf, and their
# default values, are:
#*******************************************************************************

# International characters and character encoding.

CHARSET = <string>
* When set, Splunk assumes the input from the given [<spec>] is in the
  specified encoding.
* Can only be used as the basis of [<sourcetype>] or [source::<spec>],
  not [host::<spec>].
* A list of valid encodings can be retrieved using the command "iconv
-l" on
  most *nix systems.
* If an invalid encoding is specified, a warning is logged during
initial
  configuration and further input from that [<spec>] is discarded.
* If the source encoding is valid, but some characters from the
[<spec>] are
  not valid in the specified encoding, then the characters are escaped
as
  hex (for example, "\xF3").
* When set to "AUTO", Splunk attempts to automatically determine the
character encoding and
  convert text from that encoding to UTF-8.
```

```
* For a complete list of the character sets Splunk automatically
detects,
  see the online documentation.
* This setting applies at input time, when data is first read by Splunk.
  The setting is used on a Splunk system that has configured inputs
  acquiring the data.
* Defaults to ASCII.
```

Line breaking

```
#*******************************************************************************
# Line breaking
#*******************************************************************************Line
breaking

# Use the following attributes to define the length of a line.

TRUNCATE = <non-negative integer>
* Change the default maximum line length (in bytes).
* Although this is in bytes, line length is rounded down when this would
  otherwise land mid-character for multi-byte characters.
* Set to 0 if you never want truncation (very long lines are, however,
often
  a sign of garbage data).
* Defaults to 10000 bytes.

LINE_BREAKER = <regular expression>
* Specifies a regex that determines how the raw text stream is broken
into
  initial events, before line merging takes place. (See the
SHOULD_LINEMERGE
  attribute, below)
* Defaults to ([\r\n]+), meaning data is broken into an event for each
line,
  delimited by any number of carriage return or newline characters.
* The regex must contain a capturing group -- a pair of parentheses
which
  defines an identified subcomponent of the match.
* Wherever the regex matches, Splunk considers the start of the first
  capturing group to be the end of the previous event, and considers the
end
  of the first capturing group to be the start of the next event.
* The contents of the first capturing group are discarded, and will not
be
  present in any event.  You are telling Splunk that this text comes
between
  lines.
* NOTE: You get a significant boost to processing speed when you use
```

```
  LINE_BREAKER to delimit multiline events (as opposed to using
  SHOULD_LINEMERGE to reassemble individual lines into multiline
events).
  * When using LINE_BREAKER to delimit events, SHOULD_LINEMERGE should
be set
    to false, to ensure no further combination of delimited events
occurs.
  * Using LINE_BREAKER to delimit events is discussed in more detail in
the web
    documentation at the following url:
    http://docs.splunk.com/Documentation/Splunk/latest/Data/indexmulti-lineevents

** Special considerations for LINE_BREAKER with branched expressions  **

When using LINE_BREAKER with completely independent patterns separated
by
pipes, some special issues come into play.
    EG. LINE_BREAKER = pattern1|pattern2|pattern3

Note, this is not about all forms of alternation, eg there is nothing
particular special about
    example: LINE_BREAKER = ([\r\n])+(one|two|three)
where the top level remains a single expression.

A caution: Relying on these rules is NOT encouraged.  Simpler is better,
in
both regular expressions and the complexity of the behavior they rely
on.
If possible, it is strongly recommended that you reconstruct your regex
to
have a leftmost capturing group that always matches.

It may be useful to use non-capturing groups if you need to express a
group
before the text to discard.
    EG. LINE_BREAKER = (?:one|two)([\r\n]+)
    * This will match the text one, or two, followed by any amount of
      newlines or carriage returns.  The one-or-two group is
non-capturing
      via the ?: prefix and will be skipped by LINE_BREAKER.

* A branched expression can match without the first capturing group
  matching, so the line breaker behavior becomes more complex.
  Rules:
  1: If the first capturing group is part of a match, it is considered
the
     linebreak, as normal.
  2: If the first capturing group is not part of a match, the leftmost
     capturing group which is part of a match will be considered the
linebreak.
  3: If no capturing group is part of the match, the linebreaker will
assume
```

```
     that the linebreak is a zero-length break immediately preceding the
match.

Example 1:  LINE_BREAKER = end(\n)begin|end2(\n)begin2|begin3

  * A line ending with 'end' followed a line beginning with 'begin'
would
    match the first branch, and the first capturing group would have a
match
    according to rule 1.  That particular newline would become a break
    between lines.
  * A line ending with 'end2' followed by a line beginning with
'begin2'
    would match the second branch and the second capturing group would
have
    a match.  That second capturing group would become the linebreak
    according to rule 2, and the associated newline would become a break
    between lines.
  * The text 'begin3' anywhere in the file at all would match the third
    branch, and there would be no capturing group with a match.  A
linebreak
    would be assumed immediately prior to the text 'begin3' so a
linebreak
    would be inserted prior to this text in accordance with rule 3.
 This
    means that a linebreak will occur before the text 'begin3' at any
    point in the text, whether a linebreak character exists or not.

Example 2: Example 1 would probably be better written as follows.  This
is
           not equivalent for all possible files, but for most real
files
           would be equivalent.

           LINE_BREAKER = end2?(\n)begin(2|3)?

LINE_BREAKER_LOOKBEHIND = <integer>
* When there is leftover data from a previous raw chunk,
  LINE_BREAKER_LOOKBEHIND indicates the number of bytes before the end
of
  the raw chunk (with the next chunk concatenated) that Splunk applies
the
  LINE_BREAKER regex. You may want to increase this value from its
default
  if you are dealing with especially large or multiline events.
* Defaults to 100 (bytes).

# Use the following attributes to specify how multiline events are
handled.

SHOULD_LINEMERGE = [true|false]
* When set to true, Splunk combines several lines of data into a single
```

```
  multiline event, based on the following configuration attributes.
* Defaults to true.

# When SHOULD_LINEMERGE is set to true, use the following attributes to
# define how Splunk builds multiline events.

BREAK_ONLY_BEFORE_DATE = [true|false]
* When set to true, Splunk creates a new event only if it encounters a
new
  line with a date.
  * Note, when using DATETIME_CONFIG = CURRENT or NONE, this setting is
not
    meaningful, as timestamps are not identified.
* Defaults to true.

BREAK_ONLY_BEFORE = <regular expression>
* When set, Splunk creates a new event only if it encounters a new line
that
  matches the regular expression.
* Defaults to empty.

MUST_BREAK_AFTER = <regular expression>
* When set and the regular expression matches the current line, Splunk
  creates a new event for the next input line.
* Splunk may still break before the current line if another rule
matches.
* Defaults to empty.

MUST_NOT_BREAK_AFTER = <regular expression>
* When set and the current line matches the regular expression, Splunk
does
  not break on any subsequent lines until the MUST_BREAK_AFTER
expression
  matches.
* Defaults to empty.

MUST_NOT_BREAK_BEFORE = <regular expression>
* When set and the current line matches the regular expression, Splunk
does
  not break the last event before the current line.
* Defaults to empty.

MAX_EVENTS = <integer>
* Specifies the maximum number of input lines to add to any event.
* Splunk breaks after the specified number of lines are read.
* Defaults to 256 (lines).

# Use the following attributes to handle better load balancing from UF.
# Please note the EVENT_BREAKER properties are applicable for Splunk
Universal
# Forwarder instances only.
```

```
EVENT_BREAKER_ENABLE = [true|false]
* When set to true, Splunk will split incoming data with a light-weight
  chunked line breaking processor so that data is distributed fairly
evenly
  amongst multiple indexers. Use this setting on the UF to indicate that
  data should be split on event boundaries across indexers especially
  for large files.
* Defaults to false

# Use the following to define event boundaries for multi-line events
# For single-line events, the default settings should suffice

EVENT_BREAKER = <regular expression>
* When set, Splunk will use the setting to define an event boundary at
the
  end of the first matching group instance.
```

Timestamp extraction configuration

```
#********************************************************************************
# Timestamp extraction configuration
#********************************************************************************Timestam
extraction configuration

DATETIME_CONFIG = <filename relative to $SPLUNK_HOME>
* Specifies which file configures the timestamp extractor, which
identifies
  timestamps from the event text.
* This configuration may also be set to "NONE" to prevent the timestamp
  extractor from running or "CURRENT" to assign the current system time
to
  each event.
  * "CURRENT" will set the time of the event to the time that the event
was
    merged from lines, or worded differently, the time it passed through
the
    aggregator processor.
  * "NONE" will leave the event time set to whatever time was selected
by
    the input layer
    * For data sent by splunk forwarders over the splunk protocol, the
input
      layer will be the time that was selected on the forwarder by its
input
      behavior (as below).
    * For file-based inputs (monitor, batch) the time chosen will be
the
      modification timestamp on the file being read.
    * For other inputs, the time chosen will be the current system time
```

```
when
      the event is read from the pipe/socket/etc.
  * Both "CURRENT" and "NONE" explicitly disable the per-text timestamp
    identification, so the default event boundary detection
    (BREAK_ONLY_BEFORE_DATE = true) is likely to not work as desired.
 When
    using these settings, use SHOULD_LINEMERGE and/or the BREAK_ONLY_* ,
    MUST_BREAK_* settings to control event merging.
* Defaults to /etc/datetime.xml (for example,
$SPLUNK_HOME/etc/datetime.xml).

TIME_PREFIX = <regular expression>
* If set, splunk scans the event text for a match for this regex in
event
  text before attempting to extract a timestamp.
* The timestamping algorithm only looks for a timestamp in the text
  following the end of the first regex match.
* For example, if TIME_PREFIX is set to "abc123", only text following
the
  first occurrence of the text abc123 will be used for timestamp
extraction.
* If the TIME_PREFIX cannot be found in the event text, timestamp
extraction
  will not occur.
* Defaults to empty.

MAX_TIMESTAMP_LOOKAHEAD = <integer>
* Specifies how far (in characters) into an event Splunk should look for
a
  timestamp.
* This constraint to timestamp extraction is applied from the point of
the
  TIME_PREFIX-set location.
* For example, if TIME_PREFIX positions a location 11 characters into
the
  event, and MAX_TIMESTAMP_LOOKAHEAD is set to 10, timestamp extraction
will
  be constrained to characters 11 through 20.
* If set to 0, or -1, the length constraint for timestamp recognition is
  effectively disabled.  This can have negative performance implications
  which scale with the length of input lines (or with event size when
  LINE_BREAKER is redefined for event splitting).
* Defaults to 150 (characters).

TIME_FORMAT = <strptime-style format>
* Specifies a strptime format string to extract the date.
* strptime is an industry standard for designating time formats.
* For more information on strptime, see "Configure timestamp
recognition" in
  the online documentation.
* TIME_FORMAT starts reading after the TIME_PREFIX. If both are
specified,
```

```
  the TIME_PREFIX regex must match up to and including the character
before
  the TIME_FORMAT date.
* For good results, the <strptime-style format> should describe the day
of
  the year and the time of day.
* Defaults to empty.

TZ = <timezone identifier>
* The algorithm for determining the time zone for a particular event is
as
  follows:
* If the event has a timezone in its raw text (for example, UTC,
-08:00),
  use that.
* If TZ is set to a valid timezone string, use that.
* If the event was forwarded, and the forwarder-indexer connection is
using
  the 6.0+ forwarding protocol, use the timezone provided by the
forwarder.
* Otherwise, use the timezone of the system that is running splunkd.
* Defaults to empty.

TZ_ALIAS = <key=value>[,<key=value>]...
* Provides splunk admin-level control over how timezone strings
extracted
  from events are interpreted.
  * For example, EST can mean Eastern (US) Standard time, or Eastern
    (Australian) Standard time.  There are many other three letter
timezone
    acronyms with many expansions.
* There is no requirement to use TZ_ALIAS if the traditional Splunk
default
  mappings for these values have been as expected.  For example, EST
maps to
  the Eastern US by default.
* Has no effect on TZ value; this only affects timezone strings from
event
  text, either from any configured TIME_FORMAT, or from pattern-based
guess
  fallback.
* The setting is a list of key=value pairs, separated by commas.
  * The key is matched against the text of the timezone specifier of
the
    event, and the value is the timezone specifier to use when mapping
the
    timestamp to UTC/GMT.
  * The value is another TZ specifier which expresses the desired
offset.
  * Example: TZ_ALIAS = EST=GMT+10:00 (See props.conf.example for
more/full
    examples)
```

```
* Defaults to unset.

MAX_DAYS_AGO = <integer>
* Specifies the maximum number of days in the past, from the current
date as
  provided by input layer(For e.g. forwarder current time, or modtime
for files),
  that an extracted date can be valid. Splunk still indexes events with
dates
  older than MAX_DAYS_AGO with the timestamp of the last acceptable
event. If no
  such acceptable event exists, new events with timestamps older than
  MAX_DAYS_AGO will use the current timestamp.
* For example, if MAX_DAYS_AGO = 10, Splunk applies the timestamp of the
last
  acceptable event to events with extracted timestamps older than 10
days in the
  past. If no acceptable event exists, Splunk applies the current
timestamp.
* Defaults to 2000 (days), maximum 10951.
* IMPORTANT: If your data is older than 2000 days, increase this
setting.

MAX_DAYS_HENCE = <integer>
* Specifies the maximum number of days in the future, from the current
date as
  provided by input layer(For e.g. forwarder current time, or modtime
for files),
  that an extracted date can be valid. Splunk still indexes events with
dates
  more than MAX_DAYS_HENCE in the future with the timestamp of the last
  acceptable event. If no such acceptable event exists, new events with
  timestamps after MAX_DAYS_HENCE will use the current timestamp.
* For example, if MAX_DAYS_HENCE = 3, Splunk applies the timestamp of
the last
  acceptable event to events with extracted timestamps more than 3 days
in the
  future. If no acceptable event exists, Splunk applies the current
timestamp.
* The default value includes dates from one day in the future.
* If your servers have the wrong date set or are in a timezone that is
one
  day ahead, increase this value to at least 3.
* Defaults to 2 (days), maximum 10950.
* IMPORTANT: False positives are less likely with a tighter window,
change
             with caution.

MAX_DIFF_SECS_AGO = <integer>
* This setting prevents Splunk Enterprise from rejecting events with
timestamps
  that are out of order.
```

```
* Do not use this setting to filter events because Splunk Enterprise
uses
  complicated heuristics for time parsing.
* Splunk Enterprise warns you if an event timestamp is more than
<integer>
  seconds BEFORE the previous timestamp and does not have the same time
  format as the majority of timestamps from the source.
* After Splunk Enterprise throws the warning, it only rejects an event
if it
  cannot apply a timestamp to the event (for example, if Splunk cannot
  recognize the time of the event.)
* IMPORTANT: If your timestamps are wildly out of order, consider
increasing
  this value.
* Note: if the events contain time but not date (date determined
another way,
  such as from a filename) this check will only consider the hour. (No
one
  second granularity for this purpose.)
* Defaults to 3600 (one hour), maximum 2147483646.

MAX_DIFF_SECS_HENCE = <integer>
* This setting prevents Splunk Enterprise from rejecting events with
timestamps
  that are out of order.
* Do not use this setting to filter events because Splunk Enterprise
uses
  complicated heuristics for time parsing.
* Splunk Enterprise warns you if an event timestamp is more than
<integer>
  seconds AFTER the previous timestamp and does not have the same time
format
  as the majority of timestamps from the source.
* After Splunk Enterprise throws the warning, it only rejects an event
if it
  cannot apply a timestamp to the event (for example, if Splunk cannot
  recognize the time of the event.)
* IMPORTANT: If your timestamps are wildly out of order, or you have
logs that
  are written less than once a week, consider increasing this value.
* Defaults to 604800 (one week), maximum 2147483646.
```

Structured Data Header Extraction and configuration

```
#******************************************************************************
# Structured Data Header Extraction and configuration
#*******************************************************************************Structur
Data Header Extraction and configuration
```

```
* This feature and all of its settings apply at input time, when data is
  first read by Splunk.  The setting is used on a Splunk system that has
  configured inputs acquiring the data.

# Special characters for Structured Data Header Extraction:
# Some unprintable characters can be described with escape sequences.
The
# attributes that can use these characters specifically mention that
# capability in their descriptions below.
# \f : form feed       byte: 0x0c
# \s : space           byte: 0x20
# \t : horizontal tab  byte: 0x09
# \v : vertical tab    byte: 0x0b

INDEXED_EXTRACTIONS = < CSV|W3C|TSV|PSV|JSON >
* Tells Splunk the type of file and the extraction and/or parsing method
  Splunk should use on the file.
  CSV  - Comma separated value format
  TSV  - Tab-separated value format
  PSV  - pipe "|" separated value format
  W3C  - W3C Extended Extended Log File Format
  JSON - JavaScript Object Notation format
* These settings default the values of the remaining settings to the
  appropriate values for these known formats.
* Defaults to unset.

PREAMBLE_REGEX = <regex>
* Some files contain preamble lines. This attribute specifies a regular
  expression which allows Splunk to ignore these preamble lines, based
on
  the pattern specified.

FIELD_HEADER_REGEX = <regex>
* A regular expression that specifies a pattern for prefixed headers.
Note
  that the actual header starts after the pattern and it is not included
in
  the header field.
* This attribute supports the use of the special characters described
above.

HEADER_FIELD_LINE_NUMBER = <integer>
* Tells Splunk the line number of the line within the file that
contains the
  header fields.  If set to 0, Splunk attempts to locate the header
fields
  within the file automatically.
* The default value is set to 0.

FIELD_DELIMITER = <character>
* Tells Splunk which character delimits or separates fields in the
```

```
specified
  file or source.
* This attribute supports the use of the special characters described
above.

HEADER_FIELD_DELIMITER = <character>
* Tells Splunk which character delimits or separates header fields in
the
  specified file or source.
* This attribute supports the use of the special characters described
above.

FIELD_QUOTE = <character>
* Tells Splunk the character to use for quotes in the specified file or
  source.
* This attribute supports the use of the special characters described
above.

HEADER_FIELD_QUOTE = <character>
* Specifies Splunk the character to use for quotes in the header of the
  specified file or source.
* This attribute supports the use of the special characters described
above.

TIMESTAMP_FIELDS = [ <string>,..., <string>]
* Some CSV and structured files have their timestamp encompass multiple
  fields in the event separated by delimiters. This attribue tells
Splunk to
  specify all such fields which constitute the timestamp in a
  comma-separated fashion.
* If not specified, Splunk tries to automatically extract the timestamp
of
  the event.

FIELD_NAMES = [ <string>,..., <string>]
* Some CSV and structured files might have missing headers. This
attribute
  tells Splunk to specify the header field names directly.

MISSING_VALUE_REGEX = <regex>
* Tells Splunk the placeholder to use in events where no value is
present.

JSON_TRIM_BRACES_IN_ARRAY_NAMES = <bool>
* Tell the json parser not to add the curly braces to array names.
* Note that enabling this will make json indextime extracted array fiels
names
  inconsistant with spath search processor's naming convention.
* For a json document containing the following array object, with
trimming
  enabled a indextime field 'mount_point' will be generated instead of
the
```

```
  spath consistant field 'mount_point{}'
      "mount_point": ["/disk48","/disk22"]
* Defaults to false.
```

Field extraction configuration

```
#*******************************************************************************
# Field extraction configuration
#*******************************************************************************Field
extraction configuration

NOTE: If this is your first time configuring field extractions in
      props.conf, review the following information first.

There are three different "field extraction types" that you can use to
configure field extractions: TRANSFORMS, REPORT, and EXTRACT. They
differ in
two significant ways: 1) whether they create indexed fields (fields
extracted at index time) or extracted fields (fields extracted at search
time), and 2), whether they include a reference to an additional
component
called a "field transform," which you define separately in
transforms.conf.

**Field extraction configuration: index time versus search time**

Use the TRANSFORMS field extraction type to create index-time field
extractions. Use the REPORT or EXTRACT field extraction types to create
search-time field extractions.

NOTE: Index-time field extractions have performance implications.
Creating
      additions to Splunk's default set of indexed fields is ONLY
      recommended in specific circumstances.  Whenever possible,
extract
      fields only at search time.

There are times when you may find that you need to change or add to
your set
of indexed fields. For example, you may have situations where certain
search-time field extractions are noticeably impacting search
performance.
This can happen when the value of a search-time extracted field exists
outside of the field more often than not. For example, if you commonly
search a large event set with the expression company_id=1 but the value
1
occurs in many events that do *not* have company_id=1, you may want to
add
company_id to the list of fields extracted by Splunk at index time. This
```

is
because at search time, Splunk will want to check each instance of the value
1 to see if it matches company_id, and that kind of thing slows down performance when you have Splunk searching a large set of data.

Conversely, if you commonly search a large event set with expressions like
company_id!=1 or NOT company_id=1, and the field company_id nearly *always*
takes on the value 1, you may want to add company_id to the list of fields
extracted by Splunk at index time.

For more information about index-time field extraction, search the documentation for "index-time extraction." For more information about search-time field extraction, search the online documentation for "search-time extraction."

Field extraction configuration: field transforms vs. "inline" (props.conf only) configs

The TRANSFORMS and REPORT field extraction types reference an additional component called a field transform, which you define separately in transforms.conf. Field transforms contain a field-extracting regular expression and other attributes that govern the way that the transform extracts fields. Field transforms are always created in conjunction with field extraction stanzas in props.conf; they do not stand alone.

The EXTRACT field extraction type is considered to be "inline," which means
that it does not reference a field transform. It contains the regular expression that Splunk uses to extract fields at search time. You can use
EXTRACT to define a field extraction entirely within props.conf--no transforms.conf component is required.

Search-time field extractions: Why use REPORT if EXTRACT will do?

It's a good question. And much of the time, EXTRACT is all you need for search-time field extraction. But when you build search-time field extractions, there are specific cases that require the use of REPORT and the
field transform that it references. Use REPORT if you want to:

* Reuse the same field-extracting regular expression across multiple sources, source types, or hosts. If you find yourself using the same regex
  to extract fields across several different sources, source types, and hosts, set it up as a transform, and then reference it in REPORT extractions in those stanzas. If you need to update the regex you only have to do it in one place. Handy!

```
* Apply more than one field-extracting regular expression to the same
  source, source type, or host. This can be necessary in cases where the
  field or fields that you want to extract from a particular source,
source
  type, or host appear in two or more very different event patterns.
* Set up delimiter-based field extractions. Useful if your event data
  presents field-value pairs (or just field values) separated by
delimiters
  such as commas, spaces, bars, and so on.
* Configure extractions for multivalued fields. You can have Splunk
append
  additional values to a field as it finds them in the event data.
* Extract fields with names beginning with numbers or underscores.
  Ordinarily, Splunk's key cleaning functionality removes leading
numeric
  characters and underscores from field names. If you need to keep them,
  configure your field transform to turn key cleaning off.
* Manage formatting of extracted fields, in cases where you are
extracting
  multiple fields, or are extracting both the field name and field
value.

**Precedence rules for TRANSFORMS, REPORT, and EXTRACT field extraction
types**

* For each field extraction, Splunk takes the configuration from the
highest
  precedence configuration stanza (see precedence rules at the beginning
of
  this file).
* If a particular field extraction is specified for a source and a
source
  type, the field extraction for source wins out.
* Similarly, if a particular field extraction is specified in ../local/
for
  a <spec>, it overrides that field extraction in ../default/.


TRANSFORMS-<class> = <transform_stanza_name>,
<transform_stanza_name2>,...
* Used for creating indexed fields (index-time field extractions).
* <class> is a unique literal string that identifies the namespace of
the
  field you're extracting.
  **Note:** <class> values do not have to follow field name syntax
  restrictions. You can use characters other than a-z, A-Z, and 0-9,
and
  spaces are allowed. <class> values are not subject to key cleaning.
* <transform_stanza_name> is the name of your stanza from
transforms.conf.
* Use a comma-separated list to apply multiple transform stanzas to a
single
```

```
  TRANSFORMS extraction. Splunk applies them in the list order. For
example,
  this sequence ensures that the [yellow] transform stanza gets applied
  first, then [blue], and then [red]:
        [source::color_logs]
        TRANSFORMS-colorchange = yellow, blue, red

REPORT-<class> = <transform_stanza_name>, <transform_stanza_name2>,...
* Used for creating extracted fields (search-time field extractions)
that
  reference one or more transforms.conf stanzas.
* <class> is a unique literal string that identifies the namespace of
the
  field you're extracting.
  **Note:** <class> values do not have to follow field name syntax
  restrictions. You can use characters other than a-z, A-Z, and 0-9,
and
  spaces are allowed. <class> values are not subject to key cleaning.
* <transform_stanza_name> is the name of your stanza from
transforms.conf.
* Use a comma-separated list to apply multiple transform stanzas to a
single
  REPORT extraction.
  Splunk applies them in the list order. For example, this sequence
insures
  that the [yellow] transform stanza gets applied first, then [blue],
and
  then [red]:
    [source::color_logs]
    REPORT-colorchange = yellow, blue, red

EXTRACT-<class> = [<regex>|<regex> in <src_field>]
* Used to create extracted fields (search-time field extractions) that
do
  not reference transforms.conf stanzas.
* Performs a regex-based field extraction from the value of the source
  field.
* <class> is a unique literal string that identifies the namespace of
the
  field you're extracting.
  **Note:** <class> values do not have to follow field name syntax
  restrictions. You can use characters other than a-z, A-Z, and 0-9,
and
  spaces are allowed. <class> values are not subject to key cleaning.
* The <regex> is required to have named capturing groups. When the
<regex>
  matches, the named capturing groups and their values are added to the
  event.
* dotall (?s) and multiline (?m) modifiers are added in front of the
regex.
  So internally, the regex becomes (?ms)<regex>.
* Use '<regex> in <src_field>' to match the regex against the values of
```

```
a
  specific field.  Otherwise it just matches against _raw (all raw event
  data).
* NOTE: <src_field> can only contain alphanumeric characters and
underscore
  (a-z, A-Z, 0-9, and _).
* If your regex needs to end with 'in <string>' where <string> is *not*
a
  field name, change the regex to end with '[i]n <string>' to ensure
that
  Splunk doesn't try to match <string> to a field name.

KV_MODE = [none|auto|auto_escaped|multi|json|xml]
* Used for search-time field extractions only.
* Specifies the field/value extraction mode for the data.
* Set KV_MODE to one of the following:
  * none: if you want no field/value extraction to take place.
  * auto: extracts field/value pairs separated by equal signs.
  * auto_escaped: extracts fields/value pairs separated by equal signs
and
                  honors \" and \\ as escaped sequences within quoted
                  values, e.g field="value with \"nested\" quotes"
  * multi: invokes the multikv search command to expand a tabular event
into
           multiple events.
  * xml : automatically extracts fields from XML data.
  * json: automatically extracts fields from JSON data.
* Setting to 'none' can ensure that one or more user-created regexes are
not
  overridden by automatic field/value extraction for a particular host,
  source, or source type, and also increases search performance.
* Defaults to auto.
* The 'xml' and 'json' modes will not extract any fields when used on
data
  that isn't of the correct format (JSON or XML).

AUTO_KV_JSON = [true|false]
* Used for search-time field extractions only.
* Specifies whether to try json extraction automatically.
* Defaults to true.

KV_TRIM_SPACES = true|false
* Modifies the behavior of KV_MODE when set to auto, and auto_escaped.
* Traditionally, automatically identified fields have leading and
trailing
  whitespace removed from their values.
  * Example event: 2014-04-04 10:10:45 myfield=" apples "
    would result in a field called 'myfield' with a value of 'apples'.
* If this value is set to false, then external whitespace then this
outer
  space is retained.
  * Example: 2014-04-04 10:10:45 myfield=" apples "
```

```
    would result in a field called 'myfield' with a value of ' apples '.
* The trimming logic applies only to space characters, not tabs, or
other
  whitespace.
* NOTE: The Splunk UI currently has limitations with displaying and
  interactively clicking on  fields that have leading or trailing
  whitespace.  Field values with leading or trailing spaces may not look
  distinct in the event viewer, and clicking on a field value will
typically
  insert the term into the search string without its embedded spaces.
  * These warts are not specific to this feature.  Any such embedded
spaces
    will behave this way.
  * The Splunk search language and included commands will respect the
spaces.
* Defaults to true.

CHECK_FOR_HEADER = [true|false]
* Used for index-time field extractions only.
* Set to true to enable header-based field extraction for a file.
* If the file has a list of columns and each event contains a field
value
  (without field name), Splunk picks a suitable header line to use to
for
  extracting field names.
* If the file has a list of columns and each event contains a field
value
  (without a field name), Splunk picks a suitable header line to use for
  field extraction.
* Can only be used on the basis of [<sourcetype>] or [source::<spec>],
  not [host::<spec>].
* Disabled when LEARN_SOURCETYPE = false.
* Will cause the indexed source type to have an appended numeral; for
  example, sourcetype-2, sourcetype-3, and so on.
* The field names are stored in etc/apps/learned/local/props.conf.
  * Because of this, this feature will not work in most environments
where
    the data is forwarded.
* This setting applies at input time, when data is first read by Splunk.
  The setting is used on a Splunk system that has configured inputs
  acquiring the data.
* Defaults to false.

SEDCMD-<class> = <sed script>
* Only used at index time.
* Commonly used to anonymize incoming data at index time, such as credit
  card or social security numbers. For more information, search the
online
  documentation for "anonymize data."
* Used to specify a sed script which Splunk applies to the _raw field.
* A sed script is a space-separated list of sed commands. Currently the
  following subset of sed commands is supported:
```

```
    * replace (s) and character substitution (y).
* Syntax:
    * replace - s/regex/replacement/flags
      * regex is a perl regular expression (optionally containing
capturing
        groups).
      * replacement is a string to replace the regex match. Use \n for
back
        references, where "n" is a single digit.
      * flags can be either: g to replace all matches, or a number to
        replace a specified match.
    * substitute - y/string1/string2/
      * substitutes the string1[i] with string2[i]

FIELDALIAS-<class> = (<orig_field_name> AS <new_field_name>)+
* Use this to apply aliases to a field. The original field is not
removed.
  This just means that the original field can be searched on using any
of
  its aliases.
* You can create multiple aliases for the same field.
* <orig_field_name> is the original name of the field.
* <new_field_name> is the alias to assign to the field.
* You can include multiple field alias renames in the same stanza.
* Field aliasing is performed at search time, after field extraction,
but
  before calculated fields (EVAL-* statements) and lookups.
  This means that:
        * Any field extracted at search time can be aliased.
        * You can specify a lookup based on a field alias.
        * You cannot alias a calculated field.

EVAL-<fieldname> = <eval statement>
* Use this to automatically run the <eval statement> and assign the
value of
  the output to <fieldname>. This creates a "calculated field."
* When multiple EVAL-* statements are specified, they behave as if they
are
* run in parallel, rather than in any particular sequence.
  For example say you have two statements: EVAL-x = y*2 and EVAL-y=100.
In
  this case, "x" will be assigned the original value of "y * 2," not the
  value of "y" after it is set to 100.
* Splunk processes calculated fields after field extraction and field
  aliasing but before lookups. This means that:
  * You can use a field alias in the eval statement for a calculated
    field.
  * You cannot use a field added through a lookup in an eval statement
for a
    calculated field.

LOOKUP-<class> = $TRANSFORM (<match_field> (AS
```

```
<match_field_in_event>)?)+ (OUTPUT|OUTPUTNEW (<output_field> (AS
<output_field_in_event>)? )+ )?
* At search time, identifies a specific lookup table and describes how
that
  lookup table should be applied to events.
* <match_field> specifies a field in the lookup table to match on.
  * By default Splunk looks for a field with that same name in the event
to
    match with (if <match_field_in_event> is not provided)
  * You must provide at least one match field. Multiple match fields
are
    allowed.
* <output_field> specifies a field in the lookup entry to copy into each
  matching event, where it will be in the field
<output_field_in_event>.
  * If you do not specify an <output_field_in_event> value, Splunk
    uses <output_field>.
  * A list of output fields is not required.
* If they are not provided, all fields in the lookup table except for
the
  match fields (and the timestamp field if it is specified) will be
output
  for each matching event.
* If the output field list starts with the keyword "OUTPUTNEW" instead
of
  "OUTPUT", then each outputfield is only written out if it did not
previous
  exist. Otherwise, the output fields are always overridden. Any event
that
  has all of the <match_field> values but no matching entry in the
lookup
  table clears all of the output fields.  NOTE that OUTPUTNEW behavior
has
  changed since 4.1.x (where *none* of the output fields were written to
if
  *any* of the output fields previously existed).
* Splunk processes lookups after it processes field extractions, field
  aliases, and calculated fields (EVAL-* statements). This means that
you
  can use extracted fields, aliased fields, and calculated fields to
specify
  lookups. But you can't use fields discovered by lookups in the
  configurations of extracted fields, aliased fields, or calculated
fields.
* The LOOKUP- prefix is actually case-insensitive. Acceptable variants
include:
   LOOKUP_<class> = [...]
   LOOKUP<class>  = [...]
   lookup_<class> = [...]
   lookup<class>  = [...]
```

Binary file configuration

```
#*******************************************************************************
# Binary file configuration
#*******************************************************************************Binary
file configuration

NO_BINARY_CHECK = [true|false]
* When set to true, Splunk processes binary files.
* Can only be used on the basis of [<sourcetype>], or
[source::<source>],
  not [host::<host>].
* Defaults to false (binary files are ignored).
* This setting applies at input time, when data is first read by Splunk.
  The setting is used on a Splunk system that has configured inputs
  acquiring the data.

detect_trailing_nulls = [auto|true|false]
* When enabled, Splunk will try to avoid reading in null bytes at the
end of
  a file.
* When false, splunk will assume that all the bytes in the file should
be
  read and indexed.
* Set this value to false for UTF-16 and other encodings (CHARSET)
values
  that can have null bytes as part of the character text.
* Subtleties of 'true' vs 'auto':
  * 'true' is the splunk-on-windows historical behavior of trimming all
null
           bytes.
  * 'auto' is currently a synonym for true but will be extended to be
           sensitive to the charset selected (ie quantized for
multi-byte
           encodings, and disabled for unsafe variable-width encdings)
* This feature was introduced to work around programs which foolishly
  pre-allocate their log files with nulls and fill in data later.  The
  well-known case is Internet Information Server.
* This setting applies at input time, when data is first read by Splunk.
  The setting is used on a Splunk system that has configured inputs
  acquiring the data.
* Defaults to false on *nix, true on windows.
```

Segmentation configuration

```
#*******************************************************************************
# Segmentation configuration
#*******************************************************************************Segmenta
```

```
configuration

SEGMENTATION = <segmenter>
* Specifies the segmenter from segmenters.conf to use at index time for
the
  host, source, or sourcetype specified by <spec> in the stanza heading.
* Defaults to indexing.

SEGMENTATION-<segment selection> = <segmenter>
* Specifies that Splunk Web should use the specific segmenter (from
  segmenters.conf) for the given <segment selection> choice.
* Default <segment selection> choices are: all, inner, outer, raw. For
more
  information see the Admin Manual.
* Do not change the set of default <segment selection> choices, unless
you
  have some overriding reason for doing so. In order for a changed set
of
  <segment selection> choices to appear in Splunk Web, you will need to
edit
  the Splunk Web UI.
```

File checksum configuration

```
#*******************************************************************************
# File checksum configuration
#*******************************************************************************File
checksum configuration

CHECK_METHOD = [endpoint_md5|entire_md5|modtime]
* Set CHECK_METHOD endpoint_md5 to have Splunk checksum of the first and
  last 256 bytes of a file. When it finds matches, Splunk lists the file
as
  already indexed and indexes only new data, or ignores it if there is
no
  new data.
* Set CHECK_METHOD = entire_md5 to use the checksum of the entire file.
* Set CHECK_METHOD = modtime to check only the modification time of the
  file.
* Settings other than endpoint_md5 cause Splunk to index the entire file
for
  each detected change.
* Important: this option is only valid for [source::<source>] stanzas.
* This setting applies at input time, when data is first read by Splunk.
  The setting is used on a Splunk system that has configured inputs
  acquiring the data.
* Defaults to endpoint_md5.

initCrcLength = <integer>
```

```
* See documentation in inputs.conf.spec.
```

Small file settings

```
#*******************************************************************************
# Small file settings
#*******************************************************************************Small
file settings

PREFIX_SOURCETYPE = [true|false]
* NOTE: this attribute is only relevant to the "[too_small]" sourcetype.
* Determines the source types that are given to files smaller than 100
  lines, and are therefore not classifiable.
* PREFIX_SOURCETYPE = false sets the source type to "too_small."
* PREFIX_SOURCETYPE = true sets the source type to
"<sourcename>-too_small",
  where "<sourcename>" is a cleaned up version of the filename.
  * The advantage of PREFIX_SOURCETYPE = true is that not all small
files
    are classified as the same source type, and wildcard searching is
often
    effective.
  * For example, a Splunk search of "sourcetype=access*" will retrieve
    "access" files as well as "access-too_small" files.
* This setting applies at input time, when data is first read by Splunk.
  The setting is used on a Splunk system that has configured inputs
  acquiring the data.
* Defaults to true.
```

Sourcetype configuration

```
#*******************************************************************************
# Sourcetype configuration
#*******************************************************************************Sourcety
configuration

sourcetype = <string>
* Can only be set for a [source::...] stanza.
* Anything from that <source> is assigned the specified source type.
* Is used by file-based inputs, at input time (when accessing logfiles)
such
  as on a forwarder, or indexer monitoring local files.
* sourcetype assignment settings on a system receiving forwarded splunk
data
  will not be applied to forwarded data.
```

```
* For logfiles read locally, data from logfiles matching <source> is
  assigned the specified source type.
* Defaults to empty.

# The following attribute/value pairs can only be set for a stanza that
# begins with [<sourcetype>]:

rename = <string>
* Renames [<sourcetype>] as <string> at search time
* With renaming, you can search for the [<sourcetype>] with
  sourcetype=<string>
* To search for the original source type without renaming it, use the
  field _sourcetype.
* Data from a a renamed sourcetype will only use the search-time
  configuration for the target sourcetype. Field extractions
  (REPORTS/EXTRACT) for this stanza sourcetype will be ignored.
* Defaults to empty.

invalid_cause = <string>
* Can only be set for a [<sourcetype>] stanza.
* If invalid_cause is set, the Tailing code (which handles uncompressed
  logfiles) will not read the data, but hand it off to other components
or
  throw an error.
* Set <string> to "archive" to send the file to the archive processor
  (specified in unarchive_cmd).
* When set to "winevt", this causes the file to be handed off to the
  eventlog input processor.
* Set to any other string to throw an error in the splunkd.log if you
are
  running Splunklogger in debug mode.
* This setting applies at input time, when data is first read by Splunk.
  The setting is used on a Splunk system that has configured inputs
  acquiring the data.
* Defaults to empty.

is_valid = [true|false]
* Automatically set by invalid_cause.
* This setting applies at input time, when data is first read by Splunk,
  such as on a forwarder.
* This setting applies at input time, when data is first read by Splunk.
  The setting is used on a Splunk system that has configured inputs
  acquiring the data.
* DO NOT SET THIS.
* Defaults to true.

unarchive_cmd = <string>
* Only called if invalid_cause is set to "archive".
* This field is only valid on [source::<source>] stanzas.
* <string> specifies the shell command to run to extract an archived
source.
* Must be a shell command that takes input on stdin and produces output
```

```
on
  stdout.
* Use _auto for Splunk's automatic handling of archive files (tar,
tar.gz,
  tgz, tbz, tbz2, zip)
* This setting applies at input time, when data is first read by Splunk.
  The setting is used on a Splunk system that has configured inputs
  acquiring the data.
* Defaults to empty.

unarchive_sourcetype = <string>
* Sets the source type of the contents of the matching archive file.
Use
  this field instead of the sourcetype field to set the source type of
  archive files that have the following extensions: gz, bz, bz2, Z.
* If this field is empty (for a matching archive file props lookup)
Splunk
  strips off the archive file's extension (.gz, bz etc) and lookup
another
  stanza to attempt to determine the sourcetype.
* This setting applies at input time, when data is first read by Splunk.
  The setting is used on a Splunk system that has configured inputs
  acquiring the data.
* Defaults to empty.

LEARN_SOURCETYPE = [true|false]
* Determines whether learning of known or unknown sourcetypes is
enabled.
  * For known sourcetypes, refer to LEARN_MODEL.
  * For unknown sourcetypes, refer to the rule:: and delayedrule::
    configuration (see below).
* Setting this field to false disables CHECK_FOR_HEADER as well (see
above).
* This setting applies at input time, when data is first read by Splunk.
  The setting is used on a Splunk system that has configured inputs
  acquiring the data.
* Defaults to true.

LEARN_MODEL = [true|false]
* For known source types, the file classifier adds a model file to the
  learned directory.
* To disable this behavior for diverse source types (such as sourcecode,
  where there is no good exemple to make a sourcetype) set LEARN_MODEL =
  false.
* This setting applies at input time, when data is first read by Splunk.
  The setting is used on a Splunk system that has configured inputs
  acquiring the data.
* Defaults to true.

maxDist = <integer>
* Determines how different a source type model may be from the current
file.
```

```
* The larger the maxDist value, the more forgiving Splunk will be with
  differences.
  * For example, if the value is very small (for example, 10), then
files
    of the specified sourcetype should not vary much.
  * A larger value indicates that files of the given source type can
vary
    quite a bit.
* If you're finding that a source type model is matching too broadly,
reduce
  its maxDist value by about 100 and try again. If you're finding that a
  source type model is being too restrictive, increase its maxDist value
by
  about 100 and try again.
* This setting applies at input time, when data is first read by Splunk.
  The setting is used on a Splunk system that has configured inputs
  acquiring the data.
* Defaults to 300.

# rule:: and delayedrule:: configuration

MORE_THAN<optional_unique_value>_<number> = <regular expression>
(empty)
LESS_THAN<optional_unique_value>_<number> = <regular expression>
(empty)

* These settingg apply at input time, when data is first read by Splunk,
  such as on a forwarder.

An example:

[rule::bar_some]
sourcetype = source_with_lots_of_bars
# if more than 80% of lines have "----", but fewer than 70% have "####"
# declare this a "source_with_lots_of_bars"
MORE_THAN_80 = ----
LESS_THAN_70 = ####

A rule can have many MORE_THAN and LESS_THAN patterns, and all are
required
for the rule to match.
```

Annotation Processor configured

```
#********************************************************************************
# Annotation Processor configured
#********************************************************************************Annotati
Processor configured
```

```
ANNOTATE_PUNCT = [true|false]
* Determines whether to index a special token starting with "punct::"
  * The "punct::" key contains punctuation in the text of the event.
    It can be useful for finding similar events
  * If it is not useful for your dataset, or if it ends up taking
    too much space in your index it is safe to disable it
* Defaults to true.
```

Header Processor configuration

```
#*******************************************************************************
# Header Processor configuration
#*******************************************************************************Header
Processor configuration

HEADER_MODE = <empty> | always | firstline | none
* Determines whether to use the inline ***SPLUNK*** directive to rewrite
index-time fields.
  * If "always", any line with ***SPLUNK*** can be used to rewrite
    index-time fields.
  * If "firstline", only the first line can be used to rewrite
    index-time fields.
  * If "none", the string ***SPLUNK*** is treated as normal data.
  * If <empty>, scripted inputs take the value "always" and file inputs
    take the value "none".
* This setting applies at input time, when data is first read by Splunk.
  The setting is used on a Splunk system that has configured inputs
  acquiring the data.
* Defaults to <empty>.
```

Internal settings

```
#*******************************************************************************
# Internal settings
#*******************************************************************************Internal
settings

# NOT YOURS. DO NOT SET.

_actions = <string>
* Internal field used for user-interface control of objects.
* Defaults to "new,edit,delete".

pulldown_type = <bool>
* Internal field used for user-interface control of source types.
* Defaults to empty.
```

```
given_type = <string>
* Internal field used by the CHECK_FOR_HEADER feature to remember the
  original sourcetype.
* This setting applies at input time, when data is first read by Splunk.
  The setting is used on a Splunk system that has configured inputs
  acquiring the data.
* Default to unset.
```

Sourcetype Category and Descriptions

```
#*******************************************************************************
# Sourcetype Category and Descriptions
#*******************************************************************************Sourcety
Category and Descriptions

description = <string>
* Field used to describe the sourcetype. Does not affect indexing
behaviour.
* Defaults to unset.

category = <string>
* Field used to classify sourcetypes for organization in the front end.
Case
  sensitive. Does not affect indexing behaviour.
* Defaults to unset.
```

props.conf.example

```
#   Version 6.5.0
#
# The following are example props.conf configurations. Configure
properties for
# your data.
#
# To use one or more of these configurations, copy the configuration
block into
# props.conf in $SPLUNK_HOME/etc/system/local/. You must restart Splunk
to
# enable configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

```
########
# Line merging settings
########

# The following example linemerges source data into multi-line events
for
# apache_error sourcetype.

[apache_error]
SHOULD_LINEMERGE = True



########
# Settings for tuning
########

# The following example limits the amount of characters indexed per
event from
# host::small_events.

[host::small_events]
TRUNCATE = 256

# The following example turns off DATETIME_CONFIG (which can speed up
indexing)
# from any path that ends in /mylogs/*.log.
#
# In addition, the default splunk behavior of finding event boundaries
# via per-event timestamps can't work with NONE, so we disable
# SHOULD_LINEMERGE, essentially declaring that all events in this file
are
# single-line.

[source::.../mylogs/*.log]
DATETIME_CONFIG = NONE
SHOULD_LINEMERGE = false



########
# Timestamp extraction configuration
########

# The following example sets Eastern Time Zone if host matches nyc*.

[host::nyc*]
TZ = US/Eastern
```

```
# The following example uses a custom datetime.xml that has been created
and
# placed in a custom app directory. This sets all events coming in from
hosts
# starting with dharma to use this custom file.

[host::dharma*]
DATETIME_CONFIG = <etc/apps/custom_time/datetime.xml>

########
## Timezone alias configuration
########

# The following example uses a custom alias to disambiguate the
Australian
# meanings of EST/EDT

TZ_ALIAS = EST=GMT+10:00,EDT=GMT+11:00

# The following example gives a sample case wherein, one timezone field
is
# being replaced by/interpreted as another.

TZ_ALIAS = EST=AEST,EDT=AEDT

########
# Transform configuration
########

# The following example creates a search field for host::foo if tied to
a
# stanza in transforms.conf.

[host::foo]
TRANSFORMS-foo=foobar

# The following stanza extracts an ip address from _raw
[my_sourcetype]
EXTRACT-extract_ip = (?<ip>\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})

# The following example shows how to configure lookup tables
[my_lookuptype]
LOOKUP-foo = mylookuptable userid AS myuserid OUTPUT username AS
myusername

# The following shows how to specify field aliases
FIELDALIAS-foo = user AS myuser id AS myid


########
# Sourcetype configuration
########
```

```
# The following example sets a sourcetype for the file web_access.log
for a
# unix path.

[source::.../web_access.log]
sourcetype = splunk_web_access

# The following example sets a sourcetype for the Windows file iis6.log.
 Note:
# Backslashes within Windows file paths must be escaped.

[source::...\\iis\\iis6.log]
sourcetype = iis_access

# The following example untars syslog events.

[syslog]
invalid_cause = archive
unarchive_cmd = gzip -cd -


# The following example learns a custom sourcetype and limits the range
between
# different examples with a smaller than default maxDist.

[custom_sourcetype]
LEARN_MODEL = true
maxDist = 30


# rule:: and delayedrule:: configuration
# The following examples create sourcetype rules for custom sourcetypes
with
# regex.


[rule::bar_some]
sourcetype = source_with_lots_of_bars
MORE_THAN_80 = ----


[delayedrule::baz_some]
sourcetype = my_sourcetype
LESS_THAN_70 = ####


########
# File configuration
########

# Binary file configuration
```

```
# The following example eats binary files from the sourcetype
# "imported_records".

[imported_records]
NO_BINARY_CHECK = true


# File checksum configuration
# The following example checks the entirety of every file in the
web_access dir
# rather than skipping files that appear to be the same.

[source::.../web_access/*]
CHECK_METHOD = entire_md5
```

pubsub.conf

The following are the spec and example files for pubsub.conf.

pubsub.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attributes and values for configuring a
client of
# the PubSub system (broker).
#
# To set custom configurations, place a pubsub.conf in
# $SPLUNK_HOME/etc/system/local/.
# For examples, see pubsub.conf.example. You must restart Splunk to
enable
# configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
```

```
top of
#     the file.
#   * Each conf file should have at most one default stanza. If there
are
#     multiple default stanzas, attributes are combined. In the case of
#     multiple definitions of the same attribute, the last definition in
the
#     file wins.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.

#*******************************************************************
# Configure the physical location where deploymentServer is running.
# This configuration is used by the clients of the pubsub system.
#*******************************************************************
```

[pubsub-server:deploymentServer]

```
[pubsub-server:deploymentServer]
disabled = <false or true>
* defaults to 'false'

targetUri = <IP:Port>|<hostname:Port>|direct
* specify either the url of a remote server in case the broker is
remote, or
  just the keyword "direct" when broker is in-process.
* It is usually a good idea to co-locate the broker and the Deployment
Server
  on the same Splunk. In such a configuration, all
* deployment clients would have targetUri set to deploymentServer:port.

#*******************************************************************
# The following section is only relevant to Splunk developers.
#*******************************************************************

# This "direct" configuration is always available, and cannot be
overridden.
```

[pubsub-server:direct]

```
[pubsub-server:direct]
disabled = false
targetUri = direct
```

[pubsub-server:<logicalName>]

```
[pubsub-server:<logicalName>]
* It is possible for any Splunk to be a broker. If you have multiple
brokers,
  assign a logicalName that is used by the clients to refer to it.

disabled = <false or true>
* defaults to 'false'

targetUri = <IP:Port>|<hostname:Port>|direct
* The Uri of a Splunk that is being used as a broker.
* The keyword "direct" implies that the client is running on the same
Splunk
  instance as the broker.
```

pubsub.conf.example

```
#   Version 6.5.0

[pubsub-server:deploymentServer]
disabled=false
targetUri=somehost:8089

[pubsub-server:internalbroker]
disabled=false
targetUri=direct
```

restmap.conf

The following are the spec and example files for restmap.conf.

restmap.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attribute and value pairs for creating new
# Representational State Transfer (REST) endpoints.
#
# There is a restmap.conf in $SPLUNK_HOME/etc/system/default/. To set
custom
```

```
# configurations, place a restmap.conf in
$SPLUNK_HOME/etc/system/local/. For
# help, see restmap.conf.example. You must restart Splunk to enable
# configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

# NOTE: You must register every REST endpoint via this file to make it
# available.

############################
# Global stanza

[global]
* This stanza sets global configurations for all REST endpoints.
* Follow this stanza name with any number of the following
attribute/value
  pairs.

allowGetAuth=[true|false]
* Allow user/password to be passed as a GET parameter to endpoint
  services/auth/login.
* Setting this to true, while convenient, may result in user/password
getting
  logged as cleartext in Splunk's logs *and* any proxy servers in
between.
* Defaults to false.

allowRestReplay=[true|false]
* POST/PUT/DELETE requests can be replayed on other nodes in the
deployment.
* This enables centralized management.
* Turn on or off this feature. You can also control replay at each
endpoint
  level. This feature is currently INTERNAL and should not be turned on
witout
  consulting splunk support.
* Defaults to false

defaultRestReplayStanza=<string>
* Points to global rest replay configuration stanza.
* Related to allowRestReplay
* Defaults to "restreplayshc"

pythonHandlerPath=<path>
* Path to 'main' python script handler.
* Used by the script handler to determine where the actual 'main'
script is
```

```
  located.
* Typically, you should not need to change this.
* Defaults to $SPLUNK_HOME/bin/rest_handler.py.

############################
# Applicable to all REST stanzas
# Stanza definitions below may supply additional information for these.
#

[<rest endpoint name>:<endpoint description string>]
match=<path>
* Specify the URI that calls the handler.
* For example if match=/foo, then https://$SERVER:$PORT/services/foo
calls this
  handler.
* NOTE: You must start your path with a /.

requireAuthentication=[true|false]
* This optional attribute determines if this endpoint requires
authentication.
* Defaults to 'true'.

authKeyStanza=<stanza>
* This optional attribute determines the location of the pass4SymmKey in
the
  server.conf to be used for endpoint authentication.
* Defaults to 'general' stanza.
* Only applicable if the requireAuthentication is set true.

restReplay=[true|false]
* This optional attribute enables rest replay on this endpoint group
* Related to allowRestReplay
* This feature is currently INTERNAL and should not be turned on without
consulting
  splunk support.
* Defaults to false

restReplayStanza=<string>
* This points to stanza which can override the
[global]/defaultRestReplayStanza
  value on a per endpoint/regex basis
* Defaults to empty

capability=<capabilityName>
capability.<post|delete|get|put>=<capabilityName>
* Depending on the HTTP method, check capabilities on the authenticated
session user.
* If you use 'capability.post|delete|get|put,' then the associated
method is
  checked against the authenticated user's role.
* If you just use 'capability,' then all calls get checked against this
  capability (regardless of the HTTP method).
```

```
acceptFrom=<network_acl> ...
* Lists a set of networks or addresses to allow this endpoint to be
accessed
  from.
* This shouldn't be confused with the setting of the same name in the
  [httpServer] stanza of server.conf which controls whether a host can
  make HTTP requests at all
* Each rule can be in the following forms:
    1. A single IPv4 or IPv6 address (examples: "10.1.2.3", "fe80::4a3")
    2. A CIDR block of addresses (examples: "10/8", "fe80:1234/32")
    3. A DNS name, possibly with a '*' used as a wildcard (examples:
       "myhost.example.com", "*.splunk.com")
    4. A single '*' which matches anything
* Entries can also be prefixed with '!' to cause the rule to reject the
  connection.  Rules are applied in order, and the first one to match is
  used.  For example, "!10.1/16, *" will allow connections from
everywhere
  except the 10.1.*.* network.
* Defaults to "*" (accept from anywhere)

includeInAccessLog=[true|false]
* If this is set to false, requests to this endpoint will not appear
  in splunkd_access.log
* Defaults to 'true'.

############################
# Per-endpoint stanza
# Specify a handler and other handler-specific settings.
# The handler is responsible for implementing arbitrary namespace
underneath
# each REST endpoint.

[script:<uniqueName>]
* NOTE: The uniqueName must be different for each handler.
* Call the specified handler when executing this endpoint.
* The following attribute/value pairs support the script handler.

scripttype=python
* Tell the system what type of script to execute when using this
endpoint.
* Defaults to python.
* If set to "persist" it will run the script via a persistent-process
that
  uses the protocol from persistconn/appserver.py.

handler=<SCRIPT>.<CLASSNAME>
* The name and class name of the file to execute.
* The file *must* live in an application's bin subdirectory.
* For example, $SPLUNK_HOME/etc/apps/<APPNAME>/bin/TestHandler.py has a
class
  called MyHandler (which, in the case of python must be derived from a
```

```
base
  class called 'splunk.rest.BaseRestHandler'). The tag/value pair for
this is:
  "handler=TestHandler.MyHandler".

xsl=<path to XSL transform file>
* Optional.
* Perform an optional XSL transform on data returned from the handler.
* Only use this if the data is XML.
* Does not apply to scripttype=persist.

script=<path to a script executable>
* For scripttype=python this is optional.  It allows you to run a
script
  which is *not* derived from 'splunk.rest.BaseRestHandler'.  This is
  rarely used.  Do not use this unless you know what you are doing.
* For scripttype=persist this is the path with is sent to the driver
  to execute.  In that case, environment variables are substituted.

script.arg.<N>=<string>
* Only has effect for scripttype=persist.
* List of arguments which are passed to the driver to start the script
.
* The script can make use of this information however it wants.
* Environment variables are substituted.

script.param=<string>
* Optional.
* Only has effect for scripttype=persist.
* Free-form argument that is passed to the driver when it starts the
  script.
* The script can make use of this information however it wants.
* Environment variables are substituted.

output_modes=<csv list>
* Specifies which output formats can be requested from this endpoint.
* Valid values are: json, xml.
* Defaults to xml.

passSystemAuth=<bool>
* Specifies whether or not to pass in a system-level authentication
token on
  each request.
* Defaults to false.

driver=<path>
* For scripttype=persist, specifies the command to start a persistent
  server for this process.
* Endpoints that share the same driver configuration can share
processes.
* Environment variables are substituted.
* Defaults to using the persistconn/appserver.py server.
```

```
driver.arg.<n> = <string>
* For scripttype=persist, specifies the command to start a persistent
  server for this process.
* Environment variables are substituted.
* Only takes effect when "driver" is specifically set.

driver.env.<name>=<value>
* For scripttype=persist, specifies an environment variable to set when
running
  the driver process.

passConf=<bool>
* If set, the script is sent the contents of this configuration stanza
  as part of the request.
* Only has effect for scripttype=persist.
* Defaults to true.

passPayload=[true | false | base64]
* If set to true, sends the driver the raw, unparsed body of the
  POST/PUT as a "payload" string.
* If set to "base64", the same body is instead base64-encoded and
  sent as a "payload_base64" string.
* Only has effect for scripttype=persist.
* Defaults to false.

passSession=<bool>
* If set to true, sends the driver information about the user's
  session.  This includes the user's name, an active authtoken,
  and other details.
* Only has effect for scripttype=persist.
* Defaults to true.

passHttpHeaders=<bool>
* If set to true, sends the driver the HTTP headers of the request.
* Only has effect for scripttype=persist.
* Defaults to false.

passHttpCookies=<bool>
* If set to true, sends the driver the HTTP cookies of the request.
* Only has effect for scripttype=persist.
* Defaults to false.

############################
# 'admin'
# The built-in handler for the Extensible Administration Interface.
# Exposes the listed EAI handlers at the given URL.
#

[admin:<uniqueName>]

match=<partial URL>
```

```
* URL which, when accessed, will display the handlers listed below.

members=<csv list>
* List of handlers to expose at this URL.
* See https://localhost:8089/services/admin for a list of all possible
  handlers.

############################
# 'admin_external'
# Register Python handlers for the Extensible Administration Interface.
# Handler will be exposed via its "uniqueName".
#

[admin_external:<uniqueName>]

handlertype=<script type>
* Currently only the value 'python' is valid.

handlerfile=<unique filename>
* Script to execute.
* For bin/myAwesomeAppHandler.py, specify only myAwesomeAppHandler.py.

handlerpersistentmode=[true|false]
* Set to true to run the script in persistent mode and keep the process
running
  between requests.

handleractions=<comma separated list>
* List of EAI actions supported by this handler.
* Valid values are: create, edit, list, delete, _reload.

########################
# Validation stanzas
# Add stanzas using the following definition to add arg validation to
# the appropriate EAI handlers.

[validation:<handler-name>]

<field> =  <validation-rule>

* <field> is the name of the field whose value would be validated when
an
  object is being saved.
* <validation-rule> is an eval expression using the validate() function
to
  evaluate arg correctness and return an error message. If you use a
boolean
  returning function, a generic message is displayed.
* <handler-name> is the name of the REST endpoint which this stanza
applies to
  handler-name is what is used to access the handler via
  /servicesNS/<user>/<app/admin/<handler-name>.
```

```
* For example:
  action.email.sendresult = validate(
isbool('action.email.sendresults'), "'action.email.sendresults' must be
a boolean value").
* NOTE: use ' or $ to enclose field names that contain non alphanumeric
characters.

############################
# 'eai'
# Settings to alter the behavior of EAI handlers in various ways.
# These should not need to be edited by users.
#

[eai:<EAI handler name>]

showInDirSvc = [true|false]
* Whether configurations managed by this handler should be enumerated
via the
  directory service, used by SplunkWeb's "All Configurations" management
page.
  Defaults to false.

desc = <human readable string>
* Allows for renaming the configuration type of these objects when
enumerated
  via the directory service.


############################
# Miscellaneous
# The un-described parameters in these stanzas all operate according to
the
# descriptions listed under "script:", above.
# These should not need to be edited by users - they are here only to
quiet
# down the configuration checker.
#

[input:...]
dynamic = [true|false]
* If set to true, listen on the socket for data.
* If false, data is contained within the request body.
* Defaults to false.

[peerupload:...]
path = <directory path>
* Path to search through to find configuration bundles from search
peers.

untar = [true|false]
* Whether or not a file should be untarred once the transfer is
complete.
```

```
[restreplayshc]
methods =  <comma separated strings>
* REST methods which will be replayed. POST, PUT, DELETE, HEAD, GET are
the
  available options

nodelists = <comma separated string>
* strategies for replay. Allowed values are shc, nodes, filternodes
* shc - replay to all other nodes in Search Head Cluster
* nodes - provide raw comma separated URIs in nodes variable
* filternodes - filter out specific nodes. Always applied after other
  strategies

nodes = <comma separated management uris>
* list of specific nodes that you want the REST call to be replayed to

filternodes = <comma separated management uris>
* list of specific nodes that you do not want the REST call to be
replayed to

[proxy:appsbrowser]
destination = <splunkbaseAPIURL>
* protocol, subdomain, domain, port, and path of the splunkbase api used
to browse apps
* Defaults to https://splunkbase.splunk.com/api
```

restmap.conf.example

```
#   Version 6.5.0
#
# This file contains example REST endpoint configurations.
#
# To use one or more of these configurations, copy the configuration
block into
# restmap.conf in $SPLUNK_HOME/etc/system/local/. You must restart
Splunk to
# enable configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles


# The following are default REST configurations.  To create your own
endpoints,
```

```
# modify the values by following the spec outlined in
restmap.conf.spec.


#
////////////////////////////////////////////////////////////////////////////
#  global settings
#
////////////////////////////////////////////////////////////////////////////

[global]

# indicates if auths are allowed via GET params
allowGetAuth=false

#The default handler (assuming that we have PYTHONPATH set)
pythonHandlerPath=$SPLUNK_HOME/bin/rest_handler.py



#
////////////////////////////////////////////////////////////////////////////
#  internal C++ handlers
# NOTE: These are internal Splunk-created endpoints. 3rd party
developers can
# only use script or search can be used as handlers.
# (Please see restmap.conf.spec for help with configurations.)
#
////////////////////////////////////////////////////////////////////////////

[SBA:sba]
match=/properties
capability=get_property_map

[asyncsearch:asyncsearch]
match=/search
capability=search
```

savedsearches.conf

The following are the spec and example files for savedsearches.conf.

savedsearches.conf.spec

```
#   Version 6.5.0
```

```
#
# This file contains possible attribute/value pairs for saved search
entries in
# savedsearches.conf.  You can configure saved searches by creating
your own
# savedsearches.conf.
#
# There is a default savedsearches.conf in
$SPLUNK_HOME/etc/system/default. To
# set custom configurations, place a savedsearches.conf in
# $SPLUNK_HOME/etc/system/local/.  For examples, see
# savedsearches.conf.example. You must restart Splunk to enable
configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#  * You can also define global settings outside of any stanza, at the
top of
#    the file.
#  * Each conf file should have at most one default stanza. If there
are
#    multiple default stanzas, attributes are combined. In the case of
multiple
#    definitions of the same attribute, the last definition in the file
wins.
#  * If an attribute is defined at both the global level and in a
specific
#    stanza, the value in the specific stanza takes precedence.
```

The possible attribute/value pairs for savedsearches.conf are:

```
#*******
# The possible attribute/value pairs for savedsearches.conf are:
#*******The possible attribute/value pairs for savedsearches.conf are:

[<stanza name>]
* Create a unique stanza name for each saved search.
* Follow the stanza name with any number of the following
```

```
attribute/value
  pairs.
* If you do not specify an attribute, Splunk uses the default.

disabled = [0|1]
* Disable your search by setting to 1.
* A disabled search cannot run until it is enabled.
* This setting is typically used to keep a scheduled search from running
on
  its schedule without deleting the search definition.
* Defaults to 0.

search = <string>
* Actual search terms of the saved search.
* For example, search = index::sampledata http NOT 500.
* Your search can include macro searches for substitution.
* To learn more about creating a macro search, search the documentation
for
  "macro search."
* Multi-line search strings currently have some limitations.  For
example use
  with the search command '|savedseach' does not currently work with
multi-line
  search strings.
* Defaults to empty string.

dispatchAs = [user|owner]
* When the saved search is dispatched via the
"saved/searches/{name}/dispatch"
  endpoint, this setting controls, what user that search is dispatched
as.
* This setting is only meaningful for shared saved searches.
* When dispatched as user it will be executed as if the requesting user
owned
  the search.
* When dispatched as owner it will be executed as if the owner of the
search
  dispatched it no matter what user requested it.
* If the 'force_saved_search_dispatch_as_user' attribute, in the
limits.conf
  file, is set to true then the dispatchAs attribute is reset to 'user'
while
  the saved search is dispatching.
* Defaults to owner.
```

Scheduling options

```
#*******
```

```
# Scheduling options
#*******Scheduling options

enableSched = [0|1]
* Set this to 1 to run your search on a schedule.
* Defaults to 0.

cron_schedule = <cron string>
* The cron schedule used to execute this search.
* For example: */5 * * * *  causes the search to execute every 5
minutes.
* Cron lets you use standard cron notation to define your scheduled
search
  interval.
  In particular, cron can accept this type of notation: 00,20,40 * * *
*, which
  runs the search every hour at hh:00, hh:20, hh:40. Along the same
lines, a
  cron of 03,23,43 * * * * runs the search every hour at hh:03, hh:23,
hh:43.
* Splunk recommends that you schedule your searches so that they are
staggered
  over time. This reduces system load. Running all of them every 20
minutes
  (*/20) means they would all launch at hh:00 (20, 40) and might slow
your
  system every 20 minutes.
* Splunk's cron implementation does not currently support names of
months/days.
* Defaults to empty string.

schedule = <cron-style string>
* This field is DEPRECATED as of 4.0.
* For more information, see the pre-4.0 spec file.
* Use cron_schedule to define your scheduled search interval.


max_concurrent = <unsigned int>
* The maximum number of concurrent instances of this search the
scheduler is
  allowed to run.
* Defaults to 1.

realtime_schedule = [0|1]
* Controls the way the scheduler computes the next execution time of a
  scheduled search.
* If this value is set to 1, the scheduler bases its determination of
the next
  scheduled search execution time on the current time.
* If this value is set to 0, the scheduler bases its determination of
the next
  scheduled search on the last search execution time. This is called
```

```
continuous
  scheduling.
  * If set to 1, the scheduler might skip some execution periods to
make sure
    that the scheduler is executing the searches running over the most
recent
    time range.
  * If set to 0, the scheduler never skips scheduled execution periods.
  * However, the execution
    of the saved search might fall behind depending on the scheduler's
load.
    Use continuous scheduling whenever you enable the summary index
option.
* The scheduler tries to execute searches that have realtime_schedule
set to 1
  before it executes searches that have continuous scheduling
  (realtime_schedule = 0).
* Defaults to 1

schedule_priority = default | higher | highest
* Raises scheduling priority of a search:
  + "default": No scheduling priority increase.
  + "higher": Scheduling priority is higher than other searches of the
same
    scheduling tier. While there are four tiers of priority for
scheduled
    searches, only the following are affected by this property:
    1. Real-Time-Scheduled (realtime_schedule=1).
    2. Continuous-Scheduled (realtime_schedule=0).
  + "highest": Scheduling priority is higher than other searches
regardless of
    scheduling tier. However, real-time-scheduled searches with
priority =
    highest always have priority over continuous scheduled searches with
    priority = highest.
  + Hence, the high-to-low order (where RTSS = real-time-scheduled
search,
    CSS = continuous-scheduled search, d = default, h = higher, H =
highest)
    is: RTSS(H) > CSS(H) > RTSS(h) > RTSS(d) > CSS(h) > CSS(d)
* The scheduler honors a non-default priority only when the search owner
has
  the 'edit_search_schedule_priority' capability.
* Defaults to "default".
* A non-default priority is mutually exclusive with a non-zero
'schedule_window'
  (see below).  If a user specifies both for a scheduled search, the
scheduler
  honors the priority only.
* However, if a user specifies both settings for a search, but the
search owner
  does not have the 'edit_search_scheduler_priority' capability, then
```

```
the
  scheduler ignores the priority setting and honors the
'schedule_window'.
* WARNING: Having too many searches with a non-default priority will
impede the
  ability of the scheduler to minimize search starvation.  Use this
setting
  only for mission-critical searches.

schedule_window = <unsigned int> | auto
* When schedule_window is non-zero, it indicates to the scheduler that
the
  search does not require a precise start time. This gives the scheduler
  greater flexibility when it prioritizes searches.
* When schedule_window is set to an integer greater than 0, it specifies
the
  "window" of time (in minutes) a search may start within.
  + The schedule_window must be shorter than the period of the search.
  + Schedule windows are not recommended for searches that run every
minute.
* When set to 0, there is no schedule window. The scheduler starts the
search
  as close to its scheduled time as possible.
* When set to "auto," the scheduler calculates the schedule_window value
  automatically.
  + For more information about this calculation, see the search
scheduler
    documentation.
* Defaults to 0 for searches that are owned by users with the
  edit_search_schedule_window capability. For such searches, this value
can be
  changed.
* Defaults to "auto" for searches that are owned by users that do not
have the
  edit_search_window capability. For such searches, this setting cannot
be
  changed.
* A non-zero schedule_window is mutually exclusive with a non-default
  schedule_priority (see schedule_priority for details).
```

Notification options

```
#*******
# Notification options
#*******Notification options

counttype = number of events | number of hosts | number of sources |
always
```

```
* Set the type of count for alerting.
* Used with relation and quantity (below).
* NOTE: If you specify "always," do not set relation or quantity
(below).
* Defaults to always.

relation = greater than | less than | equal to | not equal to | drops by
| rises by
* Specifies how to compare against counttype.
* Defaults to empty string.

quantity = <integer>
* Specifies a value for the counttype and relation, to determine the
condition
  under which an alert is triggered by a saved search.
* You can think of it as a sentence constructed like this: <counttype>
<relation> <quantity>.
* For example, "number of events [is] greater than 10" sends an alert
when the
  count of events is larger than by 10.
* For example, "number of events drops by 10%" sends an alert when the
count of
  events drops by 10%.
* Defaults to an empty string.

alert_condition = <search string>
* Contains a conditional search that is evaluated against the results of
the
  saved search.  Alerts are triggered if the specified search yields a
  non-empty search result list.
* NOTE: If you specify an alert_condition, do not set counttype,
relation, or
        quantity.
* Defaults to an empty string.


#*******
# generic action settings.
# For a comprehensive list of actions and their arguments, refer to
# alert_actions.conf.
#*******

action.<action_name> = 0 | 1
* Indicates whether the action is enabled or disabled for a particular
saved
  search.
* The action_name can be: email | populate_lookup | script |
summary_index
* For more about your defined alert actions see alert_actions.conf.
* Defaults to an empty string.

action.<action_name>.<parameter> = <value>
```

```
* Overrides an action's parameter (defined in alert_actions.conf) with
a new
  <value> for this saved search only.
* Defaults to an empty string.
```

Settings for email action

```
#******
# Settings for email action
#******Settings for email action

action.email = 0 | 1
* Enables or disables the email action.
* Defaults to 0.

action.email.to = <email list>
* REQUIRED. This setting is not defined in alert_actions.conf.
* Set a comma-delimited list of recipient email addresses.
* Defaults to empty string.

* When configured in Splunk Web, the following email settings
  are written to this conf file only if their values differ
  from settings in alert_actions.conf.

action.email.from = <email address>
* Set an email address to use as the sender's address.
* Defaults to splunk@<LOCALHOST> (or whatever is set in
alert_actions.conf).

action.email.subject = <string>
* Set the subject of the email delivered to recipients.
* Defaults to SplunkAlert-<savedsearchname> (or whatever is set
  in alert_actions.conf).

action.email.mailserver = <string>
* Set the address of the MTA server to be used to send the emails.
* Defaults to <LOCALHOST> (or whatever is set in alert_actions.conf).

action.email.maxresults = <integer>
* Set the maximum number of results to be emailed.
* Any alert-level results threshold greater than this number will be
capped at
  this level.
* This value affects all methods of result inclusion by email alert:
inline,
  CSV and PDF.
* Note that this setting is affected globally by "maxresults" in the
[email]
```

```
  stanza of alert_actions.conf.
* Defaults to 10000

action.email.include.results_link = [1|0]
* Specify whether to include a link to search results in the
  alert notification email.
* Defaults to 1 (or whatever is set in alert_actions.conf).

action.email.include.search = [1|0]
* Specify whether to include the query whose results triggered the
email.
* Defaults to 0 (or whatever is set in alert_actions.conf).

action.email.include.trigger = [1|0]
* Specify whether to include the alert trigger condition.
* Defaults to 0 (or whatever is set in alert_actions.conf).

action.email.include.trigger_time = [1|0]
* Specify whether to include the alert trigger time.
* Defaults to 0 (or whatever is set in alert_actions.conf).

action.email.include.view_link = [1|0]
* Specify whether to include saved search title and a link for editing
  the saved search.
* Defaults to 1 (or whatever is set in alert_actions.conf).

action.email.inline = [1|0]
* Specify whether to include search results in the body of the
  alert notification email.
* Defaults to 0 (or whatever is set in alert_actions.conf).

action.email.sendcsv = [1|0]
* Specify whether to send results as a CSV file.
* Defaults to 0 (or whatever is set in alert_actions.conf).

action.email.sendpdf = [1|0]
* Specify whether to send results as a PDF file.
* Defaults to 0 (or whatever is set in alert_actions.conf).

action.email.sendresults = [1|0]
* Specify whether to include search results in the
  alert notification email.
* Defaults to 0 (or whatever is set in alert_actions.conf).
```

Settings for script action

```
#******
# Settings for script action
```

```
#******Settings for script action

action.script = 0 | 1
* Enables or disables the script action.
* 1 to enable, 0 to disable.
* Defaults to 0

action.script.filename = <script filename>
* The filename, with no path, of the shell script to execute.
* The script should be located in: $SPLUNK_HOME/bin/scripts/
* For system shell scripts on Unix, or .bat or .cmd on windows, there
  are no further requirements.
* For other types of scripts, the first line should begin with a #!
  marker, followed by a path to the interpreter that will run the
  script.
  * Example: #!C:\Python27\python.exe
* Defaults to empty string.
```

Settings for summary index action

```
#*******
# Settings for summary index action
#*******Settings for summary index action

action.summary_index = 0 | 1
* Enables or disables the summary index action.
* Defaults to 0.

action.summary_index._name = <index>
* Specifies the name of the summary index where the results of the
scheduled
  search are saved.
* Defaults to summary.

action.summary_index.inline = <bool>
* Determines whether to execute the summary indexing action as part of
the
  scheduled search.
* NOTE: This option is considered only if the summary index action is
enabled
  and is always executed (in other words, if counttype = always).
* Defaults to true.

action.summary_index.<field> = <string>
* Specifies a field/value pair to add to every event that gets summary
indexed
  by this search.
* You can define multiple field/value pairs for a single summary index
```

```
search.
```

Settings for lookup table population parameters

```
#*******
# Settings for lookup table population parameters
#*******Settings for lookup table population parameters

action.populate_lookup = 0 | 1
* Enables or disables the lookup population action.
* Defaults to 0.

action.populate_lookup.dest = <string>
* Can be one of the following two options:
  * A lookup name from transforms.conf.
  * A path to a lookup .csv file that Splunk should copy the search
results to,
    relative to $SPLUNK_HOME.
    * NOTE: This path must point to a .csv file in either of the
following
            directories:
      * etc/system/lookups/
      * etc/apps/<app-name>/lookups
      * NOTE: the destination directories of the above files must
already exist
* Defaults to empty string.

run_on_startup = true | false
* Toggles whether this search runs when Splunk starts or any edit that
changes
  search related args happen (which includes: search and dispatch.*
args).
* If set to true the search is ran as soon as possible during startup
or after
  edit otherwise the search is ran at the next scheduled time.
* We recommend that you set run_on_startup to true for scheduled
searches that
  populate lookup tables or generate artifacts used by dashboards.
* Defaults to false.

run_n_times = <unsigned int>
* Runs this search exactly the given number of times, then never again
(until
  Splunk is restarted).
* Defaults to 0 (infinite).
```

dispatch search options

```
#*******
# dispatch search options
#*******dispatch search options

dispatch.ttl = <integer>[p]
* Indicates the time to live (in seconds) for the artifacts of the
scheduled
  search, if no actions are triggered.
* If the integer is followed by the letter 'p' Splunk interprets the ttl
as a
  multiple of the scheduled search's execution period (e.g. if the
search is
  scheduled to run hourly and ttl is set to 2p the ttl of the artifacts
will be
  set to 2 hours).
* If an action is triggered Splunk changes the ttl to that action's
ttl. If
  multiple actions are triggered, Splunk applies the largest action ttl
to the
  artifacts. To set the action's ttl, refer to alert_actions.conf.spec.
* For more info on search's ttl please see limits.conf.spec [search]
ttl
* Defaults to 2p (that is, 2 x the period of the scheduled search).

dispatch.buckets  = <integer>
* The maximum number of timeline buckets.
* Defaults to 0.

dispatch.max_count = <integer>
* The maximum number of results before finalizing the search.
* Defaults to 500000.

dispatch.max_time = <integer>
* Indicates the maximum amount of time (in seconds) before finalizing
the
  search.
* Defaults to 0.

dispatch.lookups = 1| 0
* Enables or disables lookups for this search.
* Defaults to 1.

dispatch.earliest_time = <time-str>
* Specifies the earliest time for this search. Can be a relative or
absolute
  time.
* If this value is an absolute time, use the dispatch.time_format to
format the
```

```
  value.
* Defaults to empty string.

dispatch.latest_time = <time-str>
* Specifies the latest time for this saved search. Can be a relative or
  absolute time.
* If this value is an absolute time, use the dispatch.time_format to
format the
  value.
* Defaults to empty string.

dispatch.index_earliest= <time-str>
* Specifies the earliest index time for this search. Can be a relative
or
  absolute time.
* If this value is an absolute time, use the dispatch.time_format to
format the
  value.
* Defaults to empty string.

dispatch.index_latest= <time-str>
* Specifies the latest index time for this saved search. Can be a
relative or
  absolute time.
* If this value is an absolute time, use the dispatch.time_format to
format the
  value.
* Defaults to empty string.

dispatch.time_format = <time format str>
* Defines the time format that Splunk uses to specify the earliest and
latest
  time.
* Defaults to %FT%T.%Q%:z

dispatch.spawn_process = 1 | 0
* Specifies whether Splunk spawns a new search process when this saved
search
  is executed.
* Default is 1.

dispatch.auto_cancel = <int>
* If specified, the job automatically cancels after this many seconds of
  inactivity. (0 means never auto-cancel)
* Default is 0.

dispatch.auto_pause = <int>
* If specified, the search job pauses after this many seconds of
inactivity. (0
  means never auto-pause.)
* To restart a paused search job, specify unpause as an action to POST
  search/jobs/{search_id}/control.
```

```
* auto_pause only goes into effect once. Unpausing after auto_pause does
not
  put auto_pause into effect again.
* Default is 0.

dispatch.reduce_freq = <int>
* Specifies how frequently Splunk should run the MapReduce reduce phase
on
  accumulated map values.
* Defaults to 10.

dispatch.rt_backfill = <bool>
* Specifies whether to do real-time window backfilling for scheduled
real time
  searches
* Defaults to false.

dispatch.indexedRealtime = <bool>
* Specifies whether to use indexed-realtime mode when doing realtime
searches.
* Default for saved searches is "unset" falling back to limits.conf
setting [realtime] indexed_realtime_use_by_default

dispatch.indexedRealtimeOffset = <int>
* Allows for a per-job override of limits.conf settting [realtime]
indexed_realtime_disk_sync_delay
* Default for saved searches is "unset" falling back to limits.conf
setting.

dispatch.indexedRealtimeMinSpan = <int>
* Allows for a per-job override of limits.conf settting [realtime]
indexed_realtime_default_span
* Default for saved searches is "unset" falling back to limits.conf
setting.

dispatch.rt_maximum_span = <int>
* Allows for a per-job override of limits.conf settting [realtime]
indexed_realtime_maximum_span
* Default for saved searches is "unset" falling back to limits.conf
setting.

dispatch.sample_ratio = <int>
* The integer value used to calculate the sample ratio. The formula is 1
/ <int>.
* The sample ratio specifies the likelihood of any event being included
in the sample.
* For example, if sample_ratio = 500 each event has a 1/500 chance of
being included in the sample result set.
* Defaults to 1.

restart_on_searchpeer_add = 1 | 0
* Specifies whether to restart a real-time search managed by the
```

```
scheduler when
  a search peer becomes available for this saved search.
* NOTE: The peer can be a newly added peer or a peer that has been down
and has
        become available.
* Defaults to 1.
```

auto summarization options

```
#*******
# auto summarization options
#*******auto summarization options
auto_summarize  = <bool>
* Whether the scheduler should ensure that the data for this search is
  automatically summarized
* Defaults to false.

auto_summarize.command = <string>
* A search template to be used to construct the auto summarization for
this
  search.
* DO NOT change unless you know what you're doing

auto_summarize.timespan = <time-specifier> (, <time-specifier>)*
* Comma delimited list of time ranges that each summarized chunk should
span.
  This comprises the list of available granularity levels for which
summaries
  would be available. For example a timechart over the last month whose
  granularity is at the day level should set this to 1d. If you are
going to need
  the same data summarized at the hour level because you need to have
weekly
  charts then use: 1h;1d

auto_summarize.cron_schedule = <cron-string>
* Cron schedule to be used to probe/generate the summaries for this
search

auto_summarize.dispatch.<arg-name> = <string>
* Any dispatch.* options that need to be overridden when running the
summary
  search.

auto_summarize.suspend_period      = <time-specifier>
* Amount of time to suspend summarization of this search if the
summarization
  is deemed unhelpful
```

```
* Defaults to 24h

auto_summarize.max_summary_size     = <unsigned int>
* The minimum summary size when to start testing it's helpfulness
* Defaults to 52428800 (5MB)

auto_summarize.max_summary_ratio    = <positive float>
* The maximum ratio of summary_size/bucket_size when to stop
summarization and
  deem it unhelpful for a bucket
* NOTE: the test is only performed if the summary size is larger
  than auto_summarize.max_summary_size
* Defaults to: 0.1

auto_summarize.max_disabled_buckets = <unsigned int>
* The maximum number of buckets with the suspended summarization before
the
  summarization search is completely stopped and the summarization of
the
  search is suspended for auto_summarize.suspend_period
* Defaults to: 2

auto_summarize.max_time = <unsigned int>
* The maximum amount of time that the summary search is allowed to run.
Note
  that this is an approximate time and the summarize search will be
stopped at
  clean bucket boundaries.
* Defaults to: 3600

auto_summarize.hash = <string>
auto_summarize.normalized_hash = <string>
* These are auto generated settings.

auto_summarize.max_concurrent = <unsigned int>
* The maximum number of concurrent instances of this auto summarizing
search,
  that the scheduler is allowed to run.
* Defaults to: 1
```

alert suppression/severity/expiration/tracking/viewing settings

```
#*******
# alert suppression/severity/expiration/tracking/viewing settings
#*******alert suppression/severity/expiration/tracking/viewing settings

alert.suppress = 0 | 1
* Specifies whether alert suppression is enabled for this scheduled
search.
```

```
* Defaults to 0.

alert.suppress.period = <time-specifier>
* Sets the suppression period. Use [number][time-unit] to specify a
time.
* For example: 60 = 60 seconds, 1m = 1 minute, 1h = 60 minutes = 1 hour
etc
* Honored if and only if alert.suppress = 1
* Defaults to empty string.

alert.suppress.fields = <comma-delimited-field-list>
* List of fields to use when suppressing per-result alerts. This field
*must*
  be specified if the digest mode is disabled and suppression is
enabled.
* Defaults to empty string.

alert.severity = <int>
* Sets the alert severity level.
* Valid values are: 1-debug, 2-info, 3-warn, 4-error, 5-severe, 6-fatal
* Defaults to 3.

alert.expires = <time-specifier>
* Sets the period of time to show the alert in the dashboard. Use
[number][time-unit]
  to specify a time.
* For example: 60 = 60 seconds, 1m = 1 minute, 1h = 60 minutes = 1 hour
etc
* Defaults to 24h.
* This property is valid until splunkd restarts. Restart clears the
listing of
  triggered alerts.

alert.digest_mode = true | false
* Specifies whether Splunk applies the alert actions to the entire
result set
  or on each individual result.
* Defaults to true.

alert.track = true | false | auto
* Specifies whether to track the actions triggered by this scheduled
search.
* auto  - determine whether to track or not based on the tracking
setting of
  each action, do not track scheduled searches that always trigger
actions.
* true  - force alert tracking.
* false - disable alert tracking for this search.
* Defaults to auto.

alert.display_view = <string>
* Name of the UI view where the emailed link for per result alerts
```

```
should point to.
* If not specified, the value of request.ui_dispatch_app will be used,
if that
  is missing then "search" will be used
* Defaults to empty string
```

UI-specific settings

```
#*******
# UI-specific settings
#*******UI-specific settings

displayview =<string>
* Defines the default UI view name (not label) in which to load the
results.
* Accessibility is subject to the user having sufficient permissions.
* Defaults to empty string.

vsid = <string>
* Defines the viewstate id associated with the UI view listed in
'displayview'.
* Must match up to a stanza in viewstates.conf.
* Defaults to empty string.

is_visible = true | false
* Specifies whether this saved search should be listed in the visible
saved
  search list.
* Defaults to true.

description = <string>
* Human-readable description of this saved search.
* Defaults to empty string.

request.ui_dispatch_app  = <string>
* Specifies a field used by Splunk UI to denote the app this search
should be
  dispatched in.
* Defaults to empty string.

request.ui_dispatch_view = <string>
* Specifies a field used by Splunk UI to denote the view this search
should be
  displayed in.
* Defaults to empty string.
```

Display Formatting Options

```
#******
# Display Formatting Options
#******Display Formatting Options

# General options
display.general.enablePreview = 0 | 1
display.general.type = [events|statistics|visualizations]
display.general.timeRangePicker.show = 0 | 1
display.general.migratedFromViewState = 0 | 1
display.general.locale = <string>

# Event options
display.events.fields = [<string>(, <string>)*]
display.events.type = [raw|list|table]
display.events.rowNumbers = 0 | 1
display.events.maxLines = <int>
display.events.raw.drilldown = [inner|outer|full|none]
display.events.list.drilldown = [inner|outer|full|none]
display.events.list.wrap = 0 | 1
display.events.table.drilldown = 0 | 1
display.events.table.wrap = 0 | 1

# Statistics options
display.statistics.rowNumbers = 0 | 1
display.statistics.wrap = 0 | 1
display.statistics.overlay = [none|heatmap|highlow]
display.statistics.drilldown = [row|cell|none]
display.statistics.totalsRow = 0 | 1
display.statistics.percentagesRow = 0 | 1
display.statistics.show = 0 | 1

# Visualization options
display.visualizations.show = 0 | 1
display.visualizations.type = [charting|singlevalue|mapping|custom]
display.visualizations.chartHeight = <int>
display.visualizations.charting.chart =
[line|area|column|bar|pie|scatter|bubble|radialGauge|fillerGauge|markerGauge]
display.visualizations.charting.chart.stackMode =
[default|stacked|stacked100]
display.visualizations.charting.chart.nullValueMode =
[gaps|zero|connect]
display.visualizations.charting.chart.overlayFields = <string>
display.visualizations.charting.drilldown = [all|none]
display.visualizations.charting.chart.style = [minimal|shiny]
display.visualizations.charting.layout.splitSeries = 0 | 1
display.visualizations.charting.layout.splitSeries.allowIndependentYRanges
= 0 | 1
display.visualizations.charting.legend.placement =
```

```
[right|bottom|top|left|none]
display.visualizations.charting.legend.labelStyle.overflowMode =
[ellipsisEnd|ellipsisMiddle|ellipsisStart]
display.visualizations.charting.axisTitleX.text = <string>
display.visualizations.charting.axisTitleY.text = <string>
display.visualizations.charting.axisTitleY2.text = <string>
display.visualizations.charting.axisTitleX.visibility =
[visible|collapsed]
display.visualizations.charting.axisTitleY.visibility =
[visible|collapsed]
display.visualizations.charting.axisTitleY2.visibility =
[visible|collapsed]
display.visualizations.charting.axisX.scale = linear|log
display.visualizations.charting.axisY.scale = linear|log
display.visualizations.charting.axisY2.scale = linear|log|inherit
display.visualizations.charting.axisLabelsX.majorLabelStyle.overflowMode
= [ellipsisMiddle|ellipsisNone]
display.visualizations.charting.axisLabelsX.majorLabelStyle.rotation =
[-90|-45|0|45|90]
display.visualizations.charting.axisLabelsX.majorUnit = <float> | auto
display.visualizations.charting.axisLabelsY.majorUnit = <float> | auto
display.visualizations.charting.axisLabelsY2.majorUnit = <float> | auto
display.visualizations.charting.axisX.minimumNumber = <float> | auto
display.visualizations.charting.axisY.minimumNumber = <float> | auto
display.visualizations.charting.axisY2.minimumNumber = <float> | auto
display.visualizations.charting.axisX.maximumNumber = <float> | auto
display.visualizations.charting.axisY.maximumNumber = <float> | auto
display.visualizations.charting.axisY2.maximumNumber = <float> | auto
display.visualizations.charting.axisY2.enabled = 0 | 1
display.visualizations.charting.chart.sliceCollapsingThreshold =
<float>
display.visualizations.charting.chart.showDataLabels =
[all|none|minmax]
display.visualizations.charting.gaugeColors = [<hex>(, <hex>)*]
display.visualizations.charting.chart.rangeValues = [<string>(,
<string>)*]
display.visualizations.charting.chart.bubbleMaximumSize = <int>
display.visualizations.charting.chart.bubbleMinimumSize = <int>
display.visualizations.charting.chart.bubbleSizeBy = [area|diameter]
display.visualizations.custom.type = <string>
display.visualizations.custom.height = <int>
display.visualizations.singlevalueHeight = <int>
display.visualizations.singlevalue.beforeLabel = <string>
display.visualizations.singlevalue.afterLabel = <string>
display.visualizations.singlevalue.underLabel = <string>
display.visualizations.singlevalue.unit = <string>
display.visualizations.singlevalue.unitPosition = [before|after]
display.visualizations.singlevalue.drilldown = [all|none]
display.visualizations.singlevalue.colorMode = [block|none]
display.visualizations.singlevalue.rangeValues = [<string>(,
<string>)*]
display.visualizations.singlevalue.rangeColors = [<string>(,
```

```
<string>)*]
display.visualizations.singlevalue.trendInterval = <string>
display.visualizations.singlevalue.trendColorInterpretation =
[standard|inverse]
display.visualizations.singlevalue.showTrendIndicator = 0 | 1
display.visualizations.singlevalue.showSparkline = 0 | 1
display.visualizations.singlevalue.trendDisplayMode =
[percent|absolute]
display.visualizations.singlevalue.colorBy = [value|trend]
display.visualizations.singlevalue.useColors = 0 | 1
display.visualizations.singlevalue.numberPrecision =
[0|0.0|0.00|0.000|0.0000]
display.visualizations.singlevalue.useThousandSeparators = 0 | 1
display.visualizations.mapHeight = <int>
display.visualizations.mapping.type = [marker|choropleth]
display.visualizations.mapping.drilldown = [all|none]
display.visualizations.mapping.map.center = (<float>,<float>)
display.visualizations.mapping.map.zoom = <int>
display.visualizations.mapping.map.scrollZoom = 0 | 1
display.visualizations.mapping.map.panning    = 0 | 1
display.visualizations.mapping.choroplethLayer.colorMode =
[auto|sequential|divergent|categorical]
display.visualizations.mapping.choroplethLayer.maximumColor = <string>
display.visualizations.mapping.choroplethLayer.minimumColor = <string>
display.visualizations.mapping.choroplethLayer.colorBins = <int>
display.visualizations.mapping.choroplethLayer.neutralPoint = <float>
display.visualizations.mapping.choroplethLayer.shapeOpacity = <float>
display.visualizations.mapping.choroplethLayer.showBorder = 0 | 1
display.visualizations.mapping.markerLayer.markerOpacity = <float>
display.visualizations.mapping.markerLayer.markerMinSize = <int>
display.visualizations.mapping.markerLayer.markerMaxSize = <int>
display.visualizations.mapping.data.maxClusters = <int>
display.visualizations.mapping.showTiles = 0 | 1
display.visualizations.mapping.tileLayer.tileOpacity = <float>
display.visualizations.mapping.tileLayer.url = <string>
display.visualizations.mapping.tileLayer.minZoom = <int>
display.visualizations.mapping.tileLayer.maxZoom = <int>

# Patterns options
display.page.search.patterns.sensitivity = <float>

# Page options
display.page.search.mode = [fast|smart|verbose]
display.page.search.timeline.format = [hidden|compact|full]
display.page.search.timeline.scale = [linear|log]
display.page.search.showFields = 0 | 1
display.page.search.tab = [events|statistics|visualizations|patterns]
# Deprecated
display.page.pivot.dataModel = <string>
```

Table format settings

```
#*******
# Table format settings
#*******Table format settings

# Format options
display.statistics.format.<index> = [color|number]
display.statistics.format.<index>.field = <string>
display.statistics.format.<index>.fields = [<string>(, <string>)*]

# Color format options
display.statistics.format.<index>.scale =
[category|linear|log|minMidMax|sharedCategory|threshold]
display.statistics.format.<index>.colorPalette =
[expression|list|map|minMidMax|sharedList]

# Number format options
display.statistics.format.<index>.precision = <int>
display.statistics.format.<index>.useThousandSeparators = <bool>
display.statistics.format.<index>.unit = <string>
display.statistics.format.<index>.unitPosition = [before|after]

# Scale options for 'category'
display.statistics.format.<index>.scale.categories = [<string>(,
<string>)*]

# Scale options for 'log'
display.statistics.format.<index>.scale.base = <int>

# Scale options for 'minMidMax'
display.statistics.format.<index>.scale.minType =
[number|percent|percentile]
display.statistics.format.<index>.scale.minValue = <float>
display.statistics.format.<index>.scale.midType =
[number|percent|percentile]
display.statistics.format.<index>.scale.midValue = <float>
display.statistics.format.<index>.scale.maxType =
[number|percent|percentile]
display.statistics.format.<index>.scale.maxValue = <float>

# Scale options for 'threshold'
display.statistics.format.<index>.scale.thresholds = [<float>(,
<float>)*]

# Color palette options for 'expression'
display.statistics.format.<index>.colorPalette.rule = <string>

# Color palette options for 'list'
display.statistics.format.<index>.colorPalette.colors = [<hex>(,
```

```
<hex>)*]
display.statistics.format.<index>.colorPalette.interpolate = <bool>

# Color palette options for 'map'
display.statistics.format.<index>.colorPalette.colors =
{<string>:<hex>(, <string>:<hex>)*}

# Color palette options for 'minMidMax'
display.statistics.format.<index>.colorPalette.minColor = <hex>
display.statistics.format.<index>.colorPalette.midColor = <hex>
display.statistics.format.<index>.colorPalette.maxColor = <hex>
```

Other settings

```
#*******
# Other settings
#*******Other settings

embed.enabled = 0 | 1
* Specifies whether a saved search is shared for access with a
guestpass.
* Search artifacts of a search can be viewed via a guestpass only if:
  * A token has been generated that is associated with this saved
search.
    The token is associated with a particular user and app context.
  * The user to whom the token belongs has permissions to view that
search.
  * The saved search has been scheduled and there are artifacts
available.
    Only artifacts are available via guestpass: we never dispatch a
search.
  * The save search is not disabled, it is scheduled, it is not
real-time,
    and it is not an alert.
```

deprecated settings

```
#*******
# deprecated settings
#*******deprecated settings

sendresults = <bool>
* use action.email.sendresult

action_rss = <bool>
* use action.rss
```

```
action_email = <string>
* use action.email and action.email.to

role = <string>
* see saved search permissions

userid = <string>
* see saved search permissions

query  = <string>
* use search

nextrun  = <int>
* not used anymore, the scheduler maintains this info internally

qualifiedSearch = <string>
* not used anymore, the Splunk software computes this value during
runtime
```

savedsearches.conf.example

```
#   Version 6.5.0
#
# This file contains example saved searches and alerts.
#
# To use one or more of these configurations, copy the configuration
block into
# savedsearches.conf in $SPLUNK_HOME/etc/system/local/. You must
restart Splunk
# to enable configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles


# The following searches are example searches.  To create your own
search,
# modify the values by following the spec outlined in
savedsearches.conf.spec.


[Daily indexing volume by server]
search = index=_internal todaysBytesIndexed LicenseManager-Audit NOT
source=*web_service.log NOT source=*web_access.log | eval Daily
_Indexing_Volume_in_MBs = todaysBytesIndexed/1024/1024 | timechart
```

```
avg(Daily_Indexing_Volume_in_MBs) by host
dispatch.earliest_time = -7d

[Errors in the last 24 hours]
search = error OR failed OR severe OR ( sourcetype=access_* ( 404 OR
500 OR 503 ) )
dispatch.earliest_time = -1d

[Errors in the last hour]
search = error OR failed OR severe OR ( sourcetype=access_* ( 404 OR
500 OR 503 ) )
dispatch.earliest_time = -1h

[KB indexed per hour last 24 hours]
search = 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
dispatch.earliest_time = -1d

[Messages by minute last 3 hours]
search = index=_internal eps "group=per_source_thruput" NOT filetracker
| eval events=eps*kb/kbps | timechart fixedrange=t span=1m s
um(events) by series
dispatch.earliest_time = -3h

[Splunk errors last 24 hours]
search = index=_internal " error " NOT debug source=*/splunkd.log*
dispatch.earliest_time = -24h
```

searchbnf.conf

The following are the spec and example files for searchbnf.conf.

searchbnf.conf.spec

```
#   Version 6.5.0
#
#
# This file contain descriptions of stanzas and attribute/value pairs
for
# configuring search-assistant via searchbnf.conf
#
# There is a searchbnf.conf in $SPLUNK_HOME/etc/system/default/.  It
should
# not be modified.  If your application has its own custom python
```

```
search
# commands, your application can include its own searchbnf.conf to
describe
# the commands to the search-assistant.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
top
#     of the file.
#   * Each conf file should have at most one default stanza. If there
are
#     multiple default stanzas, attributes are combined. In the case of
#     multiple definitions of the same attribute, the last definition in
the
#     file wins.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.
```

[<search-commandname>-command]

```
[<search-commandname>-command]
* This stanza enables properties for a given <search-command>.
* A searchbnf.conf file can contain multiple stanzas for any number of
  commands.  * Follow this stanza name with any number of the following
  attribute/value pairs.
* If you do not set an attribute for a given <spec>, the default is
used.
  The default values are empty.
* An example stanza name might be "geocode-command", for a "geocode"
  command.
* Search command stanzas can refer to definitions defined in others
stanzas,
  and they do not require "-command", appended to them.  For example:
```

[geocode-command]

```
[geocode-command]
  syntax = geocode <geocode-option>*
  ...
```

[geocode-option]

```
[geocode-option]
  syntax = (maxcount=<int>) | (maxhops=<int>)
  ...


#*******************************************************************************
# The possible attributes/value pairs for searchbnf.conf
#*******************************************************************************


SYNTAX = <string>
* Describes the syntax of the search command.  See the head of
  searchbnf.conf for details.
* Required

SIMPLESYNTAX = <string>

* Optional simpler version of the syntax to make it easier to
  understand at the expense of completeness.  Typically it removes
  rarely used options or alternate ways of saying the same thing.
* For example, a search command might accept values such as
  "m|min|mins|minute|minutes", but that would unnecessarily
  clutter the syntax description for the user.  In this can, the
  simplesyntax can just pick the one (e.g., "minute").

ALIAS = <commands list>
* Alternative names for the search command.  This further cleans
  up the syntax so the user does not have to know that
  'savedsearch' can also be called by 'macro' or 'savedsplunk'.

DESCRIPTION = <string>
* Detailed text description of search command.  Description can continue
on
  the next line if the line ends in "\"
* Required

SHORTDESC = <string>
* A short description of the search command.  The full DESCRIPTION
  may take up too much screen real-estate for the search assistant.
* Required
```

```
EXAMPLE = <string>
COMMENT = <string>
* 'example' should list out a helpful example of using the search
  command, and 'comment' should describe that example.
* 'example' and 'comment' can be appended with matching indexes to
  allow multiple examples and corresponding comments.
* For example:
    example2 = geocode maxcount=4
    command2 = run geocode on up to four values
    example3 = geocode maxcount=-1
    comment3 = run geocode on all values

USAGE = public|private|deprecated
* Determines if a command is public, private, depreciated.  The
  search assistant only operates on public commands.
* Required

TAGS = <tags list>
* List of tags that describe this search command.  Used to find
  commands when the use enters a synonym (e.g. "graph" -> "chart")

RELATED = <commands list>
* List of related commands to help user when using one command to
  learn about others.


#*******************************************************************************
# Optional attributes primarily used internally at Splunk
#*******************************************************************************

maintainer, appears-in, note, supports-multivalue, optout-in
```

searchbnf.conf.example

```
#   Version 6.5.0
#
# The following are example stanzas for searchbnf.conf configurations.
#

##################
# selfjoin
##################
[selfjoin-command]
syntax = selfjoin (<selfjoin-options>)* <field-list>
shortdesc = Join results with itself.
description = Join results with itself.  Must specify at least one field
to join on.
```

```
usage = public
example1 = selfjoin id
comment1 = Joins results with itself on 'id' field.
related = join
tags = join combine unite

[selfjoin-options]
syntax = overwrite=<bool> | max=<int> | keepsingle=<int>
description = The selfjoin joins each result with other results that\
  have the same value for the join fields.  'overwrite' controls if\
  fields from these 'other' results should overwrite fields of the\
  result used as the basis for the join (default=true).  max indicates\
  the maximum number of 'other' results each main result can join with.\
  (default = 1, 0 means no limit).  'keepsingle' controls whether or
not\
  results with a unique value for the join fields (and thus no other\
  results to join with) should be retained.  (default = false)
```

segmenters.conf

The following are the spec and example files for segmenters.conf.

segmenters.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attribute/value pairs for configuring
# segmentation of events in segementers.conf.
#
# There is a default segmenters.conf in
$SPLUNK_HOME/etc/system/default. To set
# custom configurations, place a segmenters.conf in
# $SPLUNK_HOME/etc/system/local/.  For examples, see
segmenters.conf.example.
# You must restart Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#     * You can also define global settings outside of any stanza, at
the top of the file.
#     * Each conf file should have at most one default stanza. If there
are multiple default
#       stanzas, attributes are combined. In the case of multiple
definitions of the same
#       attribute, the last definition in the file wins.
#     * If an attribute is defined at both the global level and in a
specific stanza, the
#       value in the specific stanza takes precedence.
```

[<SegmenterName>]

```
[<SegmenterName>]
* Name your stanza.
* Follow this stanza name with any number of the following
attribute/value
  pairs.
* If you don't specify an attribute/value pair, Splunk will use the
default.

MAJOR = <space separated list of breaking characters>
* Set major breakers.
* Major breakers are words, phrases or terms in your data that are
surrounded
  by set breaking characters.
* By default, major breakers are set to most characters and blank
spaces.
* Typically, major breakers are single characters.
* Please note: \s represents a space; \n, a newline; \r, a carriage
return; and
  \t, a tab.
* Default is [ ] < > ( ) { } | ! ; , ' " * \n \r \s \t & ?
+ %21 %26 %2526 %3B %7C %20 %2B %3D
-- %2520 %5D %5B %3A %0A %2C %28 %29


MINOR = <space separated list of strings>
* Set minor breakers.
* In addition to the segments specified by the major breakers, for each
minor
  breaker found, Splunk indexes the token from the last major breaker to
the
  current minor breaker and from the last minor breaker to the current
minor
```

```
  breaker.
* Default is / : = @ . - $ # % \\ _

INTERMEDIATE_MAJORS =  true | false
* Set this to "true" if you want an IP address to appear in typeahead as
  a, a.b, a.b.c, a.b.c.d
* The typical performance hit by setting to "true" is 30%.
* Default is "false".

FILTER = <regular expression>
* If set, segmentation will only take place if the regular expression
matches.
* Furthermore, segmentation will only take place on the first group of
the
  matching regex.
* Default is empty.

LOOKAHEAD = <integer>
* Set how far into a given event (in characters) Splunk segments.
* LOOKAHEAD applied after any FILTER rules.
* To disable segmentation, set to 0.
* Defaults to -1 (read the whole event).

MINOR_LEN = <integer>
* Specify how long a minor token can be.
* Longer minor tokens are discarded without prejudice.
* Defaults to -1.

MAJOR_LEN = <integer>
* Specify how long a major token can be.
* Longer major tokens are discarded without prejudice.
* Defaults to -1.

MINOR_COUNT = <integer>
* Specify how many minor segments to create per event.
* After the specified number of minor tokens have been created, later
ones are
  discarded without prejudice.
* Defaults to -1.

MAJOR_COUNT = <integer>
* Specify how many major segments are created per event.
* After the specified number of major segments have been created, later
ones
  are discarded without prejudice.
* Default to -1.
```

segmenters.conf.example

```
#   Version 6.5.0
#
# The following are examples of segmentation configurations.
#
# To use one or more of these configurations, copy the configuration
block into
# segmenters.conf in $SPLUNK_HOME/etc/system/local/. You must restart
Splunk to
# enable configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles


# Example of a segmenter that doesn't index the date as segments in
syslog
# data:

[syslog]
FILTER = ^.*?\d\d:\d\d:\d\d\s+\S+\s+(.*)$


# Example of a segmenter that only indexes the first 256b of events:

[limited-reach]
LOOKAHEAD = 256


# Example of a segmenter that only indexes the first line of an event:

[first-line]
FILTER = ^(.*?)(\n|$)


# Turn segmentation off completely:

[no-segmentation]
LOOKAHEAD = 0
```

server.conf

The following are the spec and example files for server.conf.

server.conf.spec

```
#   Version 6.5.0
#
# This file contains the set of attributes and values you can use to
# configure server options in server.conf.
#
# There is a server.conf in $SPLUNK_HOME/etc/system/default/.  To set
custom
# configurations, place a server.conf in
$SPLUNK_HOME/etc/system/local/.
# For examples, see server.conf.example.  You must restart Splunk to
enable
# configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
top
#     of the file.
#   * Each conf file should have at most one default stanza. If there
are
#     multiple default stanzas, attributes are combined. In the case of
#     multiple definitions of the same attribute, the last definition in
the
#     file wins.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.
```

General Server Configuration

```
############################################################################
# General Server Configuration
############################################################################General
Server Configuration
[general]
serverName = <ASCII string>
* The name used to identify this Splunk instance for features such as
  distributed search.
* Defaults to <hostname>-<user running splunk>.
* Shall not be an empty string
* May contain environment variables
* After any environment variables have been expanded, the server name
  (if not an IPv6 address) can only contain letters, numbers,
underscores,
  dots, and dashes; and it must start with a letter, number, or an
  underscore.

hostnameOption = <ASCII string>
* The option used to specify the detail in the server name used to
identify
  this Splunk instance.
* Can be one of "fullyqualifiedname" , "clustername", "shortname"
* Is applicable to Windows only
* Shall not be an empty string

sessionTimeout = <nonnegative integer>[smhd]
* The amount of time before a user session times out, expressed as a
  search-like time range
* Examples include '24h' (24 hours), '3d' (3 days),
  '7200s' (7200 seconds, or two hours)
* Defaults to '1h' (1 hour)

trustedIP = <IP address>
* All logins from this IP address are trusted, meaning password is no
longer
  required
* Only set this if you are using Single Sign On (SSO)

allowRemoteLogin = always|never|requireSetPassword
* Controls remote management by restricting general login. Note that
this
  does not apply to trusted SSO logins from trustedIP.
* If 'always', enables authentication so that all remote login attempts
are
  allowed.
* If 'never', only local logins to splunkd will be allowed. Note that
this
  will still allow remote management through splunkweb if splunkweb is
```

```
on
  the same server.
* If 'requireSetPassword' (default):
  * In the free license, remote login is disabled.
  * In the pro license, remote login is only disabled for "admin" user
if
    default password of "admin" has not been changed.

access_logging_for_phonehome = true|false
* Enables/disables logging to splunkd_access.log for client phonehomes
* defaults to true (logging enabled)

hangup_after_phonehome = true|false
* Controls whether or not the (deployment) server hangs up the
connection
  after the phonehome is done.
* By default we use persistent HTTP 1.1 connections with the server to
  handle phonehomes. This may show higher memory usage for a large
number of
  clients.
* In case we have more than maximum concurrent tcp connection number of
  deployment clients, persistent connections do not help with the reuse
of
  connections anyway, so setting this to false helps bring down memory
* usage.
* defaults to false (persistent connections for phonehome)

pass4SymmKey = <password>
* Authenticates traffic between:
  * License master and its license slaves.
  * Members of a cluster; see Note 1 below.
  * Deployment server (DS) and its deployment clients (DCs); see Note 2
    below.
* Note 1: Clustering may override the passphrase specified here, in
  the [clustering] stanza.  A clustering searchhead connecting to
multiple
  masters may further override in the [clustermaster:stanza1] stanza.
* Note 2: By default, DS-DCs passphrase auth is disabled.  To enable
DS-DCs
  passphrase auth, you must *also* add the following line to the
  [broker:broker] stanza in restmap.conf:
     requireAuthentication = true
* In all scenarios, *every* node involved must set the same passphrase
in
  the same stanza(s) (i.e. [general] and/or [clustering]); otherwise,
  respective communication (licensing and deployment in case of
[general]
  stanza, clustering in case of [clustering] stanza) will not proceed.

listenOnIPv6 = no|yes|only
* By default, splunkd will listen for incoming connections (both REST
and
```

```
  TCP inputs) using IPv4 only
* To enable IPv6 support in splunkd, set this to 'yes'.  splunkd will
  simultaneously listen for connections on both IPv4 and IPv6
* To disable IPv4 entirely, set this to 'only', which will cause splunkd
  to exclusively accept connections over IPv6.  You will probably also
  need to change mgmtHostPort in web.conf (use '[::1]' instead of
'127.0.0.1')
* Note that any setting of SPLUNK_BINDIP in your environment or
  splunk-launch.conf will override this value.  In that case splunkd
will
  listen on the exact address specified.

connectUsingIpVersion = auto|4-first|6-first|4-only|6-only
* When making outbound TCP connections (for forwarding eventdata, making
  distributed search requests, etc) this controls whether the
connections
  will be made via IPv4 or IPv6.
* If a host is available over both IPv4 and IPv6 and this is set to
  '4-first', then we will connect over IPv4 first and fallback to IPv6
if
  the connection fails.
* If it is set to '6-first' then splunkd will try IPv6 first and
fallback to
  IPv4 on failure
* If this is set to '4-only' then splunkd will only attempt to make
  connections over IPv4.
* Likewise, if this is set to '6-only', then splunkd will only attempt
to
  connect to the IPv6 address.
* The default value of 'auto' will select a reasonable value based on
  listenOnIPv6 setting.  If that value is set to 'no' it will act like
  '4-only'.  If it is set to 'yes' it will act like '6-first' and if it
is
  set to 'only' it will act like '6-only'.
* Note that connections to literal addresses are unaffected by this.
 For
  example, if a forwarder is configured to connect to "10.1.2.3" the
  connection will be made over IPv4 regardless of this setting.

guid = <globally unique identifier for this instance>
* This setting now (as of 5.0) belongs in the [general] stanza of
  SPLUNK_HOME/etc/instance.cfg file; please see specfile of
instance.cfg for
  more information.

useHTTPServerCompression = <bool>
* Whether splunkd HTTP server should support gzip content encoding. For
more
  info on how content encoding works, see
  http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html (section
14.3).
* Defaults to true.
```

```
defaultHTTPServerCompressionLevel = <integer>
* If useHTTPServerCompression is enabled, this setting constrols the
  compression "level" we attempt
* This number must be in the range 1 through 9
* Higher numbers produce smaller compressed results but require more CPU
  usage
* The default value of 6 is appropriate for most environments

skipHTTPCompressionAcl = <network_acl>
* Lists a set of networks or addresses to skip compressing data for.
  These are addresses that are considered so close that network speed
is
  never an issue, so any CPU time spent compressing a response is
wasteful.
* Note that the server may still respond with compressed data if it
  already has a compressed version of the data available.
* These rules are separated by commas or spaces
* Each rule can be in the following forms:
    1. A single IPv4 or IPv6 address (examples: "10.1.2.3", "fe80::4a3")
    2. A CIDR block of addresses (examples: "10/8", "fe80:1234/32")
    3. A DNS name, possibly with a '*' used as a wildcard (examples:
       "myhost.example.com", "*.splunk.com")
    4. A single '*' which matches anything
* Entries can also be prefixed with '!' to negate their meaning.
* Defaults to localhost addresses.

site = <site-id>
* Specifies the site that this splunk instance belongs to when multisite
is
  enabled.
* Valid values for site-id include site1 to site63

useHTTPClientCompression = true|false|on-http|on-https
* Whether gzip compression should be supported when Splunkd acts as a
client
  (including distributed searches). Note that  in order for the content
to
  be compressed, the HTTP server that the client is connecting to should
  also support compression.
* If the connection is being made over https and
  useClientSSLCompression=true (see below), then setting this option to
true
  would result in double compression work without much compression gain.
It
  is recommended that this value be set to on-http (or to true, and
  useClientSSLCompression to false).
* Defaults to false.

embedSecret = <string>
* When using report embedding, normally the generated URLs can only
  be used on the search head they were generated on
```

```
* If "embedSecret" is set, then the token in the URL will be encrypted
  with this key.  Then other search heads with the exact same setting
  can also use the same URL.
* This is needed if you want to use report embedding across multiple
  nodes on a search head pool.

parallelIngestionPipelines = <integer>
* Data being loaded into splunk, whether for indexing or forwarding,
  progresses through a series of steps arranged into "pipelines".
  By setting this to more than one, more processor threads can be set up
  to perform this work.
* Defaults to 1.
* NOTE: Be careful when changing this.  By increasing the CPU used by
  data ingestion, less is available for other tasks such as searching.
  For most installs the default setting is optimal.
* NOTE: Please note that enabling multiple ingestion pipelines could
  change the behaviour of some of the settings in limits.conf file.
  Each ingestion pipeline will enforce these limits independently.
    1. maxKBps
    2. max_fd
    3. maxHotBuckets
    4. maxHotSpanSecs

instanceType = <string>
* Should not be modified by users.
* Informs components (such as the SplunkWeb Manager section) which
  environment Splunk is running in, to allow for more customized
behaviors.
* Defaults to "download", meaning no special behaviors.

requireBootPassphrase = <bool>
* Prompt the user for a boot passphrase when starting Splunk.
* Splunk uses this passphrase to grant itself access to
platform-provided
  secret storage facilities, like the GNOME keyring.
* For more information about secret storage, see the [secrets] stanza in
  $SPLUNK_HOME/etc/system/README/authentication.conf.spec.
* Defaults to true if Common Criteria mode is enabled.
* Defaults to false if Common Criteria mode is disabled.
* NOTE: Splunk plans to submit Splunk Enterprise for Common Criteria
  evaluation. Splunk does not support using the product in Common
  Criteria mode until it has been certified by NIAP. See the "Securing
  Splunk Enterprise" manual for information on the status of Common
  Criteria certification.
```

Deployment Configuration details

```
############################################################################
# Deployment Configuration details
```

```
################################################################################Deployment
Configuration details

[deployment]
pass4SymmKey = <password>
    * Authenticates traffic between Deployment server (DS) and its
deployment
      clients (DCs).
    * By default, DS-DCs passphrase auth is disabled. To enable DS-DCs
      passphrase auth, you must *also* add the following line to the
      [broker:broker] stanza in restmap.conf:
          requireAuthentication = true
    * If it is not set in the deployment stanza, the key will be looked
in
      the general stanza
```

SSL Configuration details

```
################################################################################
# SSL Configuration details
################################################################################SSL
Configuration details

[sslConfig]
* Set SSL for communications on Splunk back-end under this stanza name.
  * NOTE: To set SSL (eg HTTPS) for Splunk Web and the browser, use
          web.conf.
* Follow this stanza name with any number of the following
attribute/value
  pairs.
* If you do not specify an entry for each attribute, Splunk will use the
  default value.

enableSplunkdSSL = <bool>
* Enables/disables SSL on the splunkd management port (8089) and KV
store
  port (8191).
* Defaults to true.
* Note: Running splunkd without SSL is not generally recommended.
* Distributed search will often perform better with SSL enabled.

useClientSSLCompression = <bool>
* Turns on HTTP client compression.
* Server-side compression is turned on by default; setting this on the
  client side enables compression between server and client.
* Enabling this potentially gives you much faster distributed searches
  across multiple Splunk instances.
* Defaults to true.
```

```
useSplunkdClientSSLCompression = <bool>
* Controls whether SSL compression would be used when splunkd is acting
as
  an HTTP client, usually during certificate exchange, bundle
replication,
  remote calls etc.
* NOTE: this setting is effective if, and only if,
useClientSSLCompression
        is set to true
* NOTE: splunkd is not involved in data transfer in distributed search,
the
        search in a separate process is.
* Defaults to true.

sslVersions = <versions_list>
* Comma-separated list of SSL versions to support for incoming
connections.
* The versions available are "ssl3", "tls1.0", "tls1.1", and "tls1.2".
* The special version "*" selects all supported versions.  The version
"tls"
  selects all versions tls1.0 or newer.
* If a version is prefixed with "-" it is removed from the list.
* SSLv2 is always disabled; "-ssl2" is accepted in the version list but
does nothing.
* When configured in FIPS mode, ssl3 is always disabled regardless
  of this configuration.
* Defaults to "*,-ssl2" (anything newer than SSLv2).

sslVersionsForClient = <versions_list>
* Comma-separated list of SSL versions to support for outgoing HTTP
connections
  from splunkd.  This includes distributed search, deployment client,
etc.
* This is usually less critical, since SSL/TLS will always pick the
highest
  version both sides support.  However, this can be used to prohibit
making
  connections to remote servers that only support older protocols.
* The syntax is the same as the sslVersions setting above
* Note that for forwarder connections, there is a separate
"sslVersions"
  setting in outputs.conf.  For connections to SAML servers, there is a
  separate "sslVersions" setting in authentication.conf.
* Defaults to "*,-ssl2" (anything newer than SSLv2).

supportSSLV3Only = <bool>
* DEPRECATED.  SSLv2 is now always disabled.  The exact set of SSL
versions
  allowed is now configurable via the "sslVersions" setting above.

sslVerifyServerCert = <bool>
* Used by distributed search: when making a search request to another
```

```
  server in the search cluster.
* Used by distributed deployment clients: when polling a deployment
  server.
* If this is set to true, you should make sure that the server that is
  being connected to is a valid one (authenticated).  Both the common
  name and the alternate name of the server are then checked for a
  match if they are specified in this configuration file.  A
  certificiate is considered verified if either is matched.
* Default is false.

sslCommonNameToCheck = <commonName1>, <commonName2>, ...
* If this value is set, and 'sslVerifyServerCert' is set to true,
  splunkd will limit most outbound HTTPS connections to hosts which use
  a cert with one of the listed common names.
* The most important scenario is distributed search.
* This feature does not work with the deployment server and client
  communication over SSL.
* Optional.  Defaults to no common name checking.

sslCommonNameList = <commonName1>, <commonName2>, ...
* DEPRECATED; use 'sslCommonNameToCheck' instead.

sslAltNameToCheck = <alternateName1>, <alternateName2>, ...
* If this value is set, and 'sslVerifyServerCert' is set to true,
  splunkd will also be willing to verify certificates which have a
  so-called "Subject Alternate Name" that matches any of the alternate
  names in this list.
  * Subject Alternate Names are effectively extended descriptive
    fields in SSL certs beyond the commonName.  A common practice for
    HTTPS certs is to use these values to store additional valid
    hostnames or domains where the cert should be considered valid.
* Accepts a comma-separated list of Subject Alternate Names to consider
  valid.
* Items in this list are never validated against the SSL Common Name.
* This feature does not work with the deployment server and client
  communication over SSL.
* Optional.  Defaults to no alternate name checking

requireClientCert = <bool>
* Requires that any HTTPS client that connects to splunkd internal HTTPS
  server has a certificate that was signed by a CA (Certificate
Authority)
  specified by 'sslRootCAPath'.
* Used by distributed search: Splunk indexing instances must be
  authenticated to connect to another splunk indexing instance.
* Used by distributed deployment: the deployment server requires that
  deployment clients are authenticated before allowing them to poll for
new
  configurations/applications.
* If true, a client can connect ONLY if a certificate created by our
  certificate authority was used on that client.
* Default is false.
```

```
cipherSuite = <cipher suite string>
* If set, Splunk uses the specified cipher string for the HTTP server.
* If not set, Splunk uses the default cipher string provided by
OpenSSL.
  This is used to ensure that the server does not accept connections
using
  weak encryption protocols.
* Must specify 'dhFile' to enable any Diffie-Hellman ciphers.

ecdhCurveName = <string>
* DEPRECATED; use 'ecdhCurves' instead.
* ECDH curve to use for ECDH key negotiation
* We only support named curves specified by their SHORT name.
* The list of valid named curves by their short/long names can be
obtained
  by executing this command:
  $SPLUNK_HOME/bin/splunk cmd openssl ecparam -list_curves
* Default is empty string.

ecdhCurves = <comma separated list of ec curves>
* ECDH curves to use for ECDH key negotiation.
* The curves should be specified in the order of preference.
* The client sends these curves as a part of Client Hello.
* The server supports only the curves specified in the list.
* We only support named curves specified by their SHORT names.
  (see struct ASN1_OBJECT in asn1.h)
* The list of valid named curves by their short/long names can be
obtained
  by executing this command:
  $SPLUNK_HOME/bin/splunk cmd openssl ecparam -list_curves
* Default is empty string.
* e.g. ecdhCurves = prime256v1,secp384r1,secp521r1

serverCert = <path>
* Full path to the PEM format server certificate file.
* Certificates are auto-generated by splunkd upon starting Splunk.
* You may replace the default cert with your own PEM format file.
* Default is $SPLUNK_HOME/etc/auth/server.pem.

sslKeysfile = <filename>
* DEPRECATED; use 'serverCert' instead.
* This file is in the directory specified by 'caPath' (see below).
* Default is server.pem.

sslPassword = <password>
* Server certificate password.
* Default is "password".

sslKeysfilePassword = <password>
* DEPRECATED; use 'sslPassword' instead.
```

```
sslRootCAPath = <path>
* Full path to the operating system's root CA (Certificate Authority)
  certificate store.
* The <path> must refer to a PEM format file containing one or more
root CA
  certificates concatenated together.
* Required for Common Criteria.
* NOTE: Splunk plans to submit Splunk Enterprise for Common Criteria
  evaluation. Splunk does not support using the product in Common
  Criteria mode until it has been certified by NIAP. See the "Securing
  Splunk Enterprise" manual for information on the status of Common
  Criteria certification.
* This setting is not used on Windows.
* Default is unset.

caCertFile = <filename>
* DEPRECATED; use 'sslRootCAPath' instead.
* Used only if 'sslRootCAPath' is unset.
* File name (relative to 'caPath') of the CA (Certificate Authority)
  certificate PEM format file containing one or more certificates
concatenated
  together.
* Default is cacert.pem.

dhFile = <path>
* PEM format Diffie-Hellman parameter file name.
* DH group size should be no less than 2048bits.
* This file is required in order to enable any Diffie-Hellman ciphers.
* Not set by default.

caPath = <path>
* DEPRECATED; use absolute paths for all certificate files.
* If certificate files given by other settings in this stanza are not
absolute
  paths, then they will be relative to this path.
* Default is $SPLUNK_HOME/etc/auth.

certCreateScript = <script name>
* Creation script for generating certs on startup of Splunk.

sendStrictTransportSecurityHeader = <bool>
* If set to true, the REST interface will send a
"Strict-Transport-Security"
  header with all responses to requests made over SSL.
* This can help avoid a client being tricked later by a
Man-In-The-Middle
  attack to accept a non-SSL request.  However, this requires a
commitment that
  no non-SSL web hosts will ever be run on this hostname on any port.
 For
  example, if splunkweb is in default non-SSL mode this can break the
  ability of browser to connect to it.  Enable with caution.
```

```
* Defaults to false

allowSslCompression = <bool>
* If set to true, the server will allow clients to negotiate
  SSL-layer data compression.
* Defaults to true.

allowSslRenegotiation = <bool>
* In the SSL protocol, a client may request renegotiation of the
connection
  settings from time to time.
* Setting this to false causes the server to reject all renegotiation
  attempts, breaking the connection.  This limits the amount of CPU a
  single TCP connection can use, but it can cause connectivity problems
  especially for long-lived connections.
* Defaults to true.
```

Splunkd HTTP server configuration

```
############################################################################
# Splunkd HTTP server configuration
############################################################################Splunkd
HTTP server configuration

[httpServer]
* Set stand-alone HTTP settings for Splunk under this stanza name.
* Follow this stanza name with any number of the following
attribute/value
  pairs.
* If you do not specify an entry for each attribute, Splunk uses the
default
  value.

atomFeedStylesheet = <string>
* Defines the stylesheet relative URL to apply to default Atom feeds.
* Set to 'none' to stop writing out xsl-stylesheet directive.
* Defaults to /static/atom.xsl.

max-age = <nonnegative integer>
* Set the maximum time (in seconds) to cache a static asset served off
of
  the '/static' directory.
* This value is passed along in the 'Cache-Control' HTTP header.
* Defaults to 3600.

follow-symlinks = true|false
* Toggle whether static file handler (serving the '/static' directory)
  follow filesystem symlinks when serving files.
* Defaults to false.
```

```
disableDefaultPort = true|false
* If true, turns off listening on the splunkd management port
  (8089 by default)
* This setting is not recommended:
  * This is the general communication path to splunkd.  If it is
disabled,
    there is no way to communicate with a running splunk.
  * This means many command line splunk invocations cannot function,
    splunkweb cannot function, the REST interface cannot function, etc.
  * If you choose to disable the port anyway, understand that you are
    selecting reduced Splunk functionality.
* Default value is 'false'.

acceptFrom = <network_acl> ...
* Lists a set of networks or addresses to accept data from.  These
rules are
  separated by commas or spaces
* Each rule can be in the following forms:
    1. A single IPv4 or IPv6 address (examples: "10.1.2.3", "fe80::4a3")
    2. A CIDR block of addresses (examples: "10/8", "fe80:1234/32")
    3. A DNS name, possibly with a '*' used as a wildcard (examples:
       "myhost.example.com", "*.splunk.com")
    4. A single '*' which matches anything
* Entries can also be prefixed with '!' to cause the rule to reject the
  connection.  Rules are applied in order, and the first one to match is
  used.  For example, "!10.1/16, *" will allow connections from
everywhere
  except the 10.1.*.* network.
* Defaults to "*" (accept from anywhere)

streamInWriteTimeout = <positive number>
* When uploading data to http server, if http server is unable to write
data
  to receiver for configured streamInWriteTimeout seconds, it aborts
write
  operation.
* Defaults to 5 seconds.

max_content_length = <int>
* Measured in bytes
* HTTP requests over this size will rejected.
* Exists to avoid allocating an unreasonable amount of memory from web
  requests
* Defaulted to 838860800 or 800MB
* In environments where indexers have enormous amounts of RAM, this
  number can be reasonably increased to handle large quantities of
  bundle data.

maxSockets = <int>
* The number of simultaneous HTTP connections that Splunk Enterprise
accepts
```

```
  simultaneously. You can limit this number to constrain resource
usage.
* If set to 0, Splunk Enterprise automatically sets it to one third of
the
  maximum allowable open files on the host.
* If this number is less than 50, it will be set to 50. If this number
is
  greater than 400000, it will be set to 400000.
* If set to a negative number, no limit will be enforced.
* Defaults to 0.

maxThreads = <int>
* The number of threads that can be used by active HTTP transactions.
  You can limit this number to constrain resource usage.
* If set to 0, Splunk Enterprise automatically sets the limit to
  one third of the maximum allowable threads on the host.
* If this number is less than 20, it will be set to 20. If this number
is
  greater than 150000, it will be set to 150000.
* If maxSockets is not negative and maxThreads is greater than
maxSockets, then
  Splunk Enterprise sets maxThreads to be equal to maxSockets.
* If set to a negative number, no limit will be enforced.
* Defaults to 0.

forceHttp10 = auto|never|always
* When set to "always", the REST HTTP server will not use some
  HTTP 1.1 features such as persistent connections or chunked
  transfer encoding.
* When set to "auto" it will do this only if the client sent no
  User-Agent header, or if the user agent is known to have bugs
  in its HTTP/1.1 support.
* When set to "never" it always will allow HTTP 1.1, even to
  clients it suspects may be buggy.
* Defaults to "auto"

crossOriginSharingPolicy = <origin_acl> ...
* List of the HTTP Origins for which to return Access-Control-Allow-*
(CORS)
  headers.
* These headers tell browsers that we trust web applications at those
sites
  to make requests to the REST interface
* The origin is passed as a URL without a path component (for example
  "https://app.example.com:8000")
* This setting can take a list of acceptable origins, separated
  by spaces and/or commas
* Each origin can also contain wildcards for any part.  Examples:
    *://app.example.com:*  (either HTTP or HTTPS on any port)
    https://*.example.com  (any host under example.com, including
example.com itself)
* An address can be prefixed with a '!' to negate the match, with
```

```
  the first matching origin taking precedence.  For example,
  "!*://evil.example.com:* *://*.example.com:*" to not avoid
  matching one host in a domain
* A single "*" can also be used to match all origins
* By default the list is empty

x_frame_options_sameorigin = true|false
* Adds a X-Frame-Options header set to "SAMEORIGIN" to every response
served by splunkd
* Defaults to true

allowEmbedTokenAuth = true|false
* If set to false, splunkd will not allow any access to artifacts
  that previously had been explicitly shared to anonymous users.
* This effectively disables all use of the "embed" feature.
* Defaults to true

cliLoginBanner = <string>
* Sets a message which will be added to the HTTP reply headers
  of requests for authentication, and to the "server/info" endpoint
* This will be printed by the Splunk CLI before it prompts
  for authentication credentials.  This can be used to print
  access policy information.
* If this string starts with a '"' character, it is treated as a
  CSV-style list with each line comprising a line of the message.
  For example: "Line 1","Line 2","Line 3"
* Defaults to empty (no message)

allowBasicAuth = true|false
* Allows clients to make authenticated requests to the splunk
  server using "HTTP Basic" authentication in addition to the
  normal "authtoken" system
* This is useful for programmatic access to REST endpoints and
  for accessing the REST API from a web browser.  It is not
  required for the UI or CLI.
* Defaults to true

basicAuthRealm = <string>
* When using "HTTP Basic" authenitcation, the 'realm' is a
  human-readable string describing the server.  Typically, a web
  browser will present this string as part of its dialog box when
  asking for the username and password.
* This can be used to display a short message describing the
  server and/or its access policy.
* Defaults to "/splunk"

allowCookieAuth = true|false
* Allows clients to request an HTTP cookie from the
/services/server/auth
  endpoint which can then be used to authenticate future requests
* Defaults to true
```

```
cookieAuthHttpOnly = true|false
* When using cookie based authentication, mark returned cookies
  with the "httponly" flag to tell the client not to allow javascript
  code to access its value
* Defaults to true
* NOTE: has no effect if allowCookieAuth=false

cookieAuthSecure = true|false
* When using cookie based authentication, mark returned cookies
  with the "secure" flag to tell the client never to send it over
  an unencrypted HTTP channel
* Defaults to true
* NOTE: has no effect if allowCookieAuth=false OR the splunkd REST
  interface has SSL disabled

dedicatedIoThreads = <int>
* If set to zero, HTTP I/O will be performed in the same thread
  that accepted the TCP connection.
* If set set to a non-zero value, separate threads will be run
  to handle the HTTP I/O, including SSL encryption.
* Defaults to "0"
* Typically this does not need to be changed.  For most usage
  scenarios using the same the thread offers the best performance.
```

Splunkd HTTPServer listener configuration

```
############################################################################
# Splunkd HTTPServer listener configuration
############################################################################Splunkd
HTTPServer listener configuration

[httpServerListener:<ip:><port>]
* Enable the splunkd REST HTTP server to listen on an additional port
number
  specified by <port>.  If a non-empty <ip> is included (for example:
  "[httpServerListener:127.0.0.1:8090]") the listening port will be
  bound only to a specific interface.
* Multiple "httpServerListener" stanzas can be specified to listen on
  more ports.
* Normally, splunkd listens only on the single REST port specified in
  web.conf's "mgmtHostPort" setting, and none of these stanzas need to
  be present.  Add these stanzas only if you want the REST HTTP server
  to listen to more than one port.

ssl = <bool>
* Toggle whether this listening ip:port will use SSL or not.
* Default value is 'true'.
* If the main REST port is SSL (the "enableSplunkdSSL" setting in this
  file's [sslConfig] stanza) and this stanza is set to "ssl=false" then
```

```
  clients on the local machine such as the CLI may connect to this port.

listenOnIPv6 = no|yes|only
* Toggle whether this listening ip:port will listen on IPv4, IPv6, or
both.
* If not present, the setting in the [general] stanza will be used

acceptFrom = <network_acl> ...
* Lists a set of networks or addresses to accept data from.  These
rules are
  separated by commas or spaces
* Each rule can be in the following forms:
    1. A single IPv4 or IPv6 address (examples: "10.1.2.3", "fe80::4a3")
    2. A CIDR block of addresses (examples: "10/8", "fe80:1234/32")
    3. A DNS name, possibly with a '*' used as a wildcard (examples:
       "myhost.example.com", "*.splunk.com")
    4. A single '*' which matches anything
* Entries can also be prefixed with '!' to cause the rule to reject the
  connection.  Rules are applied in order, and the first one to match is
  used.  For example, "!10.1/16, *" will allow connections from
everywhere
  except the 10.1.*.* network.
* Defaults to the setting in the [httpServer] stanza above
```

Static file handler MIME-type map

```
########################################################################
# Static file handler MIME-type map
########################################################################Static
file handler MIME-type map

[mimetype-extension-map]
* Map filename extensions to MIME type for files served from the static
file
  handler under this stanza name.

<file-extension> = <MIME-type>
* Instructs the HTTP static file server to mark any files ending
  in 'file-extension' with a header of 'Content-Type: <MIME-type>'.
* Defaults to:
    [mimetype-extension-map]
    gif = image/gif
    htm = text/html
    jpg = image/jpg
    png = image/png
    txt = text/plain
    xml = text/xml
    xsl = text/xml
```

Log rotation of splunkd_stderr.log & splunkd_stdout.log

```
############################################################################
# Log rotation of splunkd_stderr.log & splunkd_stdout.log
############################################################################Log
rotation of splunkd_stderr.log & splunkd_stdout.log

# These stanzas apply only on UNIX.  splunkd on Windows has no
# stdout.log or stderr.log

[stderr_log_rotation]
* Controls the data retention of the file containing all messages
written to
  splunkd's stderr file descriptor (fd 2).
* Typically this is extremely small, or mostly errors and warnings from
  linked libraries.

maxFileSize = <bytes>
* When splunkd_stderr.log grows larger than this value, it will be
rotated.
* maxFileSize is expresesed in bytes.
* You might want to increase this if you are working on a problem
  that involves large amounts of output to splunkd_stderr.log
* You might want to reduce this to allocate less storage to this log
category.
* Defaults to 10000000, which is 10 si-megabytes.

BackupIndex = <non-negative integer>
* How many rolled copies to keep.
  * For example, if this is 2, splunkd_stderr.log.1 and
splunkd_stderr.log.2
    may exist.  Further rolls will delete the current
splunkd_stderr.log.2
* You might want to increase this if you are working on a problem
  that involves large amounts of output to splunkd_stderr.log
* You might want to reduce this to allocate less storage to this log
category.
* Defaults to 2.

checkFrequency = <seconds>
* How often to check the size of splunkd_stderr.log
* Larger values may result in larger rolled file sizes but take less
resources.
* Smaller values may take more ressources but more accurately constrain
the
  file size.
* Defaults to 10, meaning 10 seconds.

[stdout_log_rotation]
* Controls the data retention of the file containing all messages
```

```
written to
  splunkd's stdout file descriptor (fd 1).
* Almost always, there is nothing in this file.

* The same settings exist for this stanza with the same defaults.  See
above
  for definitions.

maxFileSize = <bytes>
BackupIndex = <non-negative integer>
checkFrequency = <seconds>
```

Remote applications configuration (e.g. SplunkBase)

```
###############################################################################
# Remote applications configuration (e.g. SplunkBase)
###############################################################################Remote
applications configuration (e.g. SplunkBase)

[applicationsManagement]
* Set remote applications settings for Splunk under this stanza name.
* Follow this stanza name with any number of the following
attribute/value
  pairs.
* If you do not specify an entry for each attribute, Splunk uses the
default
  value.

allowInternetAccess = true|false
* Allow Splunk to access the remote applications repository.

url = <URL>
* Applications repository.
* Defaults to https://apps.splunk.com/api/apps

loginUrl = <URL>
* Applications repository login.
* Defaults to https://apps.splunk.com/api/account:login/

detailsUrl = <URL>
* Base URL for application information, keyed off of app ID.
* Defaults to https://apps.splunk.com/apps/id

useragent = <splunk-version>-<splunk-build-num>-<platform>
* User-agent string to use when contacting applications repository.
* <platform> includes information like operating system and CPU
architecture.

updateHost = <URL>
```

```
* Host section of URL to check for app updates, e.g.
https://apps.splunk.com

updatePath = <URL>
* Path section of URL to check for app updates
  For example: /api/apps:resolve/checkforupgrade

updateTimeout = <time range string>
* The minimum amount of time Splunk will wait between checks for app
updates
* Examples include '24h' (24 hours), '3d' (3 days),
  '7200s' (7200 seconds, or two hours)
* Defaults to '24h'

sslVersions = <versions_list>
* Comma-separated list of SSL versions to connect to 'url'
(https://apps.splunk.com).
* The versions available are "ssl3", "tls1.0", "tls1.1", and "tls1.2".
* The special version "*" selects all supported versions.  The version
"tls"
  selects all versions tls1.0 or newer.
* If a version is prefixed with "-" it is removed from the list.
* SSLv2 is always disabled; "-ssl2" is accepted in the version list but
does nothing.
* When configured in FIPS mode, ssl3 is always disabled regardless
  of this configuration.
* Defaults to "tls1.2".

sslVerifyServerCert = <bool>
* If this is set to true, Splunk verifies that the remote server
(specified in 'url')
  being connected to is a valid one (authenticated).  Both the common
  name and the alternate name of the server are then checked for a
  match if they are specified in 'sslCommonNameToCheck' and
'sslAltNameToCheck'.
  A certificiate is considered verified if either is matched.
* Default is true.

caCertFile = <path>
* Full path to a CA (Certificate Authority) certificate(s) PEM format
file.
* The <path> must refer to a PEM format file containing one or more
root CA
  certificates concatenated together.
* Used only if 'sslRootCAPath' is unset.
* Used for validating SSL certificate from https://apps.splunk.com/

sslCommonNameToCheck = <commonName1>, <commonName2>, ...
* If this value is set, and 'sslVerifyServerCert' is set to true,
  splunkd checks the common name(s) of the certificate presented by
  the remote server (specified in 'url') against this list of common
names.
```

```
* Defaults to 'apps.splunk.com'

sslCommonNameList = <commonName1>, <commonName2>, ...
* DEPRECATED; use 'sslCommonNameToCheck' instead.

sslAltNameToCheck =  <alternateName1>, <alternateName2>, ...
* If this value is set, and 'sslVerifyServerCert' is set to true,
  splunkd checks the alternate name(s) of the certificate presented by
  the remote server (specified in 'url') against this list of subject
alternate names.
* Defaults to 'splunkbase.splunk.com, apps.splunk.com'

cipherSuite = <cipher suite string>
* If set, uses the specified cipher string for making outbound HTTPS
connection.

ecdhCurves = <comma separated list of ec curves>
* ECDH curves to use for ECDH key negotiation.
* The curves should be specified in the order of preference.
* The client sends these curves as a part of Client Hello.
* We only support named curves specified by their SHORT names.
  (see struct ASN1_OBJECT in asn1.h)
* The list of valid named curves by their short/long names can be
obtained
  by executing this command:
  $SPLUNK_HOME/bin/splunk cmd openssl ecparam -list_curves
* Default is empty string.
* e.g. ecdhCurves = prime256v1,secp384r1,secp521r1
```

Misc. configuration

```
############################################################################
# Misc. configuration
############################################################################Misc.
configuration

[scripts]

initialNumberOfScriptProcesses = <num>
* The number of pre-forked script processes that are launched when the
  system comes up.  These scripts are reused when script REST endpoints
  *and* search scripts are executed.
  The idea is to eliminate the performance overhead of launching the
script
  interpreter every time it is invoked.  These processes are put in a
pool.
  If the pool is completely busy when a script gets invoked, a new
processes
  is fired up to handle the new invocation - but it disappears when
```

```
that
  invocation is finished.
```

Disk usage settings (for the indexer, not for Splunk log files)

```
################################################################################
# Disk usage settings (for the indexer, not for Splunk log files)
################################################################################Disk
usage settings (for the indexer, not for Splunk log files)

[diskUsage]

minFreeSpace = <num>
* Specified in megabytes.
* The default setting is 5000 (approx 5GB)
* Specifies a safe amount of space that must exist for splunkd to
continue
  operating.
* Note that this affects search and indexing
* For search:
  * Before attempting to launch a search, splunk will require this
amount of
    free space on the filesystem where the dispatch directory is stored,
    $SPLUNK_HOME/var/run/splunk/dispatch
  * Applied similarly to the search quota values in authorize.conf and
    limits.conf.
* For indexing:
  * Periodically, the indexer will check space on all partitions
    that contain splunk indexes as specified by indexes.conf.  Indexing
    will be paused and a ui banner + splunkd warning posted to indicate
    need to clear more disk space.

pollingFrequency = <num>
* After every pollingFrequency events indexed, the disk usage is
checked.
* The default frequency is every 100000 events.

pollingTimerFrequency = <num>
* After every pollingTimerFrequency seconds, the disk usage is checked
* The default value is 10 seconds
```

Queue settings

```
################################################################################
# Queue settings
```

```
############################################################################Queue
settings
[queue]

maxSize = [<integer>|<integer>[KB|MB|GB]]
* Specifies default capacity of a queue.
* If specified as a lone integer (for example, maxSize=1000), maxSize
  indicates the maximum number of events allowed in the queue.
* If specified as an integer followed by KB, MB, or GB (for example,
  maxSize=100MB), it indicates the maximum RAM allocated for queue.
* The default is 500KB.

cntr_1_lookback_time = [<integer>[s|m]]
* The lookback counters are used to track the size and count (number of
  elements in the queue) variation of the queues using an exponentially
  moving weighted average technique. Both size and count variation
  has 3 sets of counters each. The set of 3 counters is provided to be
able
  to track short, medium and long term history of size/count variation.
The
  user can customize the value of these counters or lookback time.
* Specifies how far into history should the size/count variation be
tracked
  for counter 1.
* It must be an integer followed by [s|m] which stands for seconds and
  minutes respectively.
* The default value for counter 1 is set to 60 seconds.

cntr_2_lookback_time = [<integer>[s|m]]
* See above for explanation and usage of the lookback counter.
* Specifies how far into history should the size/count variation be
tracked
  for counter 2.
* The default value for counter 2 is set to 600 seconds.

cntr_3_lookback_time = [<integer>[s|m]]
* See above for explanation and usage of the lookback counter..
* Specifies how far into history should the size/count variation be
tracked
  for counter 3.
* The default value for counter 3 is set to 900 seconds.

sampling_interval = [<integer>[s|m]]
* The lookback counters described above collects the size and count
  measurements for the queues.  This specifies at what interval the
  measurement collection will happen. Note that for a particular queue
all
  the counters sampling interval is same.
* It needs to be specified via an integer followed by [s|m] which stands
for
  seconds and minutes respectively.
* The default sampling_interval value is 1 second.
```

```
[queue=<queueName>]

maxSize = [<integer>|<integer>[KB|MB|GB]]
* Specifies the capacity of a queue. It overrides the default capacity
  specified in [queue].
* If specified as a lone integer (for example, maxSize=1000), maxSize
  indicates the maximum number of events allowed in the queue.
* If specified as an integer followed by KB, MB, or GB (for example,
  maxSize=100MB), it indicates the maximum RAM allocated for queue.
* The default is inherited from maxSize value specified in [queue]

cntr_1_lookback_time = [<integer>[s|m]]
* Same explanation as mentioned in [queue].
* Specifies the lookback time for the specific queue for counter 1.
* The default value is inherited from cntr_1_lookback_time value
specified
  in [queue].

cntr_2_lookback_time = [<integer>[s|m]]
* Specifies the lookback time for the specific queue for counter 2.
* The default value is inherited from cntr_2_lookback_time value
specified
  in [queue].

cntr_3_lookback_time = [<integer>[s|m]]
* Specifies the lookback time for the specific queue for counter 3.
* The default value is inherited from cntr_3_lookback_time value
specified
  in [queue].

sampling_interval = [<integer>[s|m]]
* Specifies the sampling interval for the specific queue.
* The default value is inherited from sampling_interval value specified
  in [queue].
```

PubSub server settings for the http endpoint.

```
############################################################################
# PubSub server settings for the http endpoint.
############################################################################PubSub
server settings for the http endpoint.

[pubsubsvr-http]

disabled = true|false
* If disabled, then http endpoint is not registered. Set this value to
  'false' to expose PubSub server on http.
* Defaults to 'true'
```

```
stateIntervalInSecs = <seconds>
* The number of seconds before a connection is flushed due to
inactivity.
  The connection is not closed, only messages for that connection are
  flushed.
* Defaults to 300 seconds (5 minutes).
```

General file input settings.

```
############################################################################
# General file input settings.
############################################################################General
file input settings.

[fileInput]

outputQueue = <queue name>
* The queue that input methods should send their data to.  Most users
will
  not need to change this value.
* Defaults to parsingQueue.
```

Settings controlling the behavior of 'splunk diag', the diagnostic tool

```
############################################################################
# Settings controlling the behavior of 'splunk diag', the diagnostic
tool
############################################################################Settings
controlling the behavior of 'splunk diag', the diagnostic tool

[diag]

# These settings provide defaults for invocations of the splunk diag
# command.  Generally these can be further modified by command line
flags to
# the diag command.

EXCLUDE-<class> = <glob expression>
* Specifies a glob / shell pattern to be excluded from diags generated
on
  this Splunk instance.
  * Example: */etc/secret_app/local/*.conf
* Further excludes can be added at the splunk diag command line, but
there
  is no facility to disable configuration-based excludes at the command
```

```
  line.
* There is one exclude by default, for the splunk.secret file.

# the following commands can be overridden entirely by their
command-line
# equivalents.

components = <comma separated list>
* Specifies which components of the diag should be gathered.
* This allows the disabling and enabling, categorically, of entire
portions
  of diag functionality.
* All of these components are further subject to the exclude feature
(see
  above), and component-specific filters (see below).
* Currently, with no configuration, all components except 'rest' are
enabled
  by default.
* Available components are:
  * index_files   : Files from the index that indicate their health
                    (Hosts|Sources|Sourcetypes.data and
bucketManifests).
                    User data is not collected.
  * index_listing : Directory listings of the index contents are
                    gathered, in order to see filenames, directory
names,
                    sizes, timestamps and the like.
  * etc           : The entire contents of the $SPLUNK_HOME/etc
                    directory.  In other words, the configuration
files.
  * log           : The contents of $SPLUNK_HOME/var/log/...
  * pool          : If search head pooling is enabled, the contents of
the
                    pool dir.
  * dispatch      : Search artifacts, without the actual results,
                    In other words var/run/splunk/dispatch, but not the
                    results or events files
  * searchpeers   : Directory listings of knowledge bundles replicated
for
                    distributed search
                    In other words: $SPLUNK_HOME/var/run/searchpeers
  * consensus     : Consensus protocol files produced by search head
clustering
                    In other words: $SPLUNK_HOME/var/run/splunk/_raft
  * conf_replication_summary : Directory listing of configuration
                    replication summaries produced by search head
clustering
                    In other words:
$SPLUNK_HOME/var/run/splunk/snapshot
  * rest          : The contents of a variety of splunkd endpoints
                    Includes server status messages (system banners),
                    licenser banners, configured monitor inputs &
```

```
tailing
                    file status (progress reading input files).
                    * On cluster masters, also gathers master info,
fixups,
                      current peer list, clustered index info, current
                      generation, & buckets in bad stats
                    * On cluster slaves, also gathers local buckets &
local
                      slave info, and the master information remotely
from
                      the configured master.
  * kvstore       : Directory listings of the KV Store data directory
                    contents are gathered, in order to see filenames,
                    directory names, sizes, and timestamps.
  * file_validate : Produce list of files that were in the install
media
                    which have been changed.  Generally this should be
an
                    empty list.

* The special value 'all' is also supported, enabling everything
explicitly.
* Further controlling the components from the command line:
    * The switch --collect replaces this list entirely.
        * Example: --collect log,etc
          This would set the componets to log and etc only, regardless
of
           onfig
    * The switch --enable adds a specific component to this list.
        * Example: --enable pool
          This would ensure that pool data is collected, regardless of
          config
    * The switch --disable removes a specific component from this list.
        * Example: --disable pool
          This would ensure that pool data is *NOT* collected,
regardless of
          config
* Currently, the default is to collect all components, save "rest".
* In the future there many be additional components which are not in the
  default set.
  * This may occur for new components that are expensive (large and/or
slow)
  * This may occur for new components that are preceived as sensitive

# Data filters; these further refine what is collected
# most of the existing ones are designed to limit the size and
collection
# time to pleasant values.

# note that most values here use underscores '_' while the command line
uses
# hyphens '-'
```

```
all_dumps = <bool>
* This setting currently is irrelevant on Unix platforms.
* Affects the 'log' component of diag. (dumps are written to the log dir
  on Windows)
* Can be overridden with the --all-dumps command line flag.
* Normally, Splunk diag will gather only three .DMP (crash dump) files
on
  Windows to limit diag size.
* If this is set to true, splunk diag will collect *all* .DMP files from
  the log directory.
* Defaults to unset / false (equivalent).

index_files = [full|manifests]
* Selects a detail level for the 'index_files' component.
* Can be overridden with the --index-files command line flag.
* 'manifests' limits the index file-content collection to just
  .bucketManifest files which give some information about Splunks idea
of
  the general state of buckets in an index.
* 'full' adds the collection of Hosts.data, Sources.data, and
  Sourcetypes.data which indicate the breakdown of count of items by
those
  categories per-bucket, and the timespans of those category entries
    * 'full' can take quite some time on very large index sizes,
especially
      when slower remote storage is involved.
* Defaults to 'manifests'

index_listing = [full|light]
* Selects a detail level for the 'index_listing' component.
* Can be overridden with the --index-listing command line flag.
* 'light' gets directory listings (ls, or dir) of the hot/warm and cold
  container directory locations of the indexes, as well as listings of
each
  hot bucket.
* 'full' gets a recursive directory listing of all the contents of every
  index location, which should mean all contents of all buckets.
  * 'full' may take significant time as well with very large bucket
counts,
    espeically on slower storage.
* Defaults to 'light'

etc_filesize_limit = <non-negative integer in kilobytes>
* This filters the 'etc' component
* Can be overridden with the --etc-filesize-limit command line flag
* This value is specified in kilobytes.
    * Example: 2000 - this would be approximately 2MB.
* Files in the $SPLUNK_HOME/etc directory which are larger than this
limit
  will not be collected in the diag.
* Diag will produce a message stating that a file has been skipped for
```

```
size
  to the console. (In practice we found these large files are often a
  surprise to the administrator and indicate problems).
* If desired, this filter may be entirely disabled by setting the value
  to 0.
* Currently, as a special exception, the file
$SPLUNK_HOME?etc/system/replication/ops.json
  is permitted to be 10x the size of this limit.
* Defaults to 10000 or 10MB.

log_age = <non-negative integer in days>
* This filters the 'log' component
* Can be overridden with the --log-age command line flag
* This value is specified in days
  * Example: 75 - this would be 75 days, or about 2.5 months.
* If desired, this filter may be entirely disabled by setting the value
to 0.
* The idea of this default filter is that data older than this is
rarely
  helpful in troubleshooting cases in any event.
* Defaults to 60, or approximately 2 months.

upload_proto_host_port = <protocol://host:port>|disabled
* URI base to use for uploading files/diags to Splunk support.
* If set to disabled (override in a local/server.conf file),
effectively
  disables diag upload functionality for this Splunk install.
* Modification may theoretically may permit operations with some forms
of
  proxies, but diag is not specifically designed for such, and support
of proxy
  configurations that do not currently work will be considered an
Enhancement
  Request.
* The communication path with api.splunk.com is over a simple but not
  documented protocol.  If for some reason you wish to accept diag
uploads into
  your own systems, it will probably be simpler to run diag and then
upload via
  your own means independently.  However if you have business reasons
that you
  want this built-in, get in touch.
* Uploading to unencrypted http definitely not recommended.
* Defaults to https://api.splunk.com
```

License manager settings for configuring the license pool(s)

```
############################################################################
# License manager settings for configuring the license pool(s)
```

```
############################################################################License
manager settings for configuring the license pool(s)

[license]
master_uri = [self|<uri>]
* An example of <uri>: <scheme>://<hostname>:<port>

active_group = Enterprise | Trial | Forwarder | Free
# these timeouts only matter if you have a master_uri set to remote
master
connection_timeout = 30
* Maximum time (in seconds) to wait before connection to master times
out

send_timeout = 30
* Maximum time (in seconds) to wait before sending data to master times
out

receive_timeout = 30
* Maximum time (in seconds) to wait before receiving data from master
times
  out

squash_threshold = <positive integer>
* Advanced setting.  Periodically the indexer must report to license
manager
  the data indexed broken down by source, sourcetype, host, and index.
 If
  the number of distinct (source,sourcetype,host,index) tuples grows
over
  the squash_threshold, we squash the {host,source} values and only
report a
  breakdown by {sourcetype,index}.  This is to prevent explosions in
  memory + license_usage.log lines.  Set this only after consulting a
Splunk
  Support engineer.  This needs to be set on license slaves as well as
license
  master.
* Default: 2000

report_interval = <nonnegative integer>[s|m|h]
* Selects a time period for reporting in license usage to the license
  master.
* This value is intended for very large deployments (hundreds of
indexers)
  where a large number of indexers may overwhelm the license server.
* The maximum permitted interval is 1 hour, and the minimum permitted
  interval is 1 minute.
* May be expressed as a positive number of seconds, minutes or hours.
* If no time unit is provided, seconds will be assumed.
* Defaults to 1 minute, or 1m.
```

```
strict_pool_quota = <boolean>
* Toggles strict pool quota enforcement
* If set to true, members of pools will receive warnings for a given day
if
  usage exceeds pool size regardless of whether overall stack quota was
  exceeded
* If set to false, members of pool will only receive warnings if both
pool
  usage exceeds pool size AND overall stack usage exceeds stack size
* Defaults to true

pool_suggestion = <string>
* Defaults to empty, which means this feature is disabled
* Suggest a pool to the master for this slave.
* The master will use this suggestion if the master doesn't have an
explicit
  rule mapping the slave to a given pool (ie...no slave list for the
  relevant license stack contains this slave explictly)
* If the pool name doesn't match any existing pool, it will be ignored,
no
  error will be generated
* This setting is intended to give an alternative management option for
  pool/slave mappings.  When onboarding an indexer, it may be easier to
  manage the mapping on the indexer itself via this setting rather than
  having to update server.conf on master for every addition of new
indexer
* NOTE: If you have multiple stacks and a slave maps to multiple pools,
this
        feature is limitted in only allowing a suggestion of a single
pool;
        This is not a common scenario however.

[lmpool:auto_generated_pool_forwarder]
* This is the auto generated pool for the forwarder stack

description = <textual description of this license pool>
quota = MAX|<maximum amount allowed by this license>
* MAX indicates the total capacity of the license. You may have only 1
pool
  with MAX size in a stack
* The quota can also be specified as a specific size eg. 20MB, 1GB etc

slaves = *|<slave list>
* An asterix(*) indicates that any slave can connect to this pool
* You can also specifiy a comma separated slave guid list

stack_id = forwarder
* The stack to which this pool belongs

[lmpool:auto_generated_pool_free]
* This is the auto generated pool for the free stack
* Field descriptions are the same as that for
```

```
  the "lmpool:auto_generated_pool_forwarder"

[lmpool:auto_generated_pool_enterprise]
* This is the auto generated pool for the enterprise stack
* Field descriptions are the same as that for
  the "lmpool:auto_generated_pool_forwarder"


[lmpool:auto_generated_pool_fixed-sourcetype_<sha256 hash of srctypes>]
* This is the auto generated pool for the enterprise fixed srctype stack
* Field descriptions are the same as that for
  the "lmpool:auto_generated_pool_forwarder"

[lmpool:auto_generated_pool_download_trial]
* This is the auto generated pool for the download trial stack
* Field descriptions are the same as that for
  the "lmpool:auto_generated_pool_forwarder"

############################################################################
#
# Search head pooling configuration
#
# Changes to a search head's pooling configuration must be made to:
#
#     $SPLUNK_HOME/etc/system/local/server.conf
#
# In other words, you may not deploy the [pooling] stanza via an app,
either
# on local disk or on shared storage.
#
# This is because these values are read before the configuration system
# itself has been completely initialized. Take the value of "storage",
for
# example.  This value cannot be placed within an app on shared storage
# because Splunk must use this value to find shared storage in the first
# place!
#
############################################################################

[pooling]

state = [enabled|disabled]
* Enables or disables search head pooling.
* Defaults to disabled.

storage = <path to shared storage>
* All members of a search head pool must have access to shared storage.
* Splunk will store configurations and search artifacts here.
* On *NIX, this should be an NFS mount.
* On Windows, this should be a UNC path to a Samba/CIFS share.

app_update_triggers = true|false|silent
```

```
* Should this search head run update triggers for apps modified by other
  search heads in the pool?
* For more information about update triggers specifically, see the
  [triggers] stanza in $SPLUNK_HOME/etc/system/README/app.conf.spec.
* If set to true, this search head will attempt to reload inputs,
indexes,
  custom REST endpoints, etc. stored within apps that are installed,
  updated, enabled, or disabled by other search heads.
* If set to false, this search head will not run any update triggers.
Note
  that this search head will still detect configuration changes and app
  state changes made by other search heads. It simply will not reload
any
  components within Splunk that might care about those changes, like
input
  processors or the HTTP server.
* Setting a value of "silent" is like setting a value of "true", with
one
  difference: update triggers will never result in restart banner
messages
  or restart warnings in the UI. Any need to restart will instead by
  signaled only by messages in splunkd.log.
* Defaults to true.

lock.timeout = <time range string>
* Timeout for acquiring file-based locks on configuration files.
* Splunk will wait up to this amount of time before aborting a
configuration
  write.
* Defaults to '10s' (10 seconds).

lock.logging = true|false
* When acquiring a file-based lock, log information into the locked
file.
* This information typically includes:
  * Which host is acquiring the lock
  * What that host intends to do while holding the lock
* There is no maximum filesize or rolling policy for this logging. If
you
  enable this setting, you must periodically truncate the locked file
  yourself to prevent unbounded growth.
* The information logged to the locked file is intended for debugging
  purposes only. Splunk makes no guarantees regarding the contents of
the
  file. It may, for example, write padding NULs to the file or truncate
the
  file at any time.
* Defaults to false.

# The following two intervals interelate; the longest possible time for
a
# state change to travel from one search pool member to the rest should
```

```
be
# approximately the sum of these two timers.
poll.interval.rebuild = <time range string>
* Rebuild or refresh in-memory configuration data structures at most
this
  often.
* Defaults to '1m' (1 minute).

poll.interval.check = <time range string>
* Check on-disk configuration files for changes at most this often.
* Defaults to '1m' (1 minute).

poll.blacklist.<name> = <regex>
* Do not check configuration files for changes if they match this
regular
  expression.
* Example: Do not check vim swap files for changes -- .swp$
```

High availability clustering configuration

```
##############################################################################
# High availability clustering configuration
##############################################################################High
availability clustering configuration

[clustering]

mode = [master|slave|searchhead|disabled]
* Sets operational mode for this cluster node.
* Only one master may exist per cluster.
* Defaults to disabled.

master_uri = [<uri> | clustermaster:stanzaName1,
clustermaster:stanzaName2]
* Only valid for mode=slave or searchhead
* URI of the cluster master that this slave or searchhead should
connect to.
* An example of <uri>: <scheme>://<hostname>:<port>
* Only for mode=searchhead - If the searchhead is a part of multiple
  clusters, the master uris can be specified by a comma separated list.

advertised_disk_capacity = <integer>
* Acceptable value range is 10 to 100.
* Percentage to use when advertising disk capacity to the cluster
master.
  This is useful for modifying weighted load balancing in indexer
discovery.
* For example, if you set this attribute to 50 for an indexer with a
```

```
500GB disk,
  the indexer will advertise its disk size as 250GB, not 500GB.
* Defaults to 100.

pass4SymmKey = <password>
* Secret shared among the nodes in the cluster to prevent any
  arbitrary node from connecting to the cluster. If a slave or
  searchhead is not configured with the same secret as the master,
  it will not be able to communicate with the master.
* Not set by default.
* If it is not set in the clustering stanza, the key will be looked in
the
  general stanza

service_interval = <zero or positive integer>
* Only valid for mode=master
* Specifies, in seconds, how often the master runs its service
  loop. In its service loop, the master checks the state of the
  peers and the buckets in the cluster and also schedules
  corrective action, if possible, for buckets that are not in
  compliance with replication policies.
* Defaults to 0
* A special default value of 0 indicates an auto mode where the service
interval
  for the next service call is determined by the time taken by previous
call.
  Service interval is bounded by the values 1 and
max_auto_service_interval.
  If previous service call takes more than max_auto_service_interval
seconds,
  next service interval will be set to max_auto_service_interval
seconds.

cxn_timeout = <seconds>
* Lowlevel timeout for establishing connection between cluster nodes.
* Defaults to 60s.

send_timeout = <seconds>
* Lowlevel timeout for sending data between cluster nodes.
* Defaults to 60s.

rcv_timeout = <seconds>
* Lowlevel timeout for receiving data between cluster nodes.
* Defaults to 60s.

rep_cxn_timeout = <seconds>
* Lowlevel timeout for establishing connection for replicating data.
* Defaults to 5s.

rep_send_timeout = <seconds>
* Lowlevel timeout for sending replication slice data between cluster
nodes.
```

```
* This is a soft timeout. When this timeout is triggered on source peer,
  it tries to determine if target is still alive. If it is still alive,
it
  reset the timeout for another rep_send_timeout interval and
continues.  If
  target has failed or cumulative timeout has exceeded
rep_max_send_timeout,
  replication fails.
* Defaults to 5s.

rep_rcv_timeout = <seconds>
* Lowlevel timeout for receiving acknowledgement data from peers.
* This is a soft timeout. When this timeout is triggered on source peer,
  it tries to determine if target is still alive. If it is still alive,
  it reset the timeout for another rep_send_timeout interval and
continues.
* If target has failed or cumulative timeout has exceeded
  rep_max_rcv_timeout, replication fails.
* Defaults to 10s.

search_files_retry_timeout = <seconds>
* Timeout after which request for search files from a peer is aborted.
* To make a bucket searchable, search specific files are copied from
another
  source peer with search files. If search files on source peers are
  undergoing chances, it asks requesting peer to retry after some time.
If
  cumulative retry period exceeds specified timeout, the requesting peer
  aborts the request and requests search files from another peer in the
  cluster that may have search files.
* Defaults to 600s.

re_add_on_bucket_request_error = true|false
* Valid only for mode=slave
* If set to true, slave re-add's itself to the cluster master if
  cluster master returns an error on any bucket request. On re-add,
  slave updates the master with the latest state of all its buckets.
* If set to false, slave doesn't re-add itself to the cluster master.
  Instead, it updates the master with those buckets that master
  returned an error.
* Defaults to false.

rep_max_send_timeout = <seconds>
* Maximum send timeout for sending replication slice data between
cluster
  nodes.
* On rep_send_timeout source peer determines if total send timeout has
  exceeded rep_max_send_timeout. If so, replication fails.
* If cumulative rep_send_timeout exceeds rep_max_send_timeout,
replication
  fails.
* Defaults to 600s.
```

```
rep_max_rcv_timeout = <seconds>
* Maximum cumulative receive timeout for receiving acknowledgement data
from
  peers.
* On rep_rcv_timeout source peer determines if total receive timeout has
  exceeded rep_max_rcv_timeout. If so, replication fails.
* Defaults to 600s.

multisite = [true|false]
* Turns on the multisite feature for this master.
* Make sure you set site parameters on the peers when you turn this to
true.
* Defaults to false.

replication_factor = <positive integer>
* Only valid for mode=master.
* Determines how many copies of rawdata are created in the cluster.
* Use site_replication_factor instead of this in case multisite is
turned
  on.
* Must be greater than 0.
* Defaults to 3

site_replication_factor = <comma-separated string>
* Only valid for mode=master and is only used if multisite is true.
* This specifies the per-site replication policy for any given
  bucket represented as a comma-separated list of per-site entries.
* Currently specified globally and applies to buckets in all
  indexes.
* Each entry is of the form <site-id>:<positive integer> which
  represents the number of copies to make in the specified site
* Valid site-ids include two mandatory keywords and optionally
  specific site-ids from site1 to site63
* The mandatory keywords are:
  - origin: Every bucket has a origin site which is the site of
  the peer that originally created this bucket. The notion of
  'origin' makes it possible to specify a policy that spans across
  multiple sites without having to enumerate it per-site.
  - total: The total number of copies we want for each bucket.
* When a site is the origin, it could potentially match both the
  origin and a specific site term. In that case, the max of the
  two is used as the count for that site.
* The total must be greater than or equal to sum of all the other
  counts (including origin).
* The difference between total and the sum of all the other counts
  is distributed across the remaining sites.
* Example 1: site_replication_factor = origin:2, total:3
  Given a cluster of 3 sites, all indexing data, every site has 2
  copies of every bucket ingested in that site and one rawdata
  copy is put in one of the other 2 sites.
* Example 2: site_replication_factor = origin:2, site3:1, total:3
```

```
  Given a cluster of 3 sites, 2 of them indexing data, every
  bucket has 2 copies in the origin site and one copy in site3. So
  site3 has one rawdata copy of buckets ingested in both site1 and
  site2 and those two sites have 2 copies of their own buckets.
* Defaults to origin:2, total:3

search_factor = <positive integer>
* Only valid for mode=master
* Determines how many buckets will have index structures pre-built.
* Must be less than or equal to replication_factor and greater than 0.
* Defaults to 2.

site_search_factor = <comma-separated string>
* Only valid for mode=master and is only used if multisite is true.
* This specifies the per-site policy for searchable copies for any
  given bucket represented as a comma-separated list of per-site
  entries.
* This is similar to site_replication_factor. Please see that
  entry for more information on the syntax.
* Defaults to origin:1, total:2

available_sites = <comma-separated string>
* Only valid for mode=master and is only used if multisite is true.
* This is a comma-separated list of all the sites in the cluster.
* Defaults to an empty string. So if multisite is turned on this needs
  to be explicitly set

site_mappings = <comma-separated string>
* Only valid for mode=master
* When you decommission a site, you must update this attribute so that
the origin
  bucket copies on the decommissioned site are mapped to a remaining
active site.
  This attribute maps decommissioned sites to active sites. The bucket
copies for
  which a decommissioned site is the origin site will then be
replicated to the active
  site specified by the mapping.
* Used only if multisite is true and sites have been decommissioned.
* Each comma-separated entry is of the form
<decommissioned_site_id>:<active_site_id>
  or default_mapping:<default_site_id>.
  <decommissioned_site_id> is a decommissioned site and
<active_site_id> is an existing site,
  specified in available_sites.
  For example, if available_sites=site1,site2,site3,site4 and you
decommission site2, you can
  map site2 to a remaining site such as site4, like this: site2:site4 .
* If a site used in a mapping is later decommissioned, its previous
mappings
  must be remapped to an available site. For instance, if you have the
mapping site1:site2
```

```
  but site2 is later decommissioned, you can remap both site1 and site2
to an active site3
  through the following replacement mappings - site1:site3,site2:site3
.
* Optional entry with syntax default_mapping:<default_site_id>
represents the
  default mapping, for cases where an explicit mapping site is not
specified.
  For example: default_mapping:site3 maps any decommissioned site to
site3, if they are not otherwise
  explicitly mapped to a site.
  There can only be one such entry.
* Defaults to an empty string.
* Example 1: site_mappings = site1:site3,default_mapping:site4.
  The cluster must include site3 and site4 in available_sites, and site1
must be decommissioned.
  The origin bucket copies for decommissioned site1 will be mapped to
site3.
  Bucket copies for any other decommissioned sites will be mapped to
site4.
* Example 2: site_mappings = site2:site3
  The cluster must include site3 in available_sites, and site2 must be
decommissioned
  The origin bucket copies for decommissioned site2 will be mapped to
site3.
  This cluster has no default.
* Example 3: site_mappings = default_mapping:site5
  The above cluster must include site5 in available_sites.
  The origin bucket copies for any decommissioned sites will be mapped
onto site5

heartbeat_timeout = <positive integer>
* Only valid for mode=master
* Determines when the master considers a slave down.  Once a slave
  is down, the master will initiate fixup steps to replicate
  buckets from the dead slave to its peers.
* Defaults to 60s.

access_logging_for_heartbeats = <bool>
* Only valid for mode=master
* Enables/disables logging to splunkd_access.log for peer heartbeats
* defaults to false (logging disabled)
* NOTE: you do not have to restart master to set this config parameter.
  Simply run the cli command on master:
    % splunk edit cluster-config -access_logging_for_heartbeats
<true|false>


restart_timeout = <positive integer>
* Only valid for mode=master
* This is the amount of time the master waits for a peer to come
  back when the peer is restarted (to avoid the overhead of
```

```
  trying to fixup the buckets that were on the peer).
* Note that this only works with the offline command or if the peer
  is restarted vi the UI.
* Defaults to 60s.

quiet_period = <positive integer>
* Only valid for mode=master
* This determines the amount of time for which the master is quiet
  right after it starts. During this period the master does not
  initiate any action but is instead waiting for the slaves to
  register themselves. At the end of this time period, it builds
  its view of the cluster based on the registered information and
  starts normal processing.
* Defaults to 60s.

generation_poll_interval = <positive integer>
* Only valid if mode=master or mode=searchhead
* Determines how often the searchhead polls the master for generation
  information.
* Defaults to 60s.

max_peer_build_load = <integer>
* This is the maximum number of concurrent tasks to make buckets
  searchable that can be assigned to a peer.
* Defaults to 2.

max_peer_rep_load = <integer>
* This is the maximum number of concurrent non-streaming
  replications that a peer can take part in as a target.
* Defaults to 5.

max_peer_sum_rep_load = <integer>
* This is the maximum number of concurrent summary replications
  that a peer can take part in as either a target or source.
* Defaults to 5.

max_replication_errors = <integer>
* Currently only valid for mode=slave
* This is the maximum number of consecutive replication errors
  (currently only for hot bucket replication) from a source peer
  to a specific target peer. Until this limit is reached, the
  source continues to roll hot buckets on streaming failures to
  this target. After the limit is reached, the source will no
  longer roll hot buckets if streaming to this specific target
  fails. This is reset if at least one successful (hot bucket)
  replication occurs to this target from this source.
* Defaults to 3.
* The special value of 0 turns off this safeguard; so the source
  always rolls hot buckets on streaming error to any target.

searchable_targets = true|false
* Only valid for mode=master
```

```
* Tells the master to make some replication targets searchable
  even while the replication is going on. This only affects
  hot bucket replication for now.
* Defaults to true

searchable_target_sync_timeout = <integer>
* Only valid for mode=slave
* If a hot bucket replication connection is inactive for this time
  (in seconds), a searchable target flushes out any pending search
  related in-memory files.
* Note that regular syncing - when the data is flowing through
  regularly and the connection is not inactive - happens at a
  faster rate (default of 5 secs controlled by
  streamingTargetTsidxSyncPeriodMsec in indexes.conf).
* The special value of 0 turns off this timeout behaviour.
* Defaults to 60 (seconds)

target_wait_time = <positive integer>
* Only valid for mode=master.
* Specifies the time that the master waits for the target of a
replication
  to register itself before it services the bucket again and potentially
  schedules another fixup.
* Defaults to 150s

summary_wait_time = <positive integer>
* Only valid for mode=master and summary_replication=true.
* Specifies the time that the master waits before scheduling fixups for
a
  newly 'done' summary that transitioned from 'hot_done'. This allows
for
  other copies of the 'hot_done' summary to also make their transition
into
  'done', avoiding unnecessary replications.
* Defaults to 660s

commit_retry_time = <positive integer>
* Only valid for mode=master
* Specifies the interval after which, if the last generation commit
failed,
  the master forces a retry. A retry is usually automatically kicked off
  after the appropriate events. This is just a backup to make sure that
the
  master does retry no matter what.
* Defaults to 300s

percent_peers_to_restart = <integer between 0-100>
* Suggested percentage of maximum peers to restart for rolling-restart.
* Actual percentage may vary due to lack of granularity for smaller peer
  sets.
* Regardless of setting, a minimum of 1 peer will be restarted per round
```

```
auto_rebalance_primaries = <bool>
* Only valid for mode=master
* Specifies if the master should automatically rebalance bucket
  primaries on certain triggers. Currently the only defined
  trigger is when a peer registers with the master. When a peer
  registers, the master redistributes the bucket primaries so the
  cluster can make use of any copies in the incoming peer.
* Defaults to true.

idle_connections_pool_size = <int>
* Only valid for mode=master
* Specifies how many idle http(s) connections we should keep alive to
reuse.
  Reusing connections improves the time it takes to send messages to
peers
  in the cluster.
* -1 (default) corresponds to "auto", letting the master determine the
  number of connections to keep around based on the number of peers in
the
  cluster.

use_batch_mask_changes = <bool>
* Only valid for mode=master
* Specifies if the master should process bucket mask changes in
  batch or inidividually one by one.
* Defaults to true.
* Set to false when there are 6.1 peers in the cluster for backwards
compatibility.

service_jobs_msec = <positive integer>
* Only valid for mode=master
* Max time in milliseconds cluster master spends in servicing finished
jobs
  per service call. Increase this if metrics.log has very high
current_size
  values.
* Defaults to 100ms.

summary_replication = true|false
* Only valid for mode=master.
* Turns on or off summary replication.
* Defaults to false.

rebalance_threshold = <number between 0.10 and 1.00>
* Only valid for mode=master.
* During rebalancing buckets amongst the cluster, this threshold is used
as a
  percentage to determine when our cluster is balanced.
* 1.00 is 100% indexers fully balanced.

max_auto_service_interval = <positive integer>
* Only valid for mode=master
```

```
* Only valid when service_interval is in auto mode (i.e
service_interval = 0)
* Indicates the maximum value that service interval is bounded by when
the
  service_interval is in auto mode. If the previous service call took
more than
  max_auto_service_interval seconds, the next service call will run
after
  max_auto_service_interval seconds.
* Defaults to 30 seconds.
* It is highly recommended that you choose a value that is one-half
  the smaller of heartbeat_timeout or restart_timeout. For example,
  the default value of 30 is based on the default value of 60 for both
  heartbeat_timeout and restart_timeout.

register_replication_address = <IP address, or fully qualified
machine/domain name>
* Only valid for mode=slave
* This is the address on which a slave will be available for accepting
  replication data. This is useful in the cases where a slave host
machine
  has multiple interfaces and only one of them can be reached by another
  splunkd instance

register_forwarder_address = <IP address, or fully qualified
machine/domain name>
* Only valid for mode=slave
* This is the address on which a slave will be available for accepting
  data from forwarder.This is useful in the cases where a splunk host
  machine has multiple interfaces and only one of them can be reached by
  another splunkd instance.

register_search_address = <IP address, or fully qualified
machine/domain name>
* Only valid for mode=slave
* This is the address on which a slave will be available as search
head.
  This is useful in the cases where a splunk host machine has multiple
  interfaces and only one of them can be reached by another splunkd
  instance.

executor_workers = <positive integer>
* Only valid if mode=master or mode=slave
* Number of threads that can be used by the clustering threadpool.
* Defaults to 10. A value of 0 will default to 1.

manual_detention = true|false
* Only valid for mode=slave
* Puts this peer node in manual detention.
* Defaults to "false".
* For the current release, this setting is for internal use only.
```

```
heartbeat_period = <non-zero positive integer>
* Only valid for mode=slave
* Controls the frequency the slave attempts to send heartbeats

notify_scan_period = <non-zero positive integer>
* Controls the frequency that the indexer scans summary folders for
summary updates.
* Only used when summary_replication is enabled on the Master.
* Defaults to 10 seconds.

enableS2SHeartbeat = true|false
* Only valid for mode=slave
* Splunk will monitor each replication connection for presence of
heartbeat,
  and if the heartbeat is not seen for s2sHeartbeatTimeout seconds, it
will
  close the connection.
* Defaults to true.

s2sHeartbeatTimeout = <seconds>
* This specifies the global timeout value for monitoring heartbeats on
  replication connections.
* Splunk will will close a replication connection if heartbeat is not
seen
  for s2sHeartbeatTimeout seconds.
* Defaults to 600 seconds (10 minutes). Replication source sends
heartbeat
  every 30 second.

throwOnBucketBuildReadError = true|false
* Valid only for mode=slave
* If set to true, index clustering slave throws an exception if it
encounters journal read error
  while building the bucket for a new searchable copy. It also throws
all the search & other
  files generated so far in this particular bucket build.
* If set to false, index clustering slave just logs the error and
preserves all the search & other
  files generated so far & finalizes them as it cannot proceed further
with this bucket.
* Defaults to false

cluster_label = <string>
* This specifies the label of the indexer cluster

[clustermaster:stanza1]
* Only valid for mode=searchhead when the searchhead is a part of
multiple
  clusters.

master_uri = <uri>
* Only valid for mode=searchhead when present in this stanza.
```

```
* URI of the cluster master that this searchhead should connect to.

pass4SymmKey = <password>
* Secret shared among the nodes in the cluster to prevent any
  arbitrary node from connecting to the cluster. If a searchhead
  is not configured with the same secret as the master,
  it will not be able to communicate with the master.
* Not set by default.
* If it is not present here, the key in the clustering stanza will be
used.
  If it is not present in the clustering stanza, the value in the
general
  stanza will be used.

site = <site-id>
* Specifies the site this searchhead belongs to for this particular
master
  when multisite is enabled (see below).
* Valid values for site-id include site1 to site63.

multisite = [true|false]
* Turns on the multisite feature for this master_uri for the searchhead.
* Make sure the master has the multisite feature turned on.
* Make sure you specify the site in case this is set to true. If no
  configuration is found in the clustermaster stanza, we default to any
  value for site that might be defined in the [general]
  stanza.
* Defaults to false.

[replication_port://<port>]
# Configure Splunk to listen on a given TCP port for replicated data
from
# another cluster member.
# If mode=slave is set in the [clustering] stanza at least one
# replication_port must be configured and not disabled.

disabled = true|false
* Set to true to disable this replication port stanza.
* Defaults to false.

listenOnIPv6 = no|yes|only
* Toggle whether this listening port will listen on IPv4, IPv6, or both.
* If not present, the setting in the [general] stanza will be used.

acceptFrom = <network_acl> ...
* Lists a set of networks or addresses to accept connections from.
 These
  rules are separated by commas or spaces
* Each rule can be in the following forms:
    1. A single IPv4 or IPv6 address (examples: "10.1.2.3", "fe80::4a3")
    2. A CIDR block of addresses (examples: "10/8", "fe80:1234/32")
    3. A DNS name, possibly with a '*' used as a wildcard (examples:
```

```
       "myhost.example.com", "*.splunk.com")
    4. A single '*' which matches anything
* Entries can also be prefixed with '!' to cause the rule to reject the
  connection.  Rules are applied in order, and the first one to match is
  used.  For example, "!10.1/16, *" will allow connections from
everywhere
  except the 10.1.*.* network.
* Defaults to "*" (accept replication data from anywhere)

[replication_port-ssl://<port>]
* This configuration is same as replication_port stanza above but uses
SSL.

disabled = true|false
* Set to true to disable this replication port stanza.
* Defaults to false.

listenOnIPv6 = no|yes|only
* Toggle whether this listening port will listen on IPv4, IPv6, or both.
* If not present, the setting in the [general] stanza will be used.

acceptFrom = <network_acl> ...
* This setting is same as setting in replication_port stanza defined
above.

serverCert = <path>
* Full path to file containing private key and server certificate.
* The <path> must refer to a PEM format file.
* There is no default value.

sslPassword = <password>
* Server certificate password, if any.
* There is no default value.

password = <password>
* DEPRECATED; use 'sslPassword' instead.

rootCA = <path>
* DEPRECATED; use '[sslConfig]/sslRootCAPath' instead.
* Full path to the root CA (Certificate Authority) certificate store.
* The <path> must refer to a PEM format file containing one or more
root CA
  certificates concatenated together.
* Default is unset.

cipherSuite = <cipher suite string>
* If set, uses the specified cipher string for the SSL connection.
* If not set, uses the default cipher string.
* provided by OpenSSL.  This is used to ensure that the server does not
  accept connections using weak encryption protocols.
* Must specify 'dhFile' to enable any Diffie-Hellman ciphers.
```

```
sslVersions = <versions_list>
* Comma-separated list of SSL versions to support.
* The versions available are "ssl3", "tls1.0", "tls1.1", and "tls1.2".
* The special version "*" selects all supported versions.  The version
"tls"
  selects all versions tls1.0 or newer.
* If a version is prefixed with "-" it is removed from the list.
* SSLv2 is always disabled; "-ssl2" is accepted in the version list but
does nothing.
* When configured in FIPS mode, ssl3 is always disabled regardless
  of this configuration.
* Defaults to "*,-ssl2" (anything newer than SSLv2).

ecdhCurves = <comma separated list of ec curves>
* ECDH curves to use for ECDH key negotiation.
* The curves should be specified in the order of preference.
* The client sends these curves as a part of Client Hello.
* The server supports only the curves specified in the list.
* We only support named curves specified by their SHORT names.
  (see struct ASN1_OBJECT in asn1.h)
* The list of valid named curves by their short/long names can be
obtained
  by executing this command:
  $SPLUNK_HOME/bin/splunk cmd openssl ecparam -list_curves
* Default is empty string.
* e.g. ecdhCurves = prime256v1,secp384r1,secp521r1

dhFile = <path>
* PEM format Diffie-Hellman parameter file name.
* DH group size should be no less than 2048bits.
* This file is required in order to enable any Diffie-Hellman ciphers.
* Not set by default.

dhfile = <path>
* DEPRECATED; use 'dhFile' instead.

supportSSLV3Only = <bool>
* DEPRECATED.  SSLv2 is now always disabled.  The exact set of SSL
versions
  allowed is now configurable via the "sslVersions" setting above.

useSSLCompression = <bool>
* If true, enables SSL compression.
* Defaults to true.

compressed = <bool>
* DEPRECATED; use 'useSSLCompression' instead.
* Used only if 'useSSLCompression' is unset.

requireClientCert = <bool>
* Requires that any peer that connects to replication port has a
certificate
```

```
  that can be validated by certificate authority specified in rootCA.
* Default is false.

allowSslRenegotiation = <bool>
* In the SSL protocol, a client may request renegotiation of the
connection
  settings from time to time.
* Setting this to false causes the server to reject all renegotiation
  attempts, breaking the connection.  This limits the amount of CPU a
  single TCP connection can use, but it can cause connectivity problems
  especially for long-lived connections.
* Defaults to true.

sslCommonNameToCheck = <commonName1>, <commonName2>, ...
* Optional. Defaults to no common name checking.
* Check the common name of the client's certificate against this list of
names.
* requireClientCert must be set to true for this setting to work.

sslAltNameToCheck =  <alternateName1>, <alternateName2>, ...
* Optional. Defaults to no alternate name checking.
* Check the alternate name of the client's certificate against this list
of names.
* If there is no match, assume that Splunk is not authenticated against
this
  server.
* requireClientCert must be set to true for this setting to work.
```

Introspection settings

```
###############################################################################
# Introspection settings
###############################################################################Introspecti
settings

[introspection:generator:disk_objects]
* For 'introspection_generator_addon', packaged with Splunk; provides
the
  data ("i-data") consumed, and reported on, by
'introspection_viewer_app'
  (due to ship with a future release).
* This stanza controls the collection of i-data about: indexes; bucket
  superdirectories (homePath, coldPath, ...); volumes; search dispatch
  artifacts.
* On forwaders the collection of index, volumes and dispatch disk
objects
  is disabled.

acquireExtra_i_data = true | false
```

```
* If true, extra Disk Objects i-data is emitted; you can gain more
insight
  into your site, but at the cost of greater resource consumption both
  directly (the collection itself) and indirectly (increased disk and
  bandwidth utilization, to store the produced i-data).
* Please consult documentation for list of regularly emitted Disk
Objects
  i-data, and extra Disk Objects i-data, appropriate to your release.
* Defaults to: false.

collectionPeriodInSecs = <positive integer>
* Controls frequency of Disk Objects i-data collection; higher frequency
  (hence, smaller period) gives a more accurate picture, but at the cost
of
  greater resource consumption both directly (the collection itself) and
  indirectly (increased disk and bandwidth utilization, to store the
  produced i-data).
* Defaults to: 600 (10 minutes).

[introspection:generator:disk_objects__indexes]
  * This stanza controls the collection of i-data about indexes.
  * Inherits the values of 'acquireExtra_i_data' and
'collectionPeriodInSecs'
    attributes from the 'introspection:generator:disk_objects' stanza,
but
    may be enabled/disabled independently of it.
  * This stanza should only be used to force collection of i-data about
    indexes on dedicated forwarders.
  * Enabled by default.

[introspection:generator:disk_objects__volumes]
  * This stanza controls the collection of i-data about volumes.
  * Inherits the values of 'acquireExtra_i_data' and
'collectionPeriodInSecs'
    attributes from the 'introspection:generator:disk_objects' stanza,
but
    may be enabled/disabled independently of it.
  * This stanza should only be used to force collection of i-data about
    volumes on dedicated forwarders.
  * Enabled by default.

[introspection:generator:disk_objects__dispatch]
  * This stanza controls the collection of i-data about search dispatch
artifacts.
  * Inherits the values of 'acquireExtra_i_data' and
'collectionPeriodInSecs'
    attributes from the 'introspection:generator:disk_objects' stanza,
but
    may be enabled/disabled independently of it.
  * This stanza should only be used to force collection of i-data about
    search dispatch artifacts on dedicated forwarders.
  * Enabled by default.
```

```
[introspection:generator:disk_objects__fishbucket]
* This stanza controls the collection of i-data about:
  $SPLUNK_DB/fishbucket, where we persist per-input status of
file-based
  inputs.
* Inherits the values of 'acquireExtra_i_data' and
'collectionPeriodInSecs'
  attributes from the 'introspection:generator:disk_objects' stanza,
but may
  be enabled/disabled independently of it.

[introspection:generator:disk_objects__bundle_replication]
* This stanza controls the collection of i-data about:
  bundle replication metrics of distributed search
* Inherits the values of 'acquireExtra_i_data' and
'collectionPeriodInSecs'
  attributes from the 'introspection:generator:disk_objects' stanza,
but may
  be enabled/disabled independently of it.

[introspection:generator:disk_objects__partitions]
* This stanza controls the collection of i-data about: disk partition
space
  utilization.
* Inherits the values of 'acquireExtra_i_data' and
'collectionPeriodInSecs'
  attributes from the 'introspection:generator:disk_objects' stanza,
but may
  be enabled/disabled independently of it.

[introspection:generator:disk_objects__summaries]
* Introspection data about summary disk space usage. Summary disk usage
  includes both data model and report summaries. The usage is collected
  per summaryId, locally at each indexer.

disabled = true | false
* If not specified, inherits the value from
  'introspection:generator:disk_objects' stanza.

collectionPeriodInSecs = <positive integer>
* Controls frequency of Disk Objects - summaries collection; higher
frequency
  (hence, smaller period) gives a more accurate picture, but at the cost
of
  greater resource consumption directly (the summaries collection
itself);
  it is not recommended for a period less than 15 minutes.
* If you enable summary collection, the first collection happens 5
minutes
  after the Splunk instance is started. For every subsequent
collection, this
```

```
  setting is honored.
* If 'collectionPeriodInSecs' smaller than 5 * 60, it will be set back
to
  30 minutes internally.
* Set to (N*300) seconds. Any remainder is ignored.
* Defaults to: 1800 (30 minutes).

[introspection:generator:resource_usage]
* For 'introspection_generator_addon', packaged with Splunk; provides
the
  data ("i-data") consumed, and reported on, by
'introspection_viewer_app'
  (due to ship with a future release).
* "Resource Usage" here refers to: CPU usage; scheduler overhead; main
  (physical) memory; virtual memory; pager overhead; swap; I/O; process
  creation (a.k.a. forking); file descriptors; TCP sockets;
receive/transmit
  networking bandwidth.
* Resource Usage i-data is collected at both hostwide and per-process
  levels; the latter, only for processes associated with this
SPLUNK_HOME.
* Per-process i-data for Splunk search processes will include
additional,
  search-specific, information.

acquireExtra_i_data = true | false
* If true, extra Resource Usage i-data is emitted; you can gain more
insight
  into your site, but at the cost of greater resource consumption both
  directly (the collection itself) and indirectly (increased disk and
  bandwidth utilization, to store the produced i-data).
* Please consult documentation for list of regularly emitted Resource
Usage
  i-data, and extra Resource Usage i-data, appropriate to your release.
* Defaults to: false.

collectionPeriodInSecs = <positive integer>
* Controls frequency of Resource Usage i-data collection; higher
frequency
  (hence, smaller period) gives a more accurate picture, but at the cost
of
  greater resource consumption both directly (the collection itself) and
  indirectly (increased disk and bandwidth utilization, to store the
  produced i-data).
* Defaults to: 600 (10 minutes) on UFs, 10 (1/6th of a minute) on
non-UFs.

[introspection:generator:resource_usage__iostats]
* This stanza controls the collection of i-data about: IO Statistics
data
* "IO Statistics" here refers to: read/write requests; read/write sizes;
  io service time; cpu usage during service
```

```
* IO Statistics i-data is sampled over the collectionPeriodInSecs
* Does not inherit the value of the 'collectionPeriodInSecs' attribute
from the
  'introspection:generator:resource_usage' stanza, and may be
enabled/disabled
  independently of it.

collectionPeriodInSecs = <positive integer>
* Controls interval of IO Statistics i-data collection; higher intervals
  gives a more accurate picture, but at the cost of greater resource
consumption
  both directly (the collection itself) and indirectly (increased disk
and
  bandwidth utilization, to store the produced i-data).
* Defaults to: 60 (1 minute)

[introspection:generator:kvstore]
* For 'introspection_generator_addon', packaged with Splunk
* "KV Store" here refers to: statistics information about KV Store
process.

serverStatsCollectionPeriodInSecs = <positive integer>
* Controls frequency of KV Store server status collection
* Defaults to: 27 seconds.

collectionStatsCollectionPeriodInSecs = <positive integer>
* Controls frequency of KV Store db statistics collection
* Defaults to: 600 seconds.

profilingStatsCollectionPeriodInSecs = <positive integer>
* Controls frequency of KV Store profiling data collection
* Defaults to: 5 seconds

rsStatsCollectionPeriodInSecs = <positive integer>
* Controls frequency of KV Store replica set stats collectiok
* Defaults to: 60 seconds
```

Settings used to control commands started by Splunk

```
############################################################################
# Settings used to control commands started by Splunk
############################################################################Settings
used to control commands started by Splunk

[commands:user_configurable]

prefix = <path>
* All non-internal commands started by splunkd will be prefixed with
this
```

```
  string, allowing for "jailed" command execution.
* Should be only one word.  In other words, commands are supported, but
  commands and arguments are not.
* Applies to commands such as: search scripts, scripted inputs, SSL
  certificate generation scripts.  (Any commands that are
  user-configurable).
* Does not apply to trusted/non-configurable command executions, such
as:
  splunk search, splunk-optimize, gunzip.
* Default is empty (no prefix).
```

search head clustering configuration

```
###############################################################################
# search head clustering configuration
###############################################################################search
head clustering configuration

[shclustering]
disabled = true|false
* Disables or enables search head clustering on this instance.
* Defaults to true; that is, disabled.
* When enabled, the captain needs to be selected via a
  bootstrap mechanism. Once bootstrapped, further captain
  selections are made via a dynamic election mechanism.
* When enabled, you will also need to specify the cluster member's own
server
  address / management uri for identification purpose. This can be
  done in 2 ways: by specifying the mgmt_uri attribute individually on
  each member or by specifying pairs of 'GUID, mgmt-uri' strings in the
  servers_list attribute.

mgmt_uri = [ mgmt-URI ]
* The management uri is used to identify the cluster member's own
address to
  itself.
* Either mgmt_uri or servers_list is necessary.
* mgmt_uri is simpler to author but is unique for each member.
* servers_list is more involved, but can be copied as a config string to
  all members in the cluster.

servers_list = [ <(GUID, mgmt-uri);>+ ]
* A semicolon separated list of instance GUIDs and management URIs.
* Each member will use its GUID to identify its own management URI.

adhoc_searchhead = <bool>
* This setting configures a member as an adhoc searchhead; i.e., the
member
  will not run any scheduled jobs.
```

```
* Use the setting captain_is_adhoc_searchhead to reduce compute load on
the
  captain.
* Defaults to false.

no_artifact_replications = <bool>
* prevent this Search Head Cluster member to be selected as a target for
replications.
* This is an advanced setting, and not to be changed without proper
understanding of the implications.
* Defaults to false

captain_is_adhoc_searchhead = <bool>
* This setting prohibits the captain from running scheduled jobs.
Captain
  will be dedicated to controlling the activities of the cluster, but
can
  also run adhoc search jobs from clients.
* Defaults to false.

preferred_captain = <bool>
* The cluster tries to assign captaincy to a member with
preferred_captain=true.
* Note that it is not always possible to assign captaincy to a member
with
  preferred_captain=true - for example, if none of the preferred
members is
  reachable over the network. In that case, captaincy might remain on a
  member with preferred_captain=false.
* Defaults to true

replication_factor = <positive integer>
* Determines how many copies of search artifacts are created in the
cluster.
* This must be set to the same value on all members.
* Defaults to 3.

pass4SymmKey = <password>
* Secret shared among the members in the search head cluster to prevent
any
  arbitrary instance from connecting to the cluster.
* All members must use the same value.
* If set in the [shclustering] stanza, it takes precedence over any
setting
  in the [general] stanza.
* Defaults to 'changeme' from the [general] stanza in the default
  server.conf.

async_replicate_on_proxy = <bool>
* If the jobs/${sid}/results REST endpoint had to be proxied to a
different
  member due to missing local replica, this attribute will automatically
```

```
  schedule an async replication to that member when set to true.
* Default is true.

master_dump_service_periods = <int>
* If SHPMaster info is switched on in log.cfg, then captain statistics
will
  be dumped in splunkd.log after the specified number of service
periods.
  Purely a debugging aid.
* Default is 500.

long_running_jobs_poll_period = <int>
* Long running delegated jobs will be polled by the captain every
  "long_running_jobs_poll_period" seconds to ascertain whether they are
  still running, in order to account for potential node/member failure.
* Default is 600, i.e. 10 minutes

scheduling_heuristic = <string>
* This setting configures the job distribution heuristic on the captain.
* There are currently two supported strategies: 'round_robin' or
  'scheduler_load_based'.
* Default is 'scheduler_load_based'.

id = <GUID>
* Unique identifier for this cluster as a whole, shared across all
cluster
  members.
* By default, Splunk will arrange for a unique value to be generated and
  shared across all members.

cxn_timeout = <seconds>
* Low-level timeout for establishing connection between cluster members.
* Defaults to 60s.

send_timeout = <seconds>
* Low-level timeout for sending data between search head cluster
members.
* Defaults to 60s.

rcv_timeout = <seconds>
* Low-level timeout for receiving data between search head cluster
members.
* Defaults to 60s.

cxn_timeout_raft = <seconds>
* Low-level timeout for establishing connection between search head
cluster
  members for the raft protocol.
* Defaults to 2s.

send_timeout_raft = <seconds>
* Low-level timeout for sending data between search head cluster members
```

```
for
  the raft protocol.
* Defaults to 5s.

rcv_timeout_raft = <seconds>
* Low-level timeout for receiving data between search head cluster
members
  for the raft protocol.
* Defaults to 5s.


rep_cxn_timeout = <seconds>
* Low-level timeout for establishing connection for replicating data.
* Defaults to 5s.

rep_send_timeout = <seconds>
* Low-level timeout for sending replication slice data between cluster
  members.
* This is a soft timeout. When this timeout is triggered on source peer,
  it tries to determine if target is still alive. If it is still alive,
  it reset the timeout for another rep_send_timeout interval and
continues.
  If target has failed or cumulative timeout has exceeded
  rep_max_send_timeout, replication fails.
* Defaults to 5s.

rep_rcv_timeout = <seconds>
* Low-level timeout for receiving acknowledgement data from members.
* This is a soft timeout. When this timeout is triggered on source
member,
  it tries to determine if target is still alive. If it is still alive,
  it reset the timeout for another rep_send_timeout interval and
continues.
  If target has failed or cumulative timeout has exceeded
  rep_max_rcv_timeout, replication fails.
* Defaults to 10s.

rep_max_send_timeout = <seconds>
* Maximum send timeout for sending replication slice data between
cluster
  members.
* On rep_send_timeout source peer determines if total send timeout has
  exceeded rep_max_send_timeout. If so, replication fails.
* If cumulative rep_send_timeout exceeds rep_max_send_timeout,
replication
  fails.
* Defaults to 600s.

rep_max_rcv_timeout = <seconds>
* Maximum cumulative receive timeout for receiving acknowledgement data
from
  members.
```

```
* On rep_rcv_timeout source member determines if total receive timeout
has
  exceeded rep_max_rcv_timeout. If so, replication fails.
* Defaults to 600s.

log_heartbeat_append_entries = <bool>
* If true, Splunk will log the the low-level heartbeats between members
in
  splunkd_access.log  . These heartbeats are used to maintain the
authority
  of the captain authority over other members.
* Defaults to false.

election_timeout_ms = <positive_integer>
* The amount of time that a member will wait before trying to become
the
  captain.
* Half of this value is the heartbeat period.
* A very low value of election_timeout_ms can lead to unnecessary
captain
  elections.
* The default is 60000ms, or 1 minute.

election_timeout_2_hb_ratio = <positive_integer>
* The ratio between the election timeout and the heartbeat time.
* A typical ratio between 5 - 20 is desirable. Default is 12 to keep the
  heartbeat time at 5s.
* This ratio determines the number of heartbeat attempts that would fail
  before a member starts to timeout and tries to become the captain.

heartbeat_timeout = <positive integer>
* Determines when the captain considers a member down. Once a member
  is down, the captain will initiate fixup steps to replicate
  artifacts from the dead member to its peers.
* Defaults to 60s.

access_logging_for_heartbeats = <bool>
* Only valid on captain
* Enables/disables logging to splunkd_access.log for member heartbeats
* Defaults to false (logging disabled)
* NOTE: you do not have to restart captain to set this config
parameter.
  Simply run the cli command on master:

  % splunk edit shcluster-config -access_logging_for_heartbeats
<true|false>

restart_timeout = <positive integer>
* This is the amount of time the captain waits for a member to come
  back when the instance is restarted (to avoid the overhead of
  trying to fixup the artifacts that were on the peer).
```

```
quiet_period = <positive integer>
* This determines the amount of time for which a newly elected
  captain waits for members to join. During this period the
  captain does not initiate any fixups but instead waits for the
  members to register themselves. Job scheduling and conf
  replication still happen as usual during this time. At the end
  of this time period, the captain builds its view of the cluster
  based on the registered peers and starts normal
  processing.
* Defaults to 60s.

max_peer_rep_load = <integer>
* This is the maximum number of concurrent replications that a
  member can take part in as a target.
* Defaults to 5.

target_wait_time = <positive integer>
* Specifies the time that the captain waits for the target of a
replication
  to register itself before it services the artifact again and
potentially
  schedules another fixup.
* Defaults to 150s.

percent_peers_to_restart = <integer between 0-100>
* The percentage of members to restart at one time during rolling
restarts.
* Actual percentage may vary due to lack of granularity for smaller peer
  sets regardless of setting, a minimum of 1 peer will be restarted per
  round.
* Do not set this attribute to a value greater than 20%. Otherwise,
issues
  can arise during the captain election process.

rolling_restart_with_captaincy_exchange = <bool>
* If this boolean is turned on, captain will try to exchange captaincy
with another
* node during rolling restart
* Default = true
* if you change it to false, captain will restart and captaincy will
transfer to
* some other node

register_replication_address = <IP address, or fully qualified
machine/domain name>
* This is the address on which a member will be available for accepting
  replication data. This is useful in the cases where a member host
machine
  has multiple interfaces and only one of them can be reached by another
  splunkd instance.

executor_workers = <positive integer>
```

```
* Number of threads that can be used by the search head clustering
  threadpool.
* Defaults to 10. A value of 0 will be interpreted as 1.

heartbeat_period = <non-zero positive integer>
* Controls the frequency with which the member attempts to send
heartbeats.

enableS2SHeartbeat = true|false
* Splunk will monitor each replication connection for presence of
heartbeat.
  If the heartbeat is not seen for s2sHeartbeatTimeout seconds, it will
  close the connection.
* Defaults to true.

s2sHeartbeatTimeout = <seconds>
* This specifies the global timeout value for monitoring heartbeats on
  replication connections.
* Splunk will will close a replication connection if heartbeat is not
seen
  for s2sHeartbeatTimeout seconds.
* Replication source sends heartbeat every 30 second.
* Defaults to 600 seconds (10 minutes).

captain_uri = [ static-captain-URI ]
* The management uri of static captain is used to identify the cluster
captain for a static captain.

election = <bool>
* This is used to classify a cluster as static or dynamic (RAFT based).
* election = false means static captain, which is used for DR situation.
* election = true means dynamic captain election enabled through RAFT
protocol

mode = <member>
* Accepted values are captain and member, mode is used to identify the
function of a node in
     static search head cluster. Setting mode as captain assumes it to
function as both captain and a member.
#proxying related
sid_proxying = <bool>
* Enable or disable search artifact proxying. Changing this will impact
the
  proxying of search results, and jobs feed will not be cluster-aware.
* Only for internal/expert use.
* Defaults to true.

ss_proxying = <bool>
* Enable or disable saved search proxying to captain. Changing this
will
  impact the behavior of Searches and Reports Page.
* Only for internal/expert use.
```

```
* Defaults to true.

ra_proxying = <bool>
* Enable or disable saved report acceleration summaries proxying to
captain.
  Changing this will impact the behavior of report acceleration
summaries
  page.
* Only for internal/expert use.
* Defaults to true.

alert_proxying = <bool>
* Enable or disable alerts proxying to captain. Changing this will
impact
  the behavior of alerts, and essentially make them not cluster-aware.
* Only for internal/expert use.
* Defaults to true.

csv_journal_rows_per_hb = <int>
* Controls how many rows of CSV from the delta-journal are sent per hb
* Used for both alerts and suppressions
* Do not alter this value without contacting splunk support.
* Defaults to 10000

conf_replication_period = <int>
* Controls how often, in seconds, a cluster member replicates
configuration changes.
* A value of 0 disables automatic replication of configuration changes.
* Defaults to 5

conf_replication_max_pull_count = <int>
* Controls the maximum number of configuration changes a member will
  replicate from the captain at one time.
* A value of 0 disables any size limits.
* Defaults to 1000.

conf_replication_max_push_count = <int>
* Controls the maximum number of configuration changes a member will
  replicate to the captain at one time.
* A value of 0 disables any size limits.
* Defaults to 100.

conf_replication_include.<conf_file_name> = <bool>
* Controls whether Splunk replicates changes to a particular type of
*.conf
  file, along with any associated permissions in *.meta files.
* Defaults to false.

conf_replication_summary.whitelist.<name> = <whitelist_pattern>
* Whitelist files to be included in configuration replication summaries.

conf_replication_summary.blacklist.<name> = <blacklist_pattern>
```

```
* Blacklist files to be excluded from configuration replication
summaries.

conf_replication_summary.concerning_file_size = <int>
* Any individual file within a configuration replication summary that is
  larger than this value (in MB) will trigger a splunkd.log warning
message.
* Defaults to 50.

conf_replication_summary.period = <timespan>
* Controls how often configuration replication summaries are created.
* Defaults to '1m' (1 minute).

conf_replication_purge.eligibile_count = <int>
* Controls how many configuration changes must be present before any
become
  eligible for purging.
* In other words: controls the minimum number of configuration changes
  Splunk will remember for replication purposes.
* Defaults to 20000.

conf_replication_purge.eligibile_age = <timespan>
* Controls how old a configuration change must be before it is eligible
for
  purging.
* Defaults to '1d' (1 day).

conf_replication_purge.period = <timespan>
* Controls how often configuration changes are purged.
* Defaults to '1h' (1 hour).

conf_deploy_repository = <path>
* Full path to directory containing configurations to deploy to cluster
  members.

conf_deploy_staging = <path>
* Full path to directory where preprocessed configurations may be
written
  before being deployed cluster members.

conf_deploy_concerning_file_size = <int>
* Any individual file within <conf_deploy_repository> that is larger
than
  this value (in MB) will trigger a splunkd.log warning message.
* Defaults to: 50

conf_deploy_fetch_url = <URL>
* Specifies the location of the deployer from which members fetch the
  configuration bundle.
* This value must be set to a <URL> in order for the configuration
bundle to
  be fetched.
```

```
* Defaults to empty.

conf_deploy_fetch_mode = auto|replace|none
* Controls configuration bundle fetching behavior when the member starts
up.
* When set to "replace", a member checks for a new configuration bundle
on
  every startup.
* When set to "none", a member does not fetch the configuration bundle
on
  startup.
* Regarding "auto":
  * If no configuration bundle has yet been fetched, "auto" is
equivalent
    to "replace".
  * If the configuration bundle has already been fetched, "auto" is
    equivalent to "none".
* Defaults to "replace".

artifact_status_fields = <field> ...
    * Give a comma separated fields to pick up values from status.csv
and info.csv for each search artifacts.
    * These fields will be shows in cli/rest endpoint splunk list
shcluster-member-artifacts
    * Default values user, app, label

encrypt_fields = <field> ...
    * These are the fields that need to be re-encrypted when Search Head
Cluster
      does its own first time run on syncing all members with a new
splunk.secret key
    * Give a comma separated fields as a triple elements
<conf-file>:<stanza-prefix>:<key elem>
    * For matching all stanzas from a conf, leave the stanza-prefix
empty, eg: "server: :pass4SymmKey" matches all stanzas with
pass4SymmKey as key in server.conf
    * Default values include storage/passwords, secret key for
clustering/shclustering, server ssl config

enable_jobs_data_lite = <bool>
*This is for memory reduction on the captain for Search head clustering,
leads to lower memory
* in captain while slaves send the artifacts status.csv as a string.
* Default : false

shcluster_label = <string>
* This specifies the label of the search head cluster

retry_autosummarize_or_data_model_acceleration_jobs = <bool>
* Controls whether the captain tries a second time to delegate an
  auto-summarized or data model acceleration job, if the first attempt
to
```

```
  delegate the job fails.
* Defaults to true.

[replication_port://<port>]
# Configures the member to listen on a given TCP port for replicated
data
# from another cluster member.
* At least one replication_port must be configured and not disabled.

disabled = true|false
* Set to true to disable this replication port stanza.
* Defaults to false.

listenOnIPv6 = no|yes|only
* Toggle whether this listening port will listen on IPv4, IPv6, or both.
* If not present, the setting in the [general] stanza will be used.

acceptFrom = <network_acl> ...
* Lists a set of networks or addresses to accept connections from.
These
  rules are separated by commas or spaces.
* Each rule can be in the following forms:
    1. A single IPv4 or IPv6 address (examples: "10.1.2.3", "fe80::4a3")
    2. A CIDR block of addresses (examples: "10/8", "fe80:1234/32")
    3. A DNS name, possibly with a '*' used as a wildcard (examples:
       "myhost.example.com", "*.splunk.com")
    4. A single '*' which matches anything
* Entries can also be prefixed with '!' to cause the rule to reject the
  connection.  Rules are applied in order, and the first one to match is
  used.  For example, "!10.1/16, *" will allow connections from
everywhere
  except the 10.1.*.* network.
* Defaults to "*" (accept replication data from anywhere)

[replication_port-ssl://<port>]
* This configuration is same as replication_port stanza above but uses
SSL.

disabled = true|false
* Set to true to disable this replication port stanza.
* Defaults to false.

listenOnIPv6 = no|yes|only
* Toggle whether this listening port will listen on IPv4, IPv6, or both.
* If not present, the setting in the [general] stanza will be used.

acceptFrom = <network_acl> ...
* This setting is same as setting in replication_port stanza defined
above.

serverCert = <path>
* Full path to file containing private key and server certificate.
```

```
* The <path> must refer to a PEM format file.
* There is no default value.

sslPassword = <password>
* Server certificate password, if any.
* There is no default value.

password = <password>
* DEPRECATED; use 'sslPassword' instead.
* Used only if 'sslPassword' is unset.

rootCA = <path>
* DEPRECATED; use '[sslConfig]/sslRootCAPath' instead.
* Used only if '[sslConfig]/sslRootCAPath' is unset.
* Full path to the root CA (Certificate Authority) certificate store.
* The <path> must refer to a PEM format file containing one or more
root CA
  certificates concatenated together.
* Default is unset.

cipherSuite = <cipher suite string>
* If set, uses the specified cipher string for the SSL connection.
* If not set, uses the default cipher string.
* provided by OpenSSL.  This is used to ensure that the server does not
  accept connections using weak encryption protocols.

supportSSLV3Only = <bool>
* DEPRECATED.  SSLv2 is now always disabled.  The exact set of SSL
versions
  allowed is now configurable via the "sslVersions" setting above.

useSSLCompression = <bool>
* If true, enables SSL compression.
* Defaults to true.

compressed = <bool>
* DEPRECATED; use 'useSSLCompression' instead.
* Used only if 'useSSLCompression' is unset.

requireClientCert = <bool>
* Requires that any peer that connects to replication port has a
certificate
  that can be validated by certificate authority specified in rootCA.
* Default is false.

allowSslRenegotiation = <bool>
* In the SSL protocol, a client may request renegotiation of the
connection
  settings from time to time.
* Setting this to false causes the server to reject all renegotiation
  attempts, breaking the connection.  This limits the amount of CPU a
  single TCP connection can use, but it can cause connectivity problems
```

```
  especially for long-lived connections.
* Defaults to true.
```

KV Store configuration

```
############################################################################
# KV Store configuration
############################################################################KV
Store configuration
[kvstore]

disabled = true|false
* Set to true to disable the KV Store process on the current server. To
  completely disable KV Store in a deployment with search head
clustering or
  search head pooling, you must also disable KV Store on each individual
  server.
* Defaults to false.

port = <port>
* Port to connect to the KV Store server.
* Defaults to 8191.

replicaset = <replset>
* Replicaset name.
* Defaults to splunkrs.

distributedLookupTimeout = <seconds>
* This setting has been removed, as it is no longer needed

shutdownTimeout = <seconds>
* Time in seconds to wait for a clean shutdown of the KV Store. If this
time
  is reached after signaling for a shutdown, KV Store will be terminated
  forcibly.
* Defaults to 100 seconds.

initAttempts = <int>
* The maximum number of attempts to initialize the KV Store when
starting
  splunkd.
* Defaults to 300.

replication_host = <host>
* The host name to access the KV Store.
* This setting has no effect on a single Splunk instance.
* When using search head clustering, if the "replication_host" value is
not
  set in the [kvstore] stanza, the host you specify for
```

```
  "mgmt_uri" in the [shclustering] stanza is used for KV
  Store connection strings and replication.
* In search head pooling, this host value is a requirement for using KV
  Store.
* This is the address on which a kvstore will be available for
accepting
  remotely.

verbose = true|false
* Set to true to enable verbose logging.
* Defaults to false.

verboseLevel = <nonnegative integer>
* When verbose logging is enabled specify verbose level for logging
  from 0 to 5, where 5 is the most verbose.
* Defaults to 2.

dbPath = <path>
* Path where KV Store data is stored.
* Changing this directory after initial startup does not move existing
data.
  The contents of the directory should be manually moved to the new
  location.
* Defaults to $SPLUNK_DB/kvstore.

oplogSize = <int>
* The size of the replication operation log, in MB, for environments
  with search head clustering or search head pooling.
  In a standalone environment, 20% of this size is used.
* Defaults to 1000MB (1GB).
* Once the KV Store has created the oplog for the first time, changing
this
  setting will NOT affect the size of the oplog. A full backup and
restart
  of the KV Store will be required.
* Do not change this setting without first consulting with Splunk
Support.

replicationWriteTimeout = <int>
* The time to wait, in seconds, for replication to complete while saving
KV store
  operations. When the value is 0, the process never times out.
* Used for replication environments (search head clustering or search
  head pooling).
* Defaults to 1800 seconds (30 minutes).

caCertFile = <path>
* DEPRECATED; use '[sslConfig]/sslRootCAPath' instead.
* Used only if 'sslRootCAPath' is unset.
* Full path to a CA (Certificate Authority) certificate(s) PEM format
file.
* If specified, it will be used in KV Store SSL connections and
```

```
  authentication.
* Only used when Common Criteria is enabled (SPLUNK_COMMON_CRITERIA=1)
  or FIPS is enabled (i.e. SPLUNK_FIPS=1).
* NOTE: Splunk plans to submit Splunk Enterprise for Common Criteria
  evaluation. Splunk does not support using the product in Common
  Criteria mode until it has been certified by NIAP. See the "Securing
  Splunk Enterprise" manual for information on the status of Common
  Criteria certification.
* Default is $SPLUNK_HOME/etc/auth/cacert.pem

caCertPath = <filepath>
* DEPRECATED; use '[sslConfig]/sslRootCAPath' instead.

serverCert = <filepath>
* A certificate file signed by the signing authority specified above by
  caCertPath.
* In search head clustering or search head pooling, the certificates at
  different members must share the same ?subject'.
* The Distinguished Name (DN) found in the certificate?s subject, must
  specify a non-empty value for at least one of the following
attributes:
  Organization (O), the Organizational Unit (OU) or the
  Domain Component (DC).
* Only used when Common Criteria is enabled (SPLUNK_COMMON_CRITERIA=1)
  or FIPS is enabled (i.e. SPLUNK_FIPS=1).
* NOTE: Splunk plans to submit Splunk Enterprise for Common Criteria
  evaluation. Splunk does not support using the product in Common
  Criteria mode until it has been certified by NIAP. See the "Securing
  Splunk Enterprise" manual for information on the status of Common
  Criteria certification.

sslKeysPath = <filepath>
* DEPRECATED; use 'serverCert' instead.
* Used only when 'serverCert' is empty.

sslPassword = <password>
* Password of the private key in the file specified by 'serverCert'
above.
* Must be specified if FIPS is enabled (i.e. SPLUNK_FIPS=1), otherwise,
KV
  Store will not be available. There is no default value.
* Only used when Common Criteria is enabled (SPLUNK_COMMON_CRITERIA=1)
  or FIPS is enabled (i.e. SPLUNK_FIPS=1).
* NOTE: Splunk plans to submit Splunk Enterprise for Common Criteria
  evaluation. Splunk does not support using the product in Common
  Criteria mode until it has been certified by NIAP. See the "Securing
  Splunk Enterprise" manual for information on the status of Common
  Criteria certification.

sslKeysPassword = <password>
* DEPRECATED; use 'sslPassword' instead.
* Used only when 'sslPassword' is empty.
```

```
sslCRLPath = <filepath>
* Certificate Revocation List file.
* Optional. Defaults to no Revocation List.
* Only used when Common Criteria is enabled (SPLUNK_COMMON_CRITERIA=1)
  or FIPS is enabled (i.e. SPLUNK_FIPS=1).
* NOTE: Splunk plans to submit Splunk Enterprise for Common Criteria
  evaluation. Splunk does not support using the product in Common
  Criteria mode until it has been certified by NIAP. See the "Securing
  Splunk Enterprise" manual for information on the status of Common
  Criteria certification.

modificationsReadIntervalMillisec = <int>
* Specifies how often, in milliseconds, to check for modifications to KV
Store
  collections in order to replicate changes for distributed searches.
* Defaults to 1000.

modificationsMaxReadSec = <int>
* Maximum time interval KVStore can spend while checking for
modifications
  before it produces collection dumps for distributed searches.
* Defaults to 30.

[indexer_discovery]
pass4SymmKey = <password>
* Security key shared between master node and forwarders.
* If specified here, the same value must also be specified on all
forwarders
  connecting to this master.

polling_rate = <integer>
* A value between 1 to 10. This value affects the forwarder polling
frequency to
  achieve the desired polling rate. The number of connected forwarders
is also
  taken into consideration.
* The formula used to determine effective polling interval, in
Milliseconds, is:
(number_of_forwarders/polling_rate + 30 seconds) * 1000
* Defaults to 10.

indexerWeightByDiskCapacity = <bool>
* If set to true, it instructs the forwarders to use weighted load
balancing.
  In weighted load balancing, load balancing is based on the total disk
capacity
  of the target indexers, with the forwarder streaming more data to
indexers
  with larger disks.
*  The traffic sent to each indexer is based on the ratio of:
   indexer_disk_capacity/total_disk_capacity_of_indexers_combined
```

```
* Defaults to false.
```

Raft Statemachine configuration

```
############################################################################
# Raft Statemachine configuration
############################################################################Raft
Statemachine configuration
[raft_statemachine]

disabled = true|false
* Set to true to disable the raft statemachine.
* This feature require search head clustering to be enabled.
* Any consensus replication among search heads use this feature
* Defaults to true.

replicate_search_peers = true|false
* Add/remove search-server request is applied on all members
  of a search head cluster, when this value to set to true.
* Require a healty search head cluster with a captain.
```

server.conf.example

```
#   Version 6.5.0
#
# This file contains an example server.conf.  Use this file to configure
SSL
# and HTTP server options.
#
# To use one or more of these configurations, copy the configuration
block
# into server.conf in $SPLUNK_HOME/etc/system/local/. You must restart
# Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

# Allow users 8 hours before they time out
[general]
sessionTimeout=8h
pass4SymmKey = changeme

# Listen on IPv6 in addition to IPv4...
```

```
listenOnIPv6 = yes
# ...but make all outgoing TCP connections on IPv4 exclusively
connectUsingIpVersion = 4-only

# Turn on SSL:

[sslConfig]
enableSplunkdSSL = true
useClientSSLCompression = true
sslKeysfile = server.pem
sslKeysfilePassword = password
caCertFile = cacert.pem
caPath = $SPLUNK_HOME/etc/auth
certCreateScript = genMyServerCert.sh



######## SSO Example ########
# This example trusts all logins from the splunk web server and
localhost
# Note that a proxy to the splunk web server should exist to enforce
# authentication
[general]
trustedIP = 127.0.0.1


############################################################################
# Set this node to be a cluster master.
############################################################################


[clustering]
mode = master
replication_factor = 3
pass4SymmKey = someSecret
search_factor = 2


############################################################################
# Set this node to be a slave to cluster master "SplunkMaster01" on
port
# 8089.
############################################################################

[clustering]
mode = slave
master_uri = https://SplunkMaster01.example.com:8089
pass4SymmKey = someSecret

############################################################################
# Set this node to be a searchhead to cluster master "SplunkMaster01" on
# port 8089.
```

```
############################################################################
[clustering]
mode = searchhead
master_uri = https://SplunkMaster01.example.com:8089
pass4SymmKey = someSecret

############################################################################
# Set this node to be a searchhead to multiple cluster masters -
# "SplunkMaster01" with pass4SymmKey set to 'someSecret and
"SplunkMaster02"
# with no pass4SymmKey set here.
############################################################################
[clustering]
mode = searchhead
master_uri = clustermaster:east, clustermaster:west

[clustermaster:east]
master_uri=https://SplunkMaster01.example.com:8089
pass4SymmKey=someSecret

[clustermaster:west]
master_uri=https://SplunkMaster02.example.com:8089

############################################################################
# Open an additional non-SSL HTTP REST port, bound to the localhost
# interface (and therefore not accessible from outside the machine)
 Local
# REST clients like the CLI can use this to avoid SSL overhead when not
# sending data across the network.
############################################################################
[httpServerListener:127.0.0.1:8090]
ssl = false
```

serverclass.conf

The following are the spec and example files for serverclass.conf.

serverclass.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attributes and values for defining server
# classes to which deployment clients can belong. These attributes and
# values specify what content a given server class member will receive
from
# the deployment server.
```

```
#
# For examples, see serverclass.conf.example. You must reload
deployment
# server ("splunk reload deploy-server"), or restart splunkd, for
changes to
# this file to take effect.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles



#*******************************************************************************
# Configure the server classes that are used by a deployment server
instance.
#
# Server classes are essentially categories.  They use filters to
control
# what clients they apply to, contain a set of applications, and may
define
# deployment server behavior for the management of those applications.
 The
# filters can be based on DNS name, IP address, build number of client
# machines, platform, and the so-called clientName.  If a target machine
# matches the filter, then the apps and configuration content that make
up
# the server class will be deployed to it.

# Property Inheritance
#
# Stanzas in serverclass.conf go from general to more specific, in the
# following order:
# [global] -> [serverClass:<name>] ->
[serverClass:<scname>:app:<appname>]
#
# Some properties defined at a general level (say [global]) can be
# overridden by a more specific stanza as it applies to them. All
# overridable properties are marked as such.
```

FIRST LEVEL: global ############

```
##############################################
############ FIRST LEVEL: global ############
##############################################FIRST LEVEL: global
```

```
############

# Global stanza that defines properties for all server classes.
[global]

disabled = true|false
* Toggles deployment server component off and on.
* Set to true to disable.
* Defaults to false.

crossServerChecksum = true|false
* Ensures that each app will have the same checksum across different
deployment
  servers.
* Useful if you have multiple deployment servers behind a load-balancer.
* Defaults to false.

excludeFromUpdate = <path>[,<path>]...
* Specifies paths to one or more top-level files or directories (and
their
  contents) to exclude from being touched during app update.  Note that
  each comma-separated entry MUST be prefixed by "$app_root$/"
(otherwise a
  warning will be generated).
* Can be overridden at the serverClass level.
* Can be overridden at the app level.
* Requires version 6.2.x or higher for both the Deployment Server and
Client.

repositoryLocation = <path>
* The repository of applications on the server machine.
* Can be overridden at the serverClass level.
* Defaults to $SPLUNK_HOME/etc/deployment-apps

targetRepositoryLocation = <path>
* The location on the deployment client where to install the apps
defined
  for this Deployment Server.
* If this value is unset, or set to empty, the repositoryLocation path
is used.
* Useful only with complex (for example, tiered) deployment strategies.
* Defaults to $SPLUNK_HOME/etc/apps, the live
  configuration directory for a Splunk instance.

tmpFolder = <path>
* Working folder used by deployment server.
* Defaults to $SPLUNK_HOME/var/run/tmp

continueMatching = true | false
* Controls how configuration is layered across classes and
server-specific
  settings.
```

```
* If true, configuration lookups continue matching server classes,
beyond
  the first match.
* If false, only the first match will be used.
* A serverClass can override this property and stop the matching.
* Matching is done in the order in which server classes are defined.
* Can be overridden at the serverClass level.
* Defaults to true

endpoint = <URL template string>
* The endpoint from which content can be downloaded by a deployment
client.
  The deployment client knows how to substitute values for variables in
the
  URL.
* Any custom URL can also be supplied here, as long as it uses the
specified
  variables.
* Need not be specified unless you have a very specific need, for
example:
  To acquire deployment application files from a third-party Web server,
for
  extremely large environments.
* Can be overridden at the serverClass level.
* Defaults to
$deploymentServerUri$/services/streams/deployment?name=$serverClassName$:$appName$

filterType = whitelist | blacklist
* The whitelist setting indicates a filtering strategy that pulls in a
  subset:
    * Items are not considered to match the stanza by default.
    * Items that match any whitelist entry, and do not match any
blacklist
      entry are considered to match the stanza.
    * Items that match any blacklist entry are not considered to match
the
      stanza, regardless of whitelist.
* The blacklist setting indicates a filtering strategy that rules out a
subset:
    * Items are considered to match the stanza by default.
    * Items that match any blacklist entry, and do not match any
whitelist
      entry are considered to not match the stanza.
    * Items that match any whitelist entry are considered to match the
      stanza.
* More briefly:
    * whitelist: default no-match -> whitelists enable -> blacklists
disable
    * blacklist: default match -> blacklists disable-> whitelists enable
* Can be overridden at the serverClass level, and the serverClass:app
level.
* Defaults to whitelist
```

```
whitelist.<n> = <clientName> | <IP address> | <hostname> | <instanceId>
blacklist.<n> = <clientName> | <IP address> | <hostname> | <instanceId>
* 'n' is an unsigned integer. The sequence may start at any value and
may be
  non-consecutive.
* The value of this attribute is matched against several things in
order:
    * Any clientName specified by the client in its
deploymentclient.conf file
    * The IP address of the connected client
    * The hostname of the connected client, as provided by reverse DNS
lookup
    * The hostname of the client, as provided by the client
    * For Splunk version > 6.4, the instanceId of the client. This is a
GUID
      string, e.g. 'ffe9fe01-a4fb-425e-9f63-56cc274d7f8b'.
* All of these can be used with wildcards.  * will match any sequence
of
  characters.  For example:
    * Match a network range: 10.1.1.*
    * Match a domain: *.splunk.com
* Can be overridden at the serverClass level, and the serverClass:app
level.
* There are no whitelist or blacklist entries by default.
* These patterns are PCRE regular expressions, with the following aids
for
  easier entry:
    * You can specify simply '.' to mean '\.'
    * You can specify simply '*' to mean '.*'
* Matches are always case-insensitive; you do not need to specify the
'(?i)' prefix.

# Note: Overriding one type of filter (whitelist/blacklist) causes the
other to
# be overridden (and hence not inherited from parent) too.

# Example with filterType=whitelist:
#     whitelist.0=*.splunk.com
#     blacklist.0=printer.splunk.com
#     blacklist.1=scanner.splunk.com
# This will cause all hosts in splunk.com, except 'printer' and
'scanner', to
# match this server class.

# Example with filterType=blacklist:
#     blacklist.0=*
#     whitelist.0=*.web.splunk.com
#     whitelist.1=*.linux.splunk.com
# This will cause only the 'web' and 'linux' hosts to match the server
class.
# No other hosts will match.
```

```
# Deployment client machine types (hardware type of respective host
machines)
# can also be used to match DCs.
# This filter will be used only if match of a client could not be
decided using
# the whitelist/blacklist filters.  The value of each machine type is
# designated by the hardware platform itself; a few common ones are:
#       linux-x86_64, windows-intel, linux-i686, freebsd-i386,
darwin-i386, sunos-sun4u.
# The method for finding it varies by platform; once a deployment client
is
# connected to the DS, however, you can determine the value of DC's
machine
# type with this Splunk CLI command on the DS:
#       <code>./splunk list deploy-clients</code>
# The <code>utsname</code> values in the output are the respective DCs'
machine
# types.

whitelist.from_pathname = <pathname>
blacklist.from_pathname = <pathname>
* As as alternative to a series of (whitelist|blacklist).<n>, the
<clientName>,
  <IP address>, and <hostname> list can be imported from <pathname> that
is
  either a plain text file or a comman-separated values (CSV) file.
* May be used in conjunction with (whitelist|blacklist).select_field,
  (whitelist|blacklist).where_field, and
(whitelist|blacklist).where_equals.
* If used by itself, then <pathname> specifies a plain text file where
one
  <clientName>, <IP address>, or <hostname> is given per line.
* If used on conjuction with select_field, where_field, and
where_equals, then
  <pathname> specifies a CSV file.
* The <pathname> is relative to $SPLUNK_HOME.
* May also be used in conjunction with (whitelist|blacklist).<n> to
specify
  additional values, but there is no direct relation between them.
* At most one from_pathname may be given per stanza.

whitelist.select_field = <field name> | <positive integer>
blacklist.select_field = <field name> | <positive integer>
* Specifies which field of the CSV file contains the <clientName>, <IP
address>,
  or <hostname> either by field name or number.
* If <field name> is given, then the first line of the CSV file MUST be
a
  header line containing the name(s) of all the field(s) and <field
name>
  specifies which field contains the value(s) to be used.  Note that
```

```
field
  names are case-sensitive.
* If <positive integer> is given, then it specifies the column number
(starting
  at 1) of the field that contains the value(s) to be used. In this
case, the
  first line of the CSV file MUST NOT be a header line.
* MUST be used in conjuction with (whitelist|blacklist).from_pathname.
* May be used in conjuction with (whitelist|blacklist).where_field and
  (whitelist|blacklist).where_equals.
* At most one select_field may be given per stanza.

whitelist.where_field = <field name> | <positive integer>
blacklist.where_field = <field name> | <positive integer>
* Specifies that only a subset of values are to be selected from
  (whitelist|blacklist).select_field.
* Specifies which field of the CSV file contains values to be compared
against
  for equality with the (whitelist|blacklist).where_equals values.
* Like (whitelist|blacklist).select_field, the field may be specified
by either
  name or number.  However, select_field and where_field MUST be
specified the
  same way, i.e., either BOTH by name or BOTH by number.
* MUST be used in conjuction with (whitelist|blacklist).select_field
and
  (whitelist|blacklist).where_equals.
* At most one where_field may be given per stanza.

whitelist.where_equals = <comma-separated list>
blacklist.where_equals = <comma-separated list>
* Specifies the value(s) that the value of
(whitelist|blacklist).where_field
  must equal in order to be selected via
(whitelist|blacklist).select_field.
* If more than one value is specified (separated by commas), then the
value
  of (whitelist|blacklist).where_field may equal ANY ONE of the values.
* Each value is a PCRE regular expression with the following aids for
easier
  entry:
    * You can specify simply '.' to mean '\.'
    * You can specify simply '*' to mean '.*'
* Matches are always case-insensitive; you do not need to specify the
'(?i)'
  prefix.
* MUST be used in conjuction with (whitelist|blacklist).select_field
and
  (whitelist|blacklist).where_field.
* At most one where_equals may be given per stanza.

machineTypesFilter = <comma-separated list>
```

```
* Not used unless specified.
* Boolean OR logic is employed: a match against any element in the list
  constitutes a match.
* This filter is used in boolean AND logic with white/blacklist
filters.
  Only clients which match the white/blacklist AND which match this
  machineTypesFilter will be included.
  * In other words, the match is an intersection of the matches for the
    white/blacklist and the matches for MachineTypesFilter.
* This filter can be overridden at the serverClass and serverClass:app
  levels.
* These patterns are PCRE regular expressions, with the following aids
for
  easier entry:
    * You can specify simply '.' to mean '\.'
    * You can specify simply '*' to mean '.*'
* Matches are always case-insensitive; you do not need to specify the
'(?i)'
  prefix.
* Unset by default.

restartSplunkWeb = true | false
* If true, restarts SplunkWeb on the client when a member app or a
directly
  configured app is updated.
* Can be overridden at the serverClass level and the serverClass:app
level.
* Defaults to false

restartSplunkd = true | false
* If true, restarts splunkd on the client when a member app or a
directly
  configured app is updated.
* Can be overridden at the serverClass level and the serverClass:app
level.
* Defaults to false

issueReload = true | false
* If true, triggers a reload of internal processors at the client when a
  member app or a directly configured app is updated
* If you don't want to immediately start using an app that is pushed to
a
  client, you should set this to false.
* defaults to false

restartIfNeeded = true | false
* This is only valid on forwarders that are newer than 6.4.
* If true and issueReload is also true, then when an updated app is
delpoyed
  to the client, that client will try to reload that app. If it fails,
it will
  then restart.
```

```
* defaults to false

stateOnClient = enabled | disabled | noop
* If set to "enabled", sets the application state to enabled on the
client,
  regardless of state on the deployment server.
* If set to "disabled", set the application state to disabled on the
client,
  regardless of state on the deployment server.
* If set to "noop", the state on the client will be the same as on the
  deployment server.
* Can be overridden at the serverClass level and the serverClass:app
level.
* Defaults to enabled.

precompressBundles = true | flase
* Controls whether the Deployment Server will generate both .bundle and
  .bundle.gz files. The pre-compressed files offer improved performance
as
  the DS is not required to compress the bundles on the fly for each
 client
  that it has to send the bundle to. However, this setting is only
  beneficial if there is no SSL compression in use and the client has
  support for HTTP compression.

* Deployment Server / server.conf
*   allowSslCompression = false
*   useHTTPServerCompression = true
*
* Deployment Client / server.conf
*   useHTTPClientCompression = true
*
* This option is inherited and available upto the serverclass level (not
  app). Apps belonging to server classes that required precompression
will
  be compressed, even if they belong to a server class which does not
  require precompression
* Defaults to true
```

SECOND LEVEL: serverClass ###########

```
##################################################
########### SECOND LEVEL: serverClass ###########
##################################################SECOND LEVEL:
serverClass ###########

[serverClass:<serverClassName>]
* This stanza defines a server class. A server class is a collection of
```

```
  applications; an application may belong to multiple server classes.
* serverClassName is a unique name that is assigned to this server
class.
* A server class can override all inheritable properties in the [global]
stanza.
* A server class name may only contain: letters, numbers, space,
underscore,
  dash, dot, tilde, and the '@' symbol.  It is case-sensitive.

# NOTE:
# The keys listed below are all described in detail in the
# [global] section above. They can be used with serverClass stanza to
# override the global setting
continueMatching = true | false
endpoint = <URL template string>
excludeFromUpdate = <path>[,<path>]...
filterType = whitelist | blacklist
whitelist.<n> = <clientName> | <IP address> | <hostname>
blacklist.<n> = <clientName> | <IP address> | <hostname>
machineTypesFilter = <comma-separated list>
restartSplunkWeb = true | false
restartSplunkd = true | false
issueReload = true | false
restartIfNeeded = true | false
stateOnClient = enabled | disabled | noop
repositoryLocation = <path>
```

THIRD LEVEL: app ###########

```
########################################
########### THIRD LEVEL: app ###########
########################################THIRD LEVEL: app ###########

[serverClass:<server class name>:app:<app name>]
* This stanza maps an application (which must already exist in
  repositoryLocation) to the specified server class.
* server class name - the server class to which this content should be
  added.
* app name can be '*' or the name of an app:
    * The value '*' refers to all content in the repositoryLocation,
adding
      it to this serverClass. '*' stanza cannot be mixed with named
stanzas,
      for a given server class.
    * The name of an app explicitly adds the app to a server class.
      Typically apps are named by the folders that contain them.
    * An application name, if it is not the special '*' sign explained
      directly above, may only contain: letters, numbers, space,
```

```
underscore,
      dash, dot, tilde, and the '@' symbol.  It is case-sensitive.

appFile=<file name>
* In cases where the app name is different from the file or directory
name,
  you can use this parameter to specify the file name. Supported
formats
  are: directories, .tar files, and .tgz files.

# May override higher-level settings.
issueReload = true | false
restartIfNeeded = true | false
excludeFromUpdate = <path>[,<path>]...
```

serverclass.conf.example

```
#   Version 6.5.0
#
# Example 1
# Matches all clients and includes all apps in the server class

[global]
whitelist.0=*
# whitelist matches all clients.
[serverClass:AllApps]
[serverClass:AllApps:app:*]
# a server class that encapsulates all apps in the repositoryLocation


# Example 2
# Assign server classes based on dns names.

[global]

[serverClass:AppsForOps]
whitelist.0=*.ops.yourcompany.com
[serverClass:AppsForOps:app:unix]
[serverClass:AppsForOps:app:SplunkLightForwarder]

[serverClass:AppsForDesktops]
filterType=blacklist
# blacklist everybody except the Windows desktop machines.
blacklist.0=*
whitelist.0=*.desktops.yourcompany.com
[serverClass:AppsForDesktops:app:SplunkDesktop]


# Example 3
```

```
# Deploy server class based on machine types

[global]

[serverClass:AppsByMachineType]
# Ensure this server class is matched by all clients. It is IMPORTANT
to
# have a general filter here, and a more specific filter at the app
level.
# An app is matched _only_ if the server class it is contained in was
# successfully matched!
whitelist.0=*

[serverClass:AppsByMachineType:app:SplunkDesktop]
# Deploy this app only to Windows boxes.
machineTypesFilter=windows-*

[serverClass:AppsByMachineType:app:unix]
# Deploy this app only to unix boxes - 32/64 bit.
machineTypesFilter=linux-i686, linux-x86_64

# Example 4
# Specify app update exclusion list.

[global]

# The local/ subdirectory within every app will not be touched upon
update.
excludeFromUpdate=$app_root$/local

[serverClass:MyApps]

[serverClass:MyApps:app:SpecialCaseApp]
# For the SpecialCaseApp, both the local/ and lookups/ subdirectories
will
# not be touched upon update.
excludeFromUpdate=$app_root$/local,$app_root$/lookups

# Example 5
# Control client reloads/restarts

[global]
restartSplunkd=false
restartSplunkWeb=true

# For this serverclass, we attempt to only reload the configuration
files
# within the app, if we fail to reload ie if there's a conf in the app
that
# requires a restart, the admin must restart the instance themselves
[serverClass:ReloadOnly]
issueReload=true
```

```
# This is an example of a best effort reloadable serverClass. ie we try
to
# reload the app, but if there are files that require a restart, only
then
# do we restart
[serverClass:tryReloadThenRestart]
issueReload=true
restartIfNeeded=true

# Example 6a
# Use (whitelist|blacklist) text file import.
[serverClass:MyApps]
whitelist.from_pathname = etc/system/local/clients.txt

# Example 6b
# Use (whitelist|blacklist) CSV file import to read all values from the
Client
# field (ignoring all other fields).
[serverClass:MyApps]
whitelist.select_field = Client
whitelist.from_pathname = etc/system/local/clients.csv

# Example 6c
# Use (whitelist|blacklist) CSV file import to read some values from
the Client
# field (ignoring all other fields) where ServerType is one of T1, T2,
or
# starts with dc.
[serverClass:MyApps]
whitelist.select_field = Client
whitelist.from_pathname = etc/system/local/server_list.csv
whitelist.where_field = ServerType
whitelist.where_equals = T1, T2, dc*

# Example 6d
# Use (whitelist|blacklist) CSV file import to read some values from
field 2
# (ignoring all other fields) where field 1 is one of T1, T2, or starts
with
# dc.
[serverClass:MyApps]
whitelist.select_field = 2
whitelist.from_pathname = etc/system/local/server_list.csv
whitelist.where_field = 1
whitelist.where_equals = T1, T2, dc*
```

serverclass.seed.xml.conf

The following are the spec and example files for serverclass.seed.xml.conf.

serverclass.seed.xml.conf.spec

```
#   Version 6.5.0

<!--
# This configuration is used by deploymentClient to seed a Splunk
installation with applications, at startup time.
# This file should be located in the workingDir folder defined by
deploymentclient.conf.
#
# An interesting fact - the DS -> DC communication on the wire also uses
this XML format.
-->
<?xml version="1.0"?>
<deployment name="somename">

    <!--
    # The endpoint from which all apps can be downloaded.  This value
can be overridden by serviceClass or ap declarations below.
    # In addition, deploymentclient.conf can control how this property
is used by deploymentClient - see deploymentclient.conf.spec.
    -->
    <endpoint>$deploymentServerUri$/services/streams/deployment?name=$serviceClassName$

    <!--
    # The location on the deploymentClient where all applications will
be installed. This value can be overridden by serviceClass or
    # app declarations below.
    # In addition, deploymentclient.conf can control how this property
is used by deploymentClient - see deploymentclient.conf.spec.
    -->
    <repositoryLocation>$SPLUNK_HOME/etc/apps</repositoryLocation>

    <serviceClass name="serviceClassName">
        <!--
        # The order in which this service class is processed.
        -->
        <order>N</order>

        <!--
        # DeploymentClients can also override these values using
serverRepositoryLocationPolicy and serverEndpointPolicy.
        -->
        <repositoryLocation>$SPLUNK_HOME/etc/myapps</repositoryLocation>
        <endpoint>splunk.com/spacecake/$serviceClassName$/$appName$.tgz</endpoint>
```

```
        <!--
        # Please See serverclass.conf.spec for how these properties are
used.
        -->
        <continueMatching>true</continueMatching>
        <restartSplunkWeb>false</restartSplunkWeb>
        <restartSplunkd>false</restartSplunkd>
        <stateOnClient>enabled</stateOnClient>

        <app name="appName1">
            <!--
            # Applications can override the endpoint property.
            -->
            <endpoint>splunk.com/spacecake/$appName$</endpoint>
        </app>
        <app name="appName2"/>

    </serviceClass>
</deployment>
```

serverclass.seed.xml.conf.example

```
<?xml version="1.0" encoding="UTF-8"?>
<deployment name="root">
  <serverClass name="spacecake_apps">
    <app name="app_0">
      <repositoryLocation>$SPLUNK_HOME/etc/myapps</repositoryLocation>
      <!-- Download app_0 from the given location -->
      <endpoint>splunk.com/spacecake/apps/app_0.tgz</endpoint>
    </app>
    <app name="app_1">
      <repositoryLocation>$SPLUNK_HOME/etc/myapps</repositoryLocation>
      <!-- Download app_1 from the given location -->
      <endpoint>splunk.com/spacecake/apps/app_1.tgz</endpoint>
    </app>
  </serverClass>
  <serverClass name="foobar_apps">
    <!-- construct url for each location based on the scheme below and
download each app -->
    <endpoint>foobar.com:5556/services/streams/deployment?name=$serverClassName$_$appNa
    <app name="app_0"/>
    <app name="app_1"/>
    <app name="app_2"/>
  </serverClass>
  <serverClass name="local_apps">
    <endpoint>foo</endpoint>
    <app name="app_0">
```

```
      <!-- app present in local filesystem -->
      <endpoint>file:/home/johndoe/splunk/ds/service_class_2_app_0.bundle</endpoint>
    </app>
    <app name="app_1">
      <!-- app present in local filesystem -->
      <endpoint>file:/home/johndoe/splunk/ds/service_class_2_app_1.bundle</endpoint>
    </app>
    <app name="app_2">
      <!-- app present in local filesystem -->
      <endpoint>file:/home/johndoe/splunk/ds/service_class_2_app_2.bundle</endpoint>
    </app>
  </serverClass>
</deployment>
```

setup.xml.conf

The following are the spec and example files for setup.xml.conf.

setup.xml.conf.spec

```
#   Version 6.5.0
#
#

<!--
This file describes the setup XML config and provides some examples.

setup.xml provides a Setup Screen that you provide to users to specify
configurations
for an app. The Setup Screen is available when the user first runs the
app or from the
Splunk Manager: Splunk > Manager > Apps > Actions > Set up

Place setup.xml in the app's default directory:

  $SPLUNK_HOME/etc/apps/<app>/default/setup.xml

The basic unit of work is an <input>, which is targeted to a triplet
(endpoint, entity, field) and other information used to model the data.
For example
data type, validation information, name/label, etc.

The (endpoint, entity, field attributes) identifies an object where the
input is
read/written to, for example:
```

```
   endpoint=saved/searches
   entity=MySavedSearch
   field=cron_schedule

The endpoint/entities addressing is relative to the app being
configured. Endpoint/entity can
be inherited from the outer blocks (see below how blocks work).

Inputs are grouped together within a <block> element:

(1) blocks provide an iteration concept when the referenced REST entity
is a regex

(2) blocks allow you to group similar configuration items

(3) blocks can contain <text> elements to provide descriptive text to
the user.

(4) blocks can be used to create a new entry rather than edit an
already existing one, set the
    entity name to "_new". NOTE: make sure to add the required field
'name' as
    an input.

(5) blocks cannot be nested

See examples below.


Block Node attributes:

endpoint - The REST endpoint relative to
"https://hostname:port/servicesNS/nobody/<app-name>/"
           of entities/object the block/input addresses. Generally, an
endpoint maps to a
           Splunk configuration file.

entity   - An object at the endpoint. Generally, this maps to a stanza
name in a configuration file.
          NOTE: entity names should be URI encoded.

mode     - (bulk | iter) used if the entity attribute is a regular
expression:

           o iter - (default value for mode) Iterate over all matching
entities and provide a
                    separate input field for each.
           o bulk - Update all matching entities with the same value.

           NOTE: splunk interprets '*' as the regex '.*'
```

```
eai_search - a search to filter entities returned by an endpoint. If not
specified the following
             search is used: eai:acl.app="" OR
eai:acl.app="<current-app>" This search matches
             only objects defined in the app which the setup page is
being used for.

             NOTE: if objects from another app are allowed to be
configured, any changes to those
                   objects will be stored in the current app.

enabled   - (true | false | in-windows | in-unix) whether this block is
enabled or not
            o true          - (default) this block is enabled
            o false         - block disabled
            o in-windows    - block is enabled only in windows
installations
            o in-unix       - block is enabled in non-windows
installations

Input Node Attributes:

endpoint          - see description above (inherited from block)

entity            - see description above (inherited from block)

field             - <string> the field which is being configured

old_style_disable - <bool> whether to perform entity disabling by
submitting the edited entity with the following
                     field set: disabled=1. (This is only relevant for
inputs whose field=disabled|enabled).
                     Defaults to false.

Nodes within an <input> element can display the name of the entity and
field values within the entity
on the setup screen. Specify $name$ to display the name of the entity.
Use $<field_name>$ to specify
the value of a specified field.

 -->

<setup>
  <block title="Basic stuff" endpoint="saved/searches/"
entity="foobar">
    <text> some description here </text>

    <input field="is_scheduled">
      <label>Enable Schedule for $name$</label>   <!-- this will be
rendered as "Enable Schedule for foobar" -->
      <type>bool</type>
    </input>
```

```
    <input field="cron_scheduled">
      <label>Cron Schedule</label>
      <type>text</type>
    </input>
    <input field="actions">
      <label>Select Active Actions</label>
      <type>list</type>
    </input>

    <!-- bulk update  -->
    <input entity="*" field="is_scheduled" mode="bulk">
      <label>Enable Schedule For All</label>
      <type>bool</type>
    </input>
  </block>

  <!-- iterative update in this block -->
  <block title="Configure search" endpoint="saved/eventypes/"
entity="*" mode="iter">
    <input field="search">
      <label>$name$ search</label>
      <type>string</type>
    </input>
    <input field="disabled">
      <label>disable $name$</label>
      <type>bool</type>
    </input>
  </block>

  <block title="Create a new eventtype" endpoint="saved/eventtypes/"
entity="_new">
    <input target="name">
      <label>Name</label>
      <type>text</type>
    </input>
    <input target="search">
      <label>Search</label>
      <type>text</type>
    </input>
  </block>

  <block title="Add Account Info" endpoint="storage/passwords"
entity="_new">
    <input field="name">
      <label>Username</label>
      <type>text</type>
    </input>
    <input field="password">
      <label>Password</label>
      <type>password</type>
    </input>
```

```
  </block>

  <!--  example config for "Windows setup" -->
  <block title="Collect local event logs"
endpoint="admin/win-eventlogs/" eai_search="" >
    <text>
      Splunk for Windows needs at least your local event logs to
demonstrate how to search them.
      You can always add more event logs after the initial setup in
Splunk Manager.
    </text>

    <input entity="System" field="enabled" old_style_disable="true">
      <label>Enable $name$</label>
      <type>bool</type>
    </input>
    <input entity="Security" field="enabled"  old_style_disable="true">
      <label>Enable $name$</label>
      <type>bool</type>
    </input>
    <input entity="Application" field="enabled"
 old_style_disable="true">
      <label>Enable $name$</label>
      <type>bool</type>
    </input>
  </block>

  <block title="Monitor Windows update logs"
endpoint="data/inputs/monitor">
    <text>
      If you monitor the Windows update flat-file log, Splunk for
Windows can show your patch history.
      You can also monitor other logs if you have them, such as IIS or
DHCP logs, from Data Inputs in Splunk Manager
    </text>
    <input entity="%24WINDIR%5CWindowsUpdate.log" field="enabled">
      <label>Enable $name$</label>
      <type>bool</type>
    </input>
  </block>
</setup>
```

setup.xml.conf.example

No example

source-classifier.conf

The following are the spec and example files for source-classifier.conf.

source-classifier.conf.spec

```
#   Version 6.5.0
#
# This file contains all possible options for configuring settings for
the
# file classifier in source-classifier.conf.
#
# There is a source-classifier.conf in $SPLUNK_HOME/etc/system/default/
To
# set custom configurations, place a source-classifier.conf in
# $SPLUNK_HOME/etc/system/local/.  For examples, see
# source-classifier.conf.example. You must restart Splunk to enable
# configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles


ignored_model_keywords = <space-separated list of terms>
* Terms to ignore when generating a sourcetype model.
* To prevent sourcetype "bundles/learned/*-model.xml" files from
containing
  sensitive terms (e.g. "bobslaptop") that occur very frequently in your
  data files, add those terms to ignored_model_keywords.

ignored_filename_keywords = <space-separated list of terms>
* Terms to ignore when comparing a new sourcename against a known
  sourcename, for the purpose of classifying a source.
```

source-classifier.conf.example

```
#   Version 6.5.0
#
# This file contains an example source-classifier.conf.  Use this file
```

```
to
# configure classification
# of sources into sourcetypes.
#
# To use one or more of these configurations, copy the configuration
block
# into source-classifier.conf in $SPLUNK_HOME/etc/system/local/. You
must
# restart Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

# terms to ignore when generating sourcetype model to prevent model from
# containing servernames
ignored_model_keywords = sun mon tue tues wed thurs fri sat sunday
monday tuesday wednesday thursday friday saturday jan feb mar apr may
jun jul aug sep oct nov dec january february march april may june july
august september october november december 2003 2004 2005 2006 2007 2008
2009 am pm ut utc gmt cet cest cetdst met mest metdst mez mesz eet eest
eetdst wet west wetdst msk msd ist jst kst hkt ast adt est edt cst cdt
mst mdt pst pdt cast cadt east eadt wast wadt

# terms to ignore when comparing a sourcename against a known sourcename
ignored_filename_keywords = log logs com common event events little
main message messages queue server splunk
```

sourcetypes.conf

The following are the spec and example files for sourcetypes.conf.

sourcetypes.conf.spec

```
#  Version 6.5.0
#
# NOTE: sourcetypes.conf is a machine-generated file that stores the
document
# models used by the file classifier for creating source types.

# Generally, you should not edit sourcetypes.conf, as most attributes
are
# machine generated.  However, there are two attributes which you can
change.
```

```
#
# There is a sourcetypes.conf in $SPLUNK_HOME/etc/system/default/ To
set custom
# configurations, place a sourcetypes..conf in
$SPLUNK_HOME/etc/system/local/.
# For examples, see sourcetypes.conf.example. You must restart Splunk
to enable
# configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
top of
#     the file.
#   * Each conf file should have at most one default stanza. If there
are
#     multiple default stanzas, attributes are combined. In the case of
#     multiple definitions of the same attribute, the last definition in
the
#     file wins.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.


_sourcetype = <value>
* Specifies the sourcetype for the model.
* Change this to change the model's sourcetype.
* Future sources that match the model will receive a sourcetype of this
new
  name.


_source = <value>
* Specifies the source (filename) for the model.
```

sourcetypes.conf.example

```
#   Version 6.5.0
#
# This file contains an example sourcetypes.conf.  Use this file to
configure
# sourcetype models.
#
# NOTE: sourcetypes.conf is a machine-generated file that stores the
document
# models used by the file classifier for creating source types.
#
# Generally, you should not edit sourcetypes.conf, as most attributes
are
# machine generated.  However, there are two attributes which you can
change.
#
# To use one or more of these configurations, copy the configuration
block into
# sourcetypes.conf in $SPLUNK_HOME/etc/system/local/. You must restart
Splunk
# to enable configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

#
# This is an example of a machine-generated sourcetype models for a
fictitious
# sourcetype cadcamlog.
#

[/Users/bob/logs/bnf.x5_Thu_Dec_13_15:59:06_2007_171714722]
_source = /Users/bob/logs/bnf.x5
_sourcetype = cadcamlog
L---------- = 0.096899
L-t<_EQ> = 0.016473
```

splunk-launch.conf

The following are the spec and example files for splunk-launch.conf.

splunk-launch.conf.spec

```
#   Version 6.5.0

# splunk-launch.conf contains values used at startup time, by the
splunk
# command and by windows services.
#

# Note: this conf file is different from most splunk conf files.  There
is
# only one in the whole system, located at
# $SPLUNK_HOME/etc/splunk-launch.conf; further, there are no stanzas,
# explicit or implicit.  Finally, any splunk-launch.conf files in
# etc/apps/... or etc/users/... will be ignored.


# Lines beginning with a # are considered comments and are ignored.

#*******
# Environment variables
#
# Primarily, this file simply sets environment variables to be used by
# Splunk programs.
#
# These environment variables are the same type of system environment
# variables that can be set, on unix, using:
#   bourne shells:
#       $ export ENV_VAR=value
#   c-shells:
#       % setenv ENV_VAR value
#
# or at a windows command prompt:
#   C:\> SET ENV_VAR=value
#*******

<environment_variable>=<value>

* Any desired environment variable can be set to any value.
  Whitespace is trimmed from around both the key and value.
* Environment variables set here will be available to all splunk
processes,
  barring operating system limitations.


#*******
# Specific Splunk environment settings
#
# These settings are primarily treated as environment variables, though
some
```

```
# have some additional logic (defaulting).
#
# There is no need to explicitly set any of these values in typical
# environments.
#*******

SPLUNK_HOME=<pathname>
* The comment in the auto-generated splunk-launch.conf is
informational, not
  a live setting, and does not need to be uncommented.
* Fully qualified path to the Splunk install directory.
* If unset, Splunk automatically determines the location of SPLUNK_HOME
  based on the location of the splunk CLI executable.
    * Specifically, the parent of the directory containing splunk or
splunk.exe
* Must be set if Common Criteria mode is enabled.
* NOTE: Splunk plans to submit Splunk Enterprise for Common Criteria
  evaluation. Splunk does not support using the product in Common
  Criteria mode until it has been certified by NIAP. See the "Securing
  Splunk Enterprise" manual for information on the status of Common
  Criteria certification.
* Defaults to unset.

SPLUNK_DB=<pathname>
* The comment in the auto-generated splunk-launch.conf is
informational, not
  a live setting, and does not need to be uncommented.
* Fully qualified path to the directory containing the splunk index
  directories.
* Primarily used by paths expressed in indexes.conf
* The comment in the autogenerated splunk-launch.conf is informational,
not
  a live setting, and does not need to be uncommented.
* If unset, becomes $SPLUNK_HOME/var/lib/splunk (unix) or
     %SPLUNK_HOME%\var\lib\splunk (windows)
* Defaults to unset.

SPLUNK_BINDIP=<ip address>
* Specifies an interface that splunkd and splunkweb should bind to, as
  opposed to binding to the default for the local operating system.
* If unset, Splunk makes no specific request to the operating system
when
  binding to ports/opening a listening socket.  This means it
effectively
  binds to '*'; i.e.  an unspecified bind.  The exact result of this is
  controlled by operating system behavior and configuration.
* NOTE: When using this setting you must update mgmtHostPort in
web.conf to
  match, or the command line and splunkweb will not know how to
  reach splunkd.
* For splunkd, this sets both the management port and the receiving
ports
```

```
  (from forwarders).
* Useful for a host with multiple IP addresses, either to enable
  access or restrict access; though firewalling is typically a superior
  method of restriction.
* Overrides the Splunkweb-specific
web.conf/[settings]/server.socket_host
  param; the latter is preferred when SplunkWeb behavior is the focus.
* Defaults to unset.

SPLUNK_IGNORE_SELINUX=true
* If unset (not present), Splunk on Linux will abort startup if it
detects
  it is running in an SELinux environment.  This is because in
  shipping/distribution-provided SELinux environments, Splunk will not
be
  permitted to work, and Splunk will not be able to identify clearly
why.
* This setting is useful in environments where you have configured
SELinux
  to enable Splunk to work.
* If set to any value, Splunk will launch, despite the presence of
SELinux.
* Defaults to unset.

SPLUNK_OS_USER = <string> | <nonnegative integer>
* The OS user whose privileges Splunk will adopt when running, if this
  parameter is set.
* Example: SPLUNK_OS_USER=fnietzsche, but a root login is used to start
  splunkd. Immediately upon starting, splunkd abandons root's
privileges,
  and acquires fnietzsche's privileges; any files created by splunkd
(index
  data, logs, etc.) will be consequently owned by fnietzsche.  So when
  splunkd is started next time by fnietzsche, files will be readable.
* When 'splunk enable boot-start -user <U>' is invoked, SPLUNK_OS_USER
  is set to <U> as a side effect.
* Under UNIX, username or apposite numeric UID are both acceptable;
  under Windows, only a username.

#*******
# Service/server names.
#
# These settings are considered internal, and altering them is not
# supported.
#
# Under Windows, they influence the expected name of the service; on
UNIX
# they influence the reported name of the appropriate server or daemon
# process.
#
# If you want to run multiple instances of Splunk as *services* under
# Windows, you will need to change the names below for 2nd, 3rd, ...,
```

```
# instances.  That is because the 1st instance has taken up service
names
# 'Splunkd' and 'Splunkweb', and you may not have multiple services with
# same name.
#*******

SPLUNK_SERVER_NAME=<name>
* Names the splunkd server/service.
* Defaults to splunkd (UNIX), or Splunkd (Windows).

SPLUNK_WEB_NAME=<name>
* Names the Python app server / web server/service.
* Defaults to splunkweb (UNIX), or Splunkweb (Windows).
```

splunk-launch.conf.example

```
No example
```

tags.conf

The following are the spec and example files for tags.conf.

tags.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attribute/value pairs for configuring
tags.  Set
# any number of tags for indexed or extracted fields.
#
# There is no tags.conf in $SPLUNK_HOME/etc/system/default/.  To set
custom
# configurations, place a tags.conf in $SPLUNK_HOME/etc/system/local/.
For
# help, see tags.conf.example.  You must restart Splunk to enable
# configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

[<fieldname>=<value>]

```
[<fieldname>=<value>]
* The field name and value to which the tags in the stanza
  apply ( eg host=localhost ).
* A tags.conf file can contain multiple stanzas. It is recommended that
the
  value be URL encoded to avoid
* config file parsing errors especially if the field value contains the
  following characters: \n, =, []
* Each stanza can refer to only one field=value

<tag1> = <enabled|disabled>
<tag2> = <enabled|disabled>
<tag3> = <enabled|disabled>
* Set whether each <tag> for this specific <fieldname><value> is
enabled or
  disabled.
* While you can have multiple tags in a stanza (meaning that multiple
tags are
  assigned to the same field/value combination), only one tag is allowed
per
  stanza line. In other words, you can't have a list of tags on one line
of the
  stanza.

* WARNING: Do not quote the <tag> value: foo=enabled, not "foo"=enabled.
```

tags.conf.example

```
#   Version 6.5.0
#
# This is an example tags.conf.  Use this file to define tags for
fields.
#
# To use one or more of these configurations, copy the configuration
block into
# tags.conf in $SPLUNK_HOME/etc/system/local/. You must restart Splunk
to
# enable configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
#
```

```
# This first example presents a situation where the field is "host" and
the
# three hostnames for which tags are being defined are "hostswitch,"
# "emailbox," and "devmachine." Each hostname has two tags applied to
it, one
# per line. Note also that the "building1" tag has been applied to two
hostname
# values (emailbox and devmachine).

[host=hostswitch]
pci = enabled
cardholder-dest = enabled

[host=emailbox]
email = enabled
building1 = enabled

[host=devmachine]
development = enabled
building1 = enabled

[src_ip=192.168.1.1]
firewall = enabled

[seekPtr=1cb58000]
EOF = enabled
NOT_EOF = disabled
```

telemetry.conf

The following are the spec and example files for telemetry.conf.

telemetry.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attributes and values for configuring
global
# telemetry settings. Please note that enabling these settings would
enable
# apps to collect telemetry data about app usage and other properties.
#
# There is no global, default telemetry.conf. Instead, a telemetry.conf
may
# exist in each app in Splunk Enterprise.
#
```

```
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#  * You can also define global settings outside of any stanza, at the
top
#    of the file.
#  * Each conf file should have at most one default stanza. If there
are
#    multiple default stanzas, attributes are combined. In the case of
#    multiple definitions of the same attribute, the last definition in
the
#    file wins.
#  * If an attribute is defined at both the global level and in a
specific
#    stanza, the value in the specific stanza takes precedence.
```

[general]

```
[general]
sendLicenseUsage = true|false
* Send the licensing usage information of splunk/app to the app owner
* Defaults to false

sendAnonymizedUsage = true|false
* Send the anonymized usage information about various categories like
  infrastructure, utilization etc of splunk/app to the app owner
* Defaults to false

precheckSendAnonymizedUsage = true|false
* Default value for sending anonymized usage in opt in modal
* Defaults to false

precheckSendLicenseUsage = true|false
* Default value for sending license usage in opt in modal
* Defaults to true

showOptInModal = true|false
* Shows the opt in modal. DO NOT SET! When a user opts in, it will
  automatically be set to false to not show the modal again.
* Defaults to true
```

```
deprecatedConfig = true|false
* Setting to determine whether the splunk deployment is following
  best practices for the platform as well as the app
* Defaults to false

precheckSendLicenseUsage = true|false
* Default value for sending license usage in opt in modal
* Defaults to true

precheckSendAnonymizedUsage = true|false
* Default value for sending anonymized usage in opt in modal
* Defaults to false

retryTransaction = <string>
* Setting that is created if the telemetry conf updates cannot be
delivered to
  the cluster master for the splunk_instrumentation app.
* Defaults to an empty string
```

telemetry.conf.example

```
#   Version 6.5.0
#
# This file contains possible attributes and values for configuring
global
# telemetry settings. Please note that enabling these settings would
enable
# apps to collect telemetry data about app usage and other properties.
#
# There is no global, default telemetry.conf. Instead, a telemetry.conf
may
# exist in each app in Splunk Enterprise.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

[general]
sendLicenseUsage = false
sendAnonymizedUsage = false
precheckSendAnonymizedUsage = false
precheckSendLicenseUsage = true
showOptInModal = true
deprecatedConfig = false
```

times.conf

The following are the spec and example files for times.conf.

times.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attribute/value pairs for creating custom
time
# ranges.
#
# To set custom configurations, place a times.conf in
# $SPLUNK_HOME/etc/system/local/.  For help, see times.conf.example.
You
# must restart Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
top
#     of the file.
#   * Each conf file should have at most one default stanza. If there
are
#     multiple default stanzas, attributes are combined. In the case of
#     multiple definitions of the same attribute, the last definition in
the
#     file wins.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.
```

[<timerange_name>]

```
[<timerange_name>]
* The token to be used when accessing time ranges via the API or
```

```
command
  line
* A times.conf file can contain multiple stanzas.

label = <string>
* The textual description used by the UI to reference this time range
* Required

header_label = <string>
* The textual description used by the UI when displaying search results
in
  this time range.
* Optional.  If omitted, the <timerange_name> is used instead.

earliest_time = <string>
* The string that represents the time of the earliest event to return,
  inclusive.
* The time can be expressed with a relative time identifier or in epoch
time.
* Optional.  If omitted, no earliest time bound is used.

latest_time = <string>
* The string that represents the time of the earliest event to return,
  inclusive.
* The time can be expressed with a relative time identifier or in epoch
  time.
* Optional.  If omitted, no latest time bound is used.  NOTE: events
that
  occur in the future (relative to the server timezone) may be
returned.

order = <integer>
* The key on which all custom time ranges are sorted, ascending.
* The default time range selector in the UI will merge and sort all time
  ranges according to the 'order' key, and then alphabetically.
* Optional.  Default value is 0.

sub_menu = <submenu name>
* If present, the time range is to be shown in the given submenu instead
  of in the main menu.
* The value for this key must be the label key of an existing stanza
name,
  and that stanza name must have an is_sub_menu = True key
* Optional. If omitted the given time option will display in the main
menu.

is_sub_menu = <boolean>
* If True, the given item is only the 'opener' element for a submenu.
* Stanzas containing this key can still be assigned an order value to
set
  the placement within the main menu, but can not themselves have
  latest_time nor earliest_time keys.
```

times.conf.example

```
#   Version 6.5.0
#
# This is an example times.conf.  Use this file to create custom time
ranges
# that can be used while interacting with the search system.
#
# To use one or more of these configurations, copy the configuration
block
# into times.conf in $SPLUNK_HOME/etc/system/local/. You must restart
Splunk
# to enable configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

# Note: These are examples.  Replace the values with your own
customizations.


# The stanza name is an alphanumeric string (no spaces) that uniquely
# identifies a time range.
[this_business_week]

# Define the label used in the time range control
label = This business week

# Define the label to be used in display headers. If omitted the 'label'
key
# will be used with the first letter lowercased.
header_label = during this business week
earliest_time = +1d@w1
latest_time = +6d@w6

# Define the ordering sequence of this time range.  All time ranges are
# sorted numerically, ascending. If the time range is in a sub menu and
not
# in the main menu, this will determine the position within the sub
menu.
order = 110


# a time range that only has a bound on the earliest time
#
```

```
[last_3_hours]
label = Last 3 hours
header_label = in the last 3 hours
earliest_time = -3h
order = 30


# Use epoch time notation to define the time bounds for the Fall
Semester
# 2013, where earliest_time is 9/4/13 00:00:00 and latest_time is
12/13/13
# 00:00:00.
#
[Fall_2013]
label = Fall Semester 2013
earliest_time = 1378278000
latest_time = 1386921600


# two time ranges that should appear in a sub menu instead of in the
main
# menu.  the order values here determine relative ordering within the
# submenu.
#
[yesterday]
label = Yesterday
earliest_time = -1d@d
latest_time = @d
order = 10
sub_menu = Other options

[day_before_yesterday]
label = Day before yesterday
header_label = from the day before yesterday
earliest_time = -2d@d
latest_time = -1d@d
order = 20
sub_menu = Other options


#
# The sub menu item that should contain the previous two time ranges.
 The
# order key here determines the submenu opener's placement within the
main
# menu.
#
[other]
label = Other options
order = 202
```

transactiontypes.conf

The following are the spec and example files for transactiontypes.conf.

transactiontypes.conf.spec

```
#   Version 6.5.0
#
# This file contains all possible attributes and value pairs for a
# transactiontypes.conf file.  Use this file to configure transaction
searches
# and their properties.
#
# There is a transactiontypes.conf in $SPLUNK_HOME/etc/system/default/.
 To set
# custom configurations, place a transactiontypes.conf in
# $SPLUNK_HOME/etc/system/local/. You must restart Splunk to enable
# configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
top of
#     the file.
#   * Each conf file should have at most one default stanza. If there
are
#     multiple default stanzas, attributes are combined. In the case of
#     multiple definitions of the same attribute, the last definition in
the
#     file wins.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.


[<TRANSACTIONTYPE>]
```

```
* Create any number of transaction types, each represented by a stanza
name and
  any number of the following attribute/value pairs.
* Use the stanza name, [<TRANSACTIONTYPE>], to search for the
transaction in
  Splunk Web.
* If you do not specify an entry for each of the following attributes,
Splunk
  uses the default value.

maxspan = [<integer> s|m|h|d|-1]
* Set the maximum time span for the transaction.
* Can be in seconds, minutes, hours, or days, or -1 for an unlimited
timespan.
  * For example:  5s, 6m, 12h or 30d.
* Defaults to: maxspan=-1

maxpause = [<integer> s|m|h|d|-1]
* Set the maximum pause between the events in a transaction.
* Can be in seconds, minutes, hours, or days, or -1 for an unlimited
pause.
  * For example:  5s, 6m, 12h or 30d.
* Defaults to: maxpause=-1

maxevents = <integer>
* The maximum number of events in a transaction. This constraint is
disabled if
  the value is a negative integer.
* Defaults to: maxevents=1000

fields = <comma-separated list of fields>
* If set, each event must have the same field(s) to be considered part
of the
  same transaction.
  * For example: fields=host,cookie
* Defaults to: ""

connected=[true|false]
* Relevant only if fields (see above) is not empty. Controls whether an
event
  that is not inconsistent and not consistent with the fields of a
transaction
  opens a new transaction (connected=true) or is added to the
transaction.
* An event can be not inconsistent and not field-consistent if it
contains
  fields required by the transaction but none of these fields has been
  instantiated in the transaction (by a previous event addition).
* Defaults to: connected=true

startswith=<transam-filter-string>
* A search or eval filtering expression which, if satisfied by an
```

```
event, marks
  the beginning of a new transaction.
* For example:
  * startswith="login"
  * startswith=(username=foobar)
  * startswith=eval(speed_field < max_speed_field)
  * startswith=eval(speed_field < max_speed_field/12)
* Defaults to: ""

endswith=<transam-filter-string>
* A search or eval filtering expression which, if satisfied by an
event, marks
  the end of a transaction.
* For example:
  * endswith="logout"
  * endswith=(username=foobar)
  * endswith=eval(speed_field > max_speed_field)
  * endswith=eval(speed_field > max_speed_field/12)
* Defaults to: ""

* For startswith/endswith <transam-filter-string> has the following
syntax:
* syntax:   "<search-expression>" | (<quoted-search-expression>) |
eval(<eval-expression>)
* Where:
  * <search-expression>        is a valid search expression that does
not contain quotes
  * <quoted-search-expression> is a valid search expression that
contains quotes
  * <eval-expression>          is a valid eval expression that
evaluates to a boolean. For example,
    startswith=eval(foo<bar*2) will match events where foo is less than
2 x bar.
* Examples:
  * "<search expression>":       startswith="foo bar"
  * <quoted-search-expression>:  startswith=(name="mildred")
  * <quoted-search-expression>:  startswith=("search literal")
  * eval(<eval-expression>):     startswith=eval(distance/time <
max_speed)

### memory constraint options ###

maxopentxn=<int>
* Specifies the maximum number of not yet closed transactions to keep in
the
  open pool. When this limit is surpassed, Splunk begins evicting
transactions
  using LRU (least-recently-used memory cache algorithm) policy.
* The default value of this attribute is read from the transactions
stanza in
  limits.conf.
```

```
maxopenevents=<int>
* Specifies the maximum number of events (can be) part of open
transactions.
  When this limit is surpassed, Splunk begins evicting transactions
using LRU
  (least-recently-used memory cache algorithm) policy.
* The default value of this attribute is read from the transactions
stanza in
  limits.conf.

keepevicted=<bool>
* Whether to output evicted transactions. Evicted transactions can be
  distinguished from non-evicted transactions by checking the value of
the
  'evicted' field, which is set to '1' for evicted transactions.
* Defaults to: keepevicted=false

### multivalue rendering options ###

mvlist=<bool>|<field-list>
* Field controlling whether the multivalued fields of the transaction
are (1) a
  list of the original events ordered in arrival order or (2) a set of
unique
  field values ordered lexigraphically. If a comma/space delimited list
of
  fields is provided only those fields are rendered as lists
* Defaults to: mvlist=f

delim=<string>
* A string used to delimit the original event values in the transaction
event
  fields.
* Defaults to: delim=" "

nullstr=<string>
* The string value to use when rendering missing field values as part of
mv
  fields in a transaction.
* This option applies only to fields that are rendered as lists.
* Defaults to: nullstr=NULL

### values only used by the searchtxn search command ###

search=<string>
* A search string used to more efficiently seed transactions of this
type.
* The value should be as specific as possible, to limit the number of
events
  that must be retrieved to find transactions.
* Example: sourcetype="sendmaill_sendmail"
* Defaults to "*" (all events)
```

transactiontypes.conf.example

```
#   Version 6.5.0
#
# This is an example transactiontypes.conf.  Use this file as a
template to
# configure transactions types.
#
# To use one or more of these configurations, copy the configuration
block into
# transactiontypes.conf in $SPLUNK_HOME/etc/system/local/.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

[default]
maxspan = 5m
maxpause = 2s
match = closest

[purchase]
maxspan  = 10m
maxpause = 5m
fields  = userid
```

transforms.conf

The following are the spec and example files for transforms.conf.

transforms.conf.spec

```
#   Version 6.5.0
#
# This file contains attributes and values that you can use to configure
# data transformations.  and event signing in transforms.conf.
#
# Transforms.conf is commonly used for:
# * Configuring regex-based host and source type overrides.
```

```
# * Anonymizing certain types of sensitive incoming data, such as credit
#   card or social security numbers.
# * Routing specific events to a particular index, when you have
multiple
#   indexes.
# * Creating new index-time field extractions. NOTE: We do not recommend
#   adding to the set of fields that are extracted at index time unless
it
#   is absolutely necessary because there are negative performance
#   implications.
# * Creating advanced search-time field extractions that involve one or
more
#   of the following:
#   * Reuse of the same field-extracting regular expression across
multiple
#     sources, source types, or hosts.
#   * Application of more than one regex to the same source, source
type, or
#     host.
#   * Using a regex to extract one or more values from the values of
another
#     field.
#   * Delimiter-based field extractions (they involve field-value pairs
that
#     are separated by commas, colons, semicolons, bars, or something
#     similar).
#   * Extraction of multiple values for the same field (multivalued
field
#     extraction).
#   * Extraction of fields with names that begin with numbers or
#     underscores.
#   * NOTE: Less complex search-time field extractions can be set up
#           entirely in props.conf.
# * Setting up lookup tables that look up fields from external sources.
#
# All of the above actions require corresponding settings in
props.conf.
#
# You can find more information on these topics by searching the Splunk
# documentation (http://docs.splunk.com/Documentation)
#
# There is a transforms.conf file in $SPLUNK_HOME/etc/system/default/.
To
# set custom configurations, place a transforms.conf
# $SPLUNK_HOME/etc/system/local/. For examples, see the
# transforms.conf.example file.
#
# You can enable configurations changes made to transforms.conf by
typing
# the following search string in Splunk Web:
#
# | extract reload=t
```

```
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#  * You can also define global settings outside of any stanza, at the
top
#    of the file.
#  * Each conf file should have at most one default stanza. If there
are
#    multiple default stanzas, attributes are combined. In the case of
#    multiple definitions of the same attribute, the last definition in
the
#    file wins.
#  * If an attribute is defined at both the global level and in a
specific
#    stanza, the value in the specific stanza takes precedence.


[<unique_transform_stanza_name>]
* Name your stanza. Use this name when you configure field extractions,
  lookup tables, and event routing in props.conf. For example, if you
are
  setting up an advanced search-time field extraction, in props.conf you
  would add REPORT-<class> = <unique_transform_stanza_name> under the
  [<spec>] stanza that corresponds with a stanza you've created in
  transforms.conf.
* Follow this stanza name with any number of the following
attribute/value
  pairs, as appropriate for what you intend to do with the transform.
* If you do not specify an entry for each attribute, Splunk uses the
default
  value.

REGEX = <regular expression>
* Enter a regular expression to operate on your data.
* NOTE: This attribute is valid for both index-time and search-time
field
  extraction.
* REGEX is required for all search-time transforms unless you are
setting up
  a delimiter-based field extraction, in which case you use DELIMS (see
the
```

```
  DELIMS attribute description, below).
* REGEX is required for all index-time transforms.
* REGEX and the FORMAT attribute:
  * Name-capturing groups in the REGEX are extracted directly to fields.
    This means that you do not need to specify the FORMAT attribute for
    simple field extraction cases (see the description of FORMAT,
below).
  * If the REGEX extracts both the field name and its corresponding
field
    value, you can use the following special capturing groups if you
want to
    skip specifying the mapping in FORMAT:
      _KEY_<string>, _VAL_<string>.
  * For example, the following are equivalent:
    * Using FORMAT:
           * REGEX  = ([a-z]+)=([a-z]+)
           * FORMAT = $1::$2
    * Without using FORMAT
           * REGEX  = (?<_KEY_1>[a-z]+)=(?<_VAL_1>[a-z]+)
    * When using either of the above formats, in a search-time
extraction,
      the regex will continue to match against the source text,
extracting
      as many fields as can be identified in the source text.
* Defaults to an empty string.

FORMAT = <string>
* NOTE: This option is valid for both index-time and search-time field
extraction. However, FORMAT
  behaves differently depending on whether the extraction is performed
at index time or
  search time.
* This attribute specifies the format of the event, including any field
names or values you want
  to add.
* FORMAT for index-time extractions:
  * Use $n (for example $1, $2, etc) to specify the output of each REGEX
    match.
  * If REGEX does not have n groups, the matching fails.
  * The special identifier $0 represents what was in the DEST_KEY before
the
    REGEX was performed.
  * At index time only, you can use FORMAT to create concatenated
fields:
    * Example: FORMAT = ipaddress::$1.$2.$3.$4
  * When you create concatenated fields with FORMAT, "$" is the only
special
    character. It is treated as a prefix for regex-capturing groups only
if
    it is followed by a number and only if the number applies to an
existing
    capturing group. So if REGEX has only one capturing group and its
```

```
value
    is "bar", then:
      * "FORMAT = foo$1" yields "foobar"
      * "FORMAT = foo$bar" yields "foo$bar"
      * "FORMAT = foo$1234" yields "foo$1234"
      * "FORMAT = foo$1\$2" yields "foobar\$2"
  * At index-time, FORMAT defaults to <stanza-name>::$1
* FORMAT for search-time extractions:
  * The format of this field as used during search time extractions is
as
    follows:
    * FORMAT = <field-name>::<field-value>(
<field-name>::<field-value>)*
      where:
      * field-name  = [<string>|$<extracting-group-number>]
      * field-value = [<string>|$<extracting-group-number>]
  * Search-time extraction examples:
      * 1. FORMAT = first::$1 second::$2 third::other-value
      * 2. FORMAT = $1::$2
  * If the key-name of a FORMAT setting is varying, for example $1 in
the
    example 2 just above, then the regex will continue to match against
the
    source key to extract as many matches as are present in the text.
  * NOTE: You cannot create concatenated fields with FORMAT at search
time.
    That functionality is only available at index time.
  * At search-time, FORMAT defaults to an empty string.

CLONE_SOURCETYPE = <string>
* This name is wrong; a transform with this setting actually clones and
  modifies events, and assigns the new events the specified sourcetype.

* If CLONE_SOURCETYPE is used as part of a transform, the transform will
  create a modified duplicate event, for all events that the transform
is
  applied to via normal props.conf rules.
* Use this feature if you need to store both the original and a
modified
  form of the data in your system, or if you want to send the original
and a
  modified form to different outbound systems.
  * A typical example would be to retain sensitive information according
to
    one policy and a version with the sensitive information removed
    according to another policy.  For example, some events may have data
    that you must retain for 30 days (such as personally identifying
    information) and only 30 days with restricted access, but you need
that
    event retained without the sensitive data for a longer time with
wider
    access.
```

```
* Specifically, for each event handled by this transform, a near-exact
copy
  is made of the original event, and the transformation is applied to
the
  copy.  The original event will continue along normal data processing
  unchanged.
* The <string> used for CLONE_SOURCETYPE selects the sourcetype that
will be
  used for the duplicated events.
* The new sourcetype MUST differ from the the original sourcetype.  If
the
  original sourcetype is the same as the target of the
CLONE_SOURCETYPE,
  Splunk will make a best effort to log warnings to splunkd.log, but
this
  setting will be silently ignored at runtime for such cases, causing
the
  transform to be applied to the original event without cloning.
* The duplicated events will receive index-time transformations & sed
  commands all transforms which match its new host/source/sourcetype.
  * This means that props matching on host or source will incorrectly
be
    applied a second time. (SPL-99120)
* Can only be used as part of of an otherwise-valid index-time
transform.  For
  example REGEX is required, there must be a valid target (DEST_KEY or
  WRITE_META), etc as above.

LOOKAHEAD = <integer>
* NOTE: This option is valid for all index time transforms, such as
  index-time field creation, or DEST_KEY modifications.
* Optional. Specifies how many characters to search into an event.
* Defaults to 4096.
* You may want to increase this value if you have event line lengths
that
  exceed 4096 characters (before linebreaking).

WRITE_META = [true|false]
* NOTE: This attribute is only valid for index-time field extractions.
* Automatically writes REGEX to metadata.
* Required for all index-time field extractions except for those where
  DEST_KEY = _meta (see the description of the DEST_KEY attribute,
below)
* Use instead of DEST_KEY = _meta.
* Defaults to false.

DEST_KEY = <KEY>
* NOTE: This attribute is only valid for index-time field extractions.
* Specifies where Splunk stores the expanded FORMAT results in
accordance
  with the REGEX match.
* Required for index-time field extractions where WRITE_META = false or
```

```
is
  not set.
* For index-time extractions, DEST_KEY can be set to a number of values
  mentioned in the KEYS section at the bottom of this file.
  * If DEST_KEY = _meta (not recommended) you should also add $0 to the
    start of your FORMAT attribute.  $0 represents the DEST_KEY value
before
    Splunk performs the REGEX (in other words, _meta).
    * The $0 value is in no way derived *from* the REGEX match. (It
      does not represent a captured group.)
* KEY names are case-sensitive, and should be used exactly as they
appear in
  the KEYs list at the bottom of this file. (For example, you would say
  DEST_KEY = MetaData:Host, *not* DEST_KEY = metadata:host .)

DEFAULT_VALUE = <string>
* NOTE: This attribute is only valid for index-time field extractions.
* Optional. Splunk writes the DEFAULT_VALUE to DEST_KEY if the REGEX
fails.
* Defaults to empty.

SOURCE_KEY = <string>
* NOTE: This attribute is valid for both index-time and search-time
field
  extractions.
* Optional. Defines the KEY that Splunk applies the REGEX to.
* For search time extractions, you can use this attribute to extract
one or
  more values from the values of another field. You can use any field
that
  is available at the time of the execution of this field extraction
* For index-time extractions use the KEYs described at the bottom of
this
  file.
  * KEYs are case-sensitive, and should be used exactly as they appear
in
    the KEYs list at the bottom of this file. (For example, you would
say
    SOURCE_KEY = MetaData:Host, *not* SOURCE_KEY = metadata:host .)
* If <string> starts with "field:" or "fields:" the meaning is changed.
  Instead of looking up a KEY, it instead looks up an already indexed
field.
  For example, if a CSV field name "price" was indexed then
  "SOURCE_KEY = field:price" causes the REGEX to match against the
contents
  of that field.  It's also possible to list multiple fields here with
  "SOURCE_KEY = fields:name1,name2,name3" which causes MATCH to be run
  against a string comprising of all three values, separated by space
  characters.
* SOURCE_KEY is typically used in conjunction with REPEAT_MATCH in
  index-time field transforms.
* Defaults to _raw, which means it is applied to the raw, unprocessed
```

```
text
  of all events.

REPEAT_MATCH = [true|false]
* NOTE: This attribute is only valid for index-time field extractions.
* Optional. When set to true Splunk runs the REGEX multiple times on the
  SOURCE_KEY.
* REPEAT_MATCH starts wherever the last match stopped, and continues
until
  no more matches are found. Useful for situations where an unknown
number
  of REGEX matches are expected per event.
* Defaults to false.

DELIMS = <quoted string list>
* NOTE: This attribute is only valid for search-time field extractions.
* IMPORTANT: If a value may contain an embedded unescaped double quote
  character, such as "foo"bar", use REGEX, not DELIMS. An escaped double
  quote (\") is ok.
* Optional. Used in place of REGEX when dealing with delimiter-based
field
  extractions, where field values (or field/value pairs) are separated
by
  delimiters such as colons, spaces, line breaks, and so on.
* Sets delimiter characters, first to separate data into field/value
pairs,
  and then to separate field from value.
* Each individual character in the delimiter string is used as a
delimiter
  to split the event.
* Delimiters must be quoted with " " (use \ to escape).
* When the event contains full delimiter-separated field/value pairs,
you
  enter two sets of quoted characters for DELIMS:
* The first set of quoted delimiters extracts the field/value pairs.
* The second set of quoted delimiters separates the field name from its
  corresponding value.
* When the event only contains delimiter-separated values (no field
names)
  you use just one set of quoted delimiters to separate the field
values.
  Then you use the FIELDS attribute to apply field names to the
extracted
  values (see FIELDS, below).
  * Alternately, Splunk reads even tokens as field names and odd tokens
as
    field values.
* Splunk consumes consecutive delimiter characters unless you specify a
list
  of field names.
* The following example of DELIMS usage applies to an event where
  field/value pairs are separated by '|' symbols and the field names are
```

```
  separated from their corresponding values by '=' symbols:
    [pipe_eq]
    DELIMS = "|", "="
* Defaults to "".

FIELDS = <quoted string list>
* NOTE: This attribute is only valid for search-time field extractions.
* Used in conjunction with DELIMS when you are performing
delimiter-based
  field extraction and only have field values to extract.
* FIELDS enables you to provide field names for the extracted field
values,
  in list format according to the order in which the values are
extracted.
* NOTE: If field names contain spaces or commas they must be quoted
with " "
        (to escape, use \).
* The following example is a delimiter-based field extraction where
three
  field values appear in an event. They are separated by a comma and
then a
  space.
    [commalist]
    DELIMS = ", "
    FIELDS = field1, field2, field3
* Defaults to "".

MV_ADD = [true|false]
* NOTE: This attribute is only valid for search-time field extractions.
* Optional. Controls what the extractor does when it finds a field which
  already exists.
* If set to true, the extractor makes the field a multivalued field and
  appends the newly found value, otherwise the newly found value is
  discarded.
* Defaults to false

CLEAN_KEYS = [true|false]
* NOTE: This attribute is only valid for search-time field extractions.
* Optional. Controls whether Splunk "cleans" the keys (field names) it
  extracts at search time.
  "Key cleaning" is the practice of replacing any non-alphanumeric
  characters (characters other than those falling between the a-z, A-Z,
or
  0-9 ranges) in field names with underscores, as well as the stripping
of
  leading underscores and 0-9 characters from field names.
* Add CLEAN_KEYS = false to your transform if you need to extract field
  names that include non-alphanumeric characters, or which begin with
  underscores or 0-9 characters.
* Defaults to true.

KEEP_EMPTY_VALS = [true|false]
```

```
* NOTE: This attribute is only valid for search-time field extractions.
* Optional. Controls whether Splunk keeps field/value pairs when the
value
  is an empty string.
* This option does not apply to field/value pairs that are generated by
  Splunk's autokv extraction. Autokv ignores field/value pairs with
empty
  values.
* Defaults to false.

CAN_OPTIMIZE = [true|false]
* NOTE: This attribute is only valid for search-time field extractions.
* Optional. Controls whether Splunk can optimize this extraction out
  (another way of saying the extraction is disabled).
* You might use this if you're running searches under a Search Mode
setting
  that disables field discovery--it ensures that Splunk *always*
discovers
  specific fields.
* Splunk only disables an extraction if it can determine that none of
the
  fields identified by the extraction will ever be needed for the
successful
  evaluation of a search.
* NOTE: This option should be rarely set to false.
* Defaults to true.
```

Lookup tables

```
#*******
# Lookup tables
#*******Lookup tables
# NOTE: Lookup tables are used ONLY during search time

filename = <string>
* Name of static lookup file.
* File should be in $SPLUNK_HOME/etc/<app_name>/lookups/ for some
<app_name>, or in
  $SPLUNK_HOME/etc/system/lookups/
* If file is in multiple 'lookups' directories, no layering is done.
* Standard conf file precedence is used to disambiguate.
* Defaults to empty string.

collection = <string>
* Name of the collection to use for this lookup.
* Collection should be defined in
$SPLUNK_HOME/etc/<app_name>/collections.conf
  for some <app_name>
```

```
* If collection is in multiple collections.conf file, no layering is
done.
* Standard conf file precedence is used to disambiguate.
* Defaults to empty string (in which case the name of the stanza is
used).

max_matches = <integer>
* The maximum number of possible matches for each input lookup value
  (range 1 - 1000).
* If the lookup is non-temporal (not time-bounded, meaning the
time_field
  attribute is not specified), Splunk uses the first <integer> entries,
in
  file order.
* If the lookup is temporal, Splunk uses the first <integer> entries in
  descending time order.  In other words, up <max_matches> lookup
entries
  will be allowed to match, and if more than this many the ones nearest
to
  the lookup value will be used.
* Default = 1000 if the lookup is not temporal, default = 1 if it is
  temporal.

min_matches = <integer>
* Minimum number of possible matches for each input lookup value.
* Default = 0 for both temporal and non-temporal lookups, which means
that
  Splunk outputs nothing if it cannot find any matches.
* However, if min_matches > 0, and Splunk get less than min_matches,
then
  Splunk provides the default_match value provided (see below).

default_match = <string>
* If min_matches > 0 and Splunk has less than min_matches for any given
  input, it provides this default_match value one or more times until
the
  min_matches threshold is reached.
* Defaults to empty string.

case_sensitive_match = <bool>
* NOTE: This attribute is not valid for KV Store-based lookups.
* If set to false, case insensitive matching will be performed for all
  fields in a lookup table
* Defaults to true (case sensitive matching)

match_type = <string>
* A comma and space-delimited list of <match_type>(<field_name>)
  specification to allow for non-exact matching
* The available match_type values are WILDCARD, CIDR, and EXACT.  EXACT
is
  the default and does not need to be specified.  Only fields that
should
```

```
  use WILDCARD or CIDR matching should be specified in this list

external_cmd = <string>
* Provides the command and arguments to invoke to perform a lookup. Use
this
  for external (or "scripted") lookups, where you interface with with an
  external script rather than a lookup table.
* This string is parsed like a shell command.
* The first argument is expected to be a python script (or executable
file)
  located in $SPLUNK_HOME/etc/<app_name>/bin (or ../etc/searchscripts).
* Presence of this field indicates that the lookup is external and
command
  based.
* Defaults to empty string.

fields_list = <string>
* A comma- and space-delimited list of all fields that are supported by
the
  external command.

external_type = [python|executable|kvstore|geo]
* Type of external command.
* "python" a python script
* "executable" a binary executable
* "geo" a point-in-polygon lookup
* Defaults to "python".

time_field = <string>
* Used for temporal (time bounded) lookups. Specifies the name of the
field
  in the lookup table that represents the timestamp.
* Defaults to an empty string, meaning that lookups are not temporal by
  default.

time_format = <string>
* For temporal lookups this specifies the 'strptime' format of the
timestamp
  field.
* You can include subseconds but Splunk will ignore them.
* Defaults to %s.%Q or seconds from unix epoch in UTC an optional
milliseconds.

max_offset_secs = <integer>
* For temporal lookups, this is the maximum time (in seconds) that the
event
  timestamp can be later than the lookup entry time for a match to
occur.
* Default is 2000000000 (no maximum, effectively).

min_offset_secs = <integer>
* For temporal lookups, this is the minimum time (in seconds) that the
```

```
event
  timestamp can be later than the lookup entry timestamp for a match to
  occur.
* Defaults to 0.

batch_index_query = <bool>
* For large file based lookups, this determines whether queries can be
  grouped to improve search performance.
* Default is unspecified here, but defaults to true (at global level in
  limits.conf)

allow_caching = <bool>
* Allow output from lookup scripts to be cached
* Default is true

max_ext_batch = <integer>
* The maximum size of external batch (range 1 - 1000).
* Only used with kvstore.
* Default = 300.

filter = <string>
* Filter results from the lookup table before returning data. Create
this filter
  like you would a typical search query using Boolean expressions
and/or comparison operators.
* For KV Store lookups, filtering is done when data is initially
retrieved to improve performance.
* For CSV lookups, filtering is done in memory.

feature_id_element = <string>
* If lookup file is a kmz file, this field can be used to specify the
xml path from
  placemark down to the name of this placemark.
* Default = /Placemark/name
* ONLY for Kmz files
```

KEYS:

```
#*******
# KEYS:
#*******KEYS:
* NOTE: Keys are case-sensitive. Use the following keys exactly as they
        appear.

queue : Specify which queue to send the event to (can be nullQueue,
indexQueue).
        * indexQueue is the usual destination for events going through
the
          transform-handling processor.
```

```
        * nullQueue is a destination which will cause the events to be
          dropped entirely.
_raw  : The raw text of the event.
_meta : A space-separated list of metadata for an event.
_time : The timestamp of the event, in seconds since 1/1/1970 UTC.

MetaData:Host       : The host associated with the event.
                      The value must be prefixed by "host::"

_MetaData:Index     : The index where the event should be stored.

MetaData:Source     : The source associated with the event.
                      The value must be prefixed by "source::"

MetaData:Sourcetype : The sourcetype of the event.
                      The value must be prefixed by "sourcetype::"

_TCP_ROUTING        : Comma separated list of tcpout group names (from
outputs.conf)
                      Defaults to groups present in 'defaultGroup' for
[tcpout].

_SYSLOG_ROUTING     : Comma separated list of syslog-stanza  names
(from outputs.conf)
                      Defaults to groups present in 'defaultGroup' for
[syslog].

* NOTE: Any KEY (field name) prefixed by '_' is not indexed by Splunk,
in general.


[accepted_keys]

<name> = <key>

* Modifies Splunk's list of key names it considers valid when
automatically
  checking your transforms for use of undocumented SOURCE_KEY or
DEST_KEY
  values in index-time transformations.
* By adding entries to [accepted_keys], you can tell Splunk that a key
that
  is not documented is a key you intend to work for reasons that are
valid
  in your environment / app / etc.
* The 'name' element is simply used to disambiguate entries, similar
  to -class entries in props.conf.  The name can be anything of your
  choosing, including a descriptive name for why you use the key.
* The entire stanza defaults to not being present, causing all keys not
  documented just above to be flagged.
```

transforms.conf.example

```
#   Version 6.5.0
#
# This is an example transforms.conf.  Use this file to create regexes
and
# rules for transforms.  Use this file in tandem with props.conf.
#
# To use one or more of these configurations, copy the configuration
block
# into transforms.conf in $SPLUNK_HOME/etc/system/local/. You must
restart
# Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

# Note: These are examples.  Replace the values with your own
customizations.


# Indexed field:

[netscreen-error]
REGEX =  device_id=\[w+\](?<err_code>[^:]+)
FORMAT = err_code::$1
WRITE_META = true


# Override host:

[hostoverride]
DEST_KEY = MetaData:Host
REGEX = \s(\w*)$
FORMAT = host::$1


# Extracted fields:

[netscreen-error-field]
REGEX = device_id=\[w+\](?<err_code>[^:]+)
FORMAT = err_code::$1


# Static lookup table

[mylookuptable]
```

```
filename = mytable.csv

# one to one lookup
# guarantees that we output a single lookup value for each input value,
if
# no match exists, we use the value of "default_match", which by
default is
# "NONE"
[mylook]
filename = mytable.csv
max_matches = 1
min_matches = 1
default_match = nothing

# Lookup and filter results
[myfilteredlookup]
filename = mytable.csv
filter = id<500 AND color="red"

# external command lookup table

[myexternaltable]
external_cmd = testadapter.py blah
fields_list = foo bar

# Temporal based static lookup table

[staticwtime]
filename = mytable.csv
time_field = timestamp
time_format = %d/%m/%y %H:%M:%S


# Mask sensitive data:

[session-anonymizer]
REGEX = (?m)^(.*)SessionId=\w+(\w{4}[&"].*)$
FORMAT = $1SessionId=########$2
DEST_KEY = _raw


# Route to an alternate index:

[AppRedirect]
REGEX = Application
DEST_KEY = _MetaData:Index
FORMAT = Verbose


# Extract comma-delimited values into fields:

[extract_csv]
```

```
DELIMS = ","
FIELDS = "field1", "field2", "field3"

# This example assigns the extracted values from _raw to field1, field2
and
# field3 (in order of extraction). If more than three values are
extracted
# the values without a matching field name are ignored.

[pipe_eq]
DELIMS = "|", "="

# The above example extracts key-value pairs which are separated by '|'
# while the key is delimited from value by '='.


[multiple_delims]
DELIMS = "|;", "=:"


# The above example extracts key-value pairs which are separated by '|'
or
# ';', while the key is delimited from value by '=' or ':'.


###### BASIC MODULAR REGULAR EXPRESSIONS DEFINITION START ###########
# When adding a new basic modular regex PLEASE add a comment that lists
# the fields that it extracts (named capturing groups), or whether it
# provides a placeholder for the group name as:
# Extracts: field1, field2....
#

[all_lazy]
REGEX = .*?

[all]
REGEX = .*

[nspaces]
# matches one or more NON space characters
REGEX = \S+

[alphas]
# matches a string containing only letters a-zA-Z
REGEX = [a-zA-Z]+

[alnums]
# matches a string containing letters + digits
REGEX = [a-zA-Z0-9]+

[qstring]
# matches a quoted "string" - extracts an unnamed variable
```

```
# name MUST be provided as in [[qstring:name]]
# Extracts: empty-name-group (needs name)
REGEX = "(?<>[^"]*+)"

[sbstring]
# matches a string enclosed in [] - extracts an unnamed variable
# name MUST be provided as in [[sbstring:name]]
# Extracts: empty-name-group (needs name)
REGEX = \[(?<>[^\]]*+)\]

[digits]
REGEX = \d+

[int]
# matches an integer or a hex number
REGEX = 0x[a-fA-F0-9]+|\d+

[float]
# matches a float (or an int)
REGEX = \d*\.\d+|[[int]]

[octet]
# this would match only numbers from 0-255 (one octet in an ip)
REGEX = (?:2(?:5[0-5]|[0-4][0-9])|[0-1][0-9][0-9]|[0-9][0-9]?)

[ipv4]
# matches a valid IPv4 optionally followed by :port_num the octets in
the ip
# would also be validated 0-255 range
# Extracts: ip, port
REGEX = (?<ip>[[octet]](?:\.[[octet]]){3})(?::[[int:port]])?

[simple_url]
# matches a url of the form proto://domain.tld/uri
# Extracts: url, domain
REGEX = (?<url>\w++://(?<domain>[a-zA-Z0-9\-.:]++)(?:/[^\s"]*)?)

[url]
# matches a url of the form proto://domain.tld/uri
# Extracts: url, proto, domain, uri
REGEX =
(?<url>[[alphas:proto]]://(?<domain>[a-zA-Z0-9\-.:]++)(?<uri>/[^\s"]*)?)

[simple_uri]
# matches a uri of the form /path/to/resource?query
# Extracts: uri, uri_path, uri_query
REGEX = (?<uri>(?<uri_path>[^\s\?"]++)(?:\\?(?<uri_query>[^\s"]+))?)

[uri]
# uri  = path optionally followed by query
[/this/path/file.js?query=part&other=var]
# path = root part followed by file        [/root/part/file.part]
```

```
# Extracts: uri, uri_path, uri_root, uri_file, uri_query, uri_domain
(optional if in proxy mode)
REGEX =
(?<uri>(?:\w++://(?<uri_domain>[^/\s]++))?(?<uri_path>(?<uri_root>/+(?:[^\s\?;=/]*+/+)*

[hide-ip-address]
# Make a clone of an event with the sourcetype masked_ip_address.  The
clone
# will be modified; its text changed to mask the ip address.
# The cloned event will be further processed by index-time transforms
and
# SEDCMD expressions according to its new sourcetype.
# In most scenarios an additional transform would be used to direct the
# masked_ip_address event to a different index than the original data.
REGEX = ^(.*?)src=\d+\.\d+\.\d+\.\d+(.*)$
FORMAT = $1src=XXXXX$2
DEST_KEY = _raw
CLONE_SOURCETYPE = masked_ip_addresses

###### BASIC MODULAR REGULAR EXPRESSIONS DEFINITION END ###########
```

ui-prefs.conf

The following are the spec and example files for ui-prefs.conf.

ui-prefs.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attribute/value pairs for ui preferences
for a
# view.
#
# There is a default ui-prefs.conf in $SPLUNK_HOME/etc/system/default.
To set
# custom configurations, place a ui-prefs.conf in
# $SPLUNK_HOME/etc/system/local/. To set custom configuration for an
app, place
# ui-prefs.conf in $SPLUNK_HOME/etc/apps/<app_name>/local/.  For
examples, see
# ui-prefs.conf.example. You must restart Splunk to enable
configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
```

```
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
top of
#     the file.
#   * Each conf file should have at most one default stanza. If there
are
#     multiple default stanzas, attributes are combined. In the case of
#     multiple definitions of the same attribute, the last definition in
the
#     file wins.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.

[<stanza name>]
* Stanza name is the name of the xml view file

dispatch.earliest_time =
dispatch.latest_time =

# Pref only options
display.prefs.autoOpenSearchAssistant = 0 | 1
display.prefs.timeline.height = <string>
display.prefs.timeline.minimized = 0 | 1
display.prefs.timeline.minimalMode = 0 | 1
display.prefs.aclFilter = [none|app|owner]
display.prefs.appFilter = <string>
display.prefs.listMode = [tiles|table]
display.prefs.searchContext = <string>
display.prefs.events.count = [10|20|50]
display.prefs.statistics.count = [10|20|50|100]
display.prefs.fieldCoverage = [0|.01|.50|.90|1]
display.prefs.enableMetaData = 0 | 1
display.prefs.showDataSummary = 0 | 1
display.prefs.customSampleRatio = <int>
display.prefs.showSPL = 0 | 1
display.prefs.livetail = 0 | 1

# count per page for listing pages
countPerPage = [10|20|50]
```

Display Formatting Options

```
#******
# Display Formatting Options
#******Display Formatting Options

# General options
display.general.enablePreview = 0 | 1

# Event options
# TODO: uncomment the fields when we are ready to merge the values
display.events.fields = <string>
display.events.type = [raw|list|table]
display.events.rowNumbers = 0 | 1
display.events.maxLines = [0|5|10|20|50|100|200]
display.events.raw.drilldown = [inner|outer|full|none]
display.events.list.drilldown = [inner|outer|full|none]
display.events.list.wrap = 0 | 1
display.events.table.drilldown = 0 | 1
display.events.table.wrap = 0 | 1

# Statistics options
display.statistics.rowNumbers = 0 | 1
display.statistics.wrap = 0 | 1
display.statistics.drilldown = [row|cell|none]

# Visualization options
display.visualizations.type = [charting|singlevalue]
display.visualizations.custom.type = <string>
display.visualizations.chartHeight = <int>
display.visualizations.charting.chart =
[line|area|column|bar|pie|scatter|radialGauge|fillerGauge|markerGauge]
display.visualizations.charting.chart.style = [minimal|shiny]
display.visualizations.charting.legend.labelStyle.overflowMode =
[ellipsisEnd|ellipsisMiddle|ellipsisStart]

# Patterns options
display.page.search.patterns.sensitivity = <float>

# Page options
display.page.search.mode = [fast|smart|verbose]
display.page.search.timeline.format = [hidden|compact|full]
display.page.search.timeline.scale = [linear|log]
display.page.search.showFields = 0 | 1
display.page.home.showGettingStarted = 0 | 1
display.page.search.searchHistoryTimeFilter = [0|@d|-7d@d|-30d@d]
```

ui-prefs.conf.example

```
#   Version 6.5.0
#
# This file contains example of ui preferences for a view.
#
# To use one or more of these configurations, copy the configuration
block into
# ui-prefs.conf in $SPLUNK_HOME/etc/system/local/. You must restart
Splunk to
# enable configurations.
#
# To learn more about configuration files (including precedence) please
see the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
#
# The following ui preferences will default timerange picker on the
search page
# from All time to Today We will store this ui-prefs.conf in
# $SPLUNK_HOME/etc/apps/search/local/ to only update search view of
search app.
[search]
dispatch.earliest_time = @d
dispatch.latest_time = now
```

ui-tour.conf

The following are the spec and example files for ui-tour.conf.

ui-tour.conf.spec

```
#   Version 6.5.0
#
# This file contains the tours available for Splunk Onboarding
#
# There is a default ui-tour.conf in $SPLUNK_HOME/etc/system/default.
# To create custom tours, place a ui-tour.conf in
# $SPLUNK_HOME/etc/system/local/. To create custom tours for an app,
place
# ui-tour.conf in $SPLUNK_HOME/etc/apps/<app_name>/local/.
#
# To learn more about configuration files (including precedence) see
the
```

```
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
#
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#   * You can also define global settings outside of any stanza, at the
top of
#     the file.
#   * This is not a typical conf file for configurations. It is used to
set/create
#     tours to demonstrate product functionality to users.
#   * If an attribute is defined at both the global level and in a
specific
#     stanza, the value in the specific stanza takes precedence.

[<stanza name>]
* Stanza name is the name of the tour

useTour = <string>
* Used to redirect this tour to another when called by Splunk.
* Optional

nextTour = <string>
* String used to determine what tour to start when current tour is
finished.
* Optional

intro = <string>
* A custom string used in a modal to describe what tour is about to be
taken.
* Optional

type = <image || interactive>
* Can either be "image" or "interactive" to determine what kind of tour
it is.
* Required

label = <string>
* The identifying name for this tour used in the tour creation app.
* Optional

tourPage = <string>
* The Splunk view this tour is associated with (only necessary if it is
linked to).
* Optional
```

```
viewed = <boolean>
* A boolean to determine if this tour has been viewed by a user.
* Set by Splunk
```

For image based tours

```
############################
## For image based tours
############################For image based tours
# Users can list as many images with captions as they want. Each new
image is created by
# incrementing the number.

imageName<int> = <string>
* The name of the image file (example.png)
* Required but Optional only after first is set

imageCaption<int> = <string>
* The caption string for corresponding image
* Optional

imgPath = <string>
* The subdirectory relative to Splunk's 'img' directory in which users
put the images.
  This will be appended to the url for image access and not make a
server request within Splunk.
  EX) If user puts images in a subdirectory 'foo': imgPath = foo.
  EX) If within an app, imgPath = foo will point to the app's img path
of
      appserver/static/img/foo
* Required only if images are not in the main 'img' directory.

context = <system || <specific app name>>
* String consisting of either 'system' or the app name the tour images
are to be stored.
* If set to 'system', it will revert to Splunk's native img path.
* Required
```

For interactive tours

```
############################
## For interactive tours
############################For interactive tours
# Users can list as many steps with captions as they want. Each new step
is created by
```

```
# incrementing the number.

urlData = <string>
* String of any querystring variables used with tourPage to create full
url executing this tour.
* Optional

stepText<int> = <string>
* The string used in specified step to describe the UI being showcased.
* Required but Optional only after first is set

stepElement<int> = <selector>
* The UI Selector used for highlighting the DOM element for
corresponding step.
* Optional

stepPosition<int> = <bottom || right || left || top>
* String that sets the position of the tooltip for corresponding step.
* Optional

stepClickEvent<int> = <click || mousedown || mouseup>
* Sets a specific click event for an element for corresponding step.
* Optional

stepClickElement<int> = <string>
* The UI selector used for a DOM element used in conjunction with click
above.
* Optional
```

ui-tour.conf.example

```
#   Version 6.5.0
#
# This file contains the tours available for Splunk Onboarding
#
# To update tours, copy the configuration block into
# ui-tour.conf in $SPLUNK_HOME/etc/system/local/. Restart the Splunk
software to
# see the changes.
#
# To learn more about configuration files (including precedence) see
the
# documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
#

# Image Tour
[tour-name]
```

```
type = image
imageName1 = TourStep1.png
imageCaption1 = This is the first caption
imageName2 = TourStep2.png
imageCaption2 = This is the second caption
imgPath = /testtour
context = system

# Interactive Tour
type = interactive
tourPage = reports
urlData =
label = Interactive Tour Test
stepText1 = Welcome to this test tour
stepText2 = This is the first step in the tour
stepElement2 = .test-selector
stepText3 = This is the second step in the tour
stepElement3 = .test-selector
stepClickEvent3 = mousedown
stepClickElement3 = .test-click-element
viewed = 0
```

user-prefs.conf

The following are the spec and example files for user-prefs.conf.

user-prefs.conf.spec

```
# This file describes some of the settings that are used, and
# can be configured on a per-user basis for use by the Splunk Web UI.

# Settings in this file are requested with user and application scope of
the
# relevant user, and the user-prefs app.

# Additionally, settings by the same name which are available in the
roles
# the user belongs to will be used at lower precedence.

# This means interactive setting of these values will cause the values
to be
# updated in
# $SPLUNK_HOME/etc/users/<username>/user-prefs/local/user-prefs.conf
where
# <username> is the username for the user altering their preferences.
```

```
# It also means that values in another app will never be used unless
they
# are exported globally (to system scope) or to the user-prefs app.

# In practice, providing values in other apps isn't very interesting,
since
# values from the authorize.conf roles settings are more typically
sensible
# ways to defaults for values in user-prefs.
```

[general]

```
[general]
default_namespace = <app name>
* Specifies the app that the user will see initially upon login to the
  Splunk Web User Interface.
* This uses the "short name" of the app, such as launcher, or search,
  which is synonymous with the app directory name.
* Splunk defaults this to 'launcher' via the default authorize.conf

tz = <timezone>
* Specifies the per-user timezone to use
* If unset, the timezone of the Splunk Server or Search Head is used.
* Only canonical timezone names such as America/Los_Angeles should be
  used (for best results use the Splunk UI).
* Defaults to unset.

lang = <language>
* Specifies the per-user language preference for non-webui operations,
where
  multiple tags are separated by commas.
* If unset, English "en-US" will be used when required.
* Only tags used in the "Accept-Language" HTTP header will be allowed,
such as
  "en-US" or "fr-FR".
* Fuzzy matching is supported, where "en" will match "en-US".
* Optional quality settings is supported, such as
"en-US,en;q=0.8,fr;q=0.6"
* Defaults to unset.

install_source_checksum = <string>
* Records a checksum of the tarball from which a given set of private
user
  configurations was installed.
* Analogous to <install_source_checksum> in app.conf.

search_syntax_highlighting = <boolean>
* Highlights different parts of a search string with different colors.
* Defaults to true.
```

```
search_assistant = [full|compact|none]
* Specifies the type of search assistant to use when constructing a
search.
* Defaults to compact.

infodelivery_enabled = <boolean>
* Enables the info delivery app
* Defaults to true

infodelivery_show_ad_modal = <boolean>
* Flag to disable/enable the ad modal for info delivery app
* Defaults to true

infodelivery_show_configure_modal = <boolean>
* Flag to disable/enable the configure modal for info delivery
* Defaults to true

datasets:showInstallDialog = <boolean>
* Flag to enable/disable the install dialog for the datasets addon
* Defaults to true
```

[default]

```
[default]
# Additional settings exist, but are entirely UI managed.
<setting> = <value>
```

[general_default]

```
[general_default]
default_earliest_time = <string>
default_latest_time = <string>
* Sets the global default time range across all apps, users, and roles
on the search page.
```

user-prefs.conf.example

```
#   Version 6.5.0
#
# This is an example user-prefs.conf.  Use this file to configure
settings
# on a per-user basis for use by the Splunk Web UI.
#
# To use one or more of these configurations, copy the configuration
block
```

```
# into user-prefs.conf in $SPLUNK_HOME/etc/system/local/. You must
restart
# Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

# Note: These are examples.  Replace the values with your own
# customizations.


# EXAMPLE: Setting the default timezone to GMT for all Power and User
role
# members, and setting a different language preference for each.

[role_power]
tz = GMT
lang = en-US

[role_user]
tz = GMT
lang = fr-FR,fr-CA;q=0
```

user-seed.conf

The following are the spec and example files for user-seed.conf.

user-seed.conf.spec

```
#   Version 6.5.0
#
# Specification for user-seed.conf.  Allows configuration of Splunk's
# initial username and password.  Currently, only one user can be
configured
# with user-seed.conf.
#
# Specification for user-seed.conf.  Allows configuration of Splunk's
initial username and password.
# Currently, only one user can be configured with user-seed.conf.
#
# To override the default username and password, place user-seed.conf in
# $SPLUNK_HOME/etc/system/local. You must restart Splunk to enable
configurations.
```

```
#
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
# To learn more about configuration files (including precedence) please
see the documentation
# located at
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

[user_info]

```
[user_info]
* Default is Admin.

USERNAME = <string>
          * Username you want to associate with a password.
          * Default is Admin.

PASSWORD = <password>
          * Password you wish to set for that user.
          * Default is changeme.
```

user-seed.conf.example

```
#   Version 6.5.0
#
# This is an example user-seed.conf.  Use this file to create an
initial
# This is an example user-seed.conf.  Use this file to create an
initial login.
#
# NOTE: To change the default start up login and password, this file
must be
# NOTE: To change the default start up login and password, this file
must be in
# $SPLUNK_HOME/etc/system/default/ prior to starting Splunk for the
first time.
#
# To use this configuration, copy the configuration block into
# To use this configuration, copy the configuration block into
user-seed.conf
# in $SPLUNK_HOME/etc/system/local/. You must restart Splunk to enable
configurations.
#
# To learn more about configuration files (including precedence) please
see
# To learn more about configuration files (including precedence) please
see the documentation
```

```
# located at
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles


[user_info]
USERNAME = admin
PASSWORD = myowndefaultPass
```

viewstates.conf

The following are the spec and example files for viewstates.conf.

viewstates.conf.spec

```
#   Version 6.5.0
#
# This file explains how to format viewstates.
#
# To use this configuration, copy the configuration block into
# viewstates.conf in $SPLUNK_HOME/etc/system/local/. You must restart
Splunk
# to enable configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#  * You can also define global settings outside of any stanza, at the
top
#    of the file.
#  * Each conf file should have at most one default stanza. If there
are
#    multiple default stanzas, attributes are combined. In the case of
#    multiple definitions of the same attribute, the last definition in
the
#    file wins.
#  * If an attribute is defined at both the global level and in a
specific
```

```
#    stanza, the value in the specific stanza takes precedence.
```

[<view_name>:<viewstate_id>]

```
[<view_name>:<viewstate_id>]
* Auto-generated persistence stanza label that corresponds to UI views
* The <view_name> is the URI name (not label) of the view to persist
* if <view_name> = "*", then this viewstate is considered to be 'global'
* The <viewstate_id> is the unique identifier assigned to this set of
  parameters
* <viewstate_id> = '_current' is a reserved name for normal view
  'sticky state'
* <viewstate_id> = '_empty' is a reserved name for no persistence,
  i.e., all defaults

<module_id>.<setting_name> = <string>
* The <module_id> is the runtime id of the UI module requesting
persistence
* The <setting_name> is the setting designated by <module_id> to persist
```

viewstates.conf.example

```
#   Version 6.5.0
#
# This is an example viewstates.conf.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

[charting:g3b5fa7l]
ChartTypeFormatter_0_7_0.default = area
Count_0_6_0.count = 10
LegendFormatter_0_13_0.default = right
LineMarkerFormatter_0_10_0.default = false
NullValueFormatter_0_12_0.default = gaps

[*:g3jck9ey]
Count_0_7_1.count = 20
DataOverlay_0_12_0.dataOverlayMode = none
DataOverlay_1_13_0.dataOverlayMode = none
FieldPicker_0_6_1.fields = host sourcetype source date_hour date_mday
date_minute date_month
FieldPicker_0_6_1.sidebarDisplay = True
FlashTimeline_0_5_0.annotationSearch = search index=twink
```

```
FlashTimeline_0_5_0.enableAnnotations = true
FlashTimeline_0_5_0.minimized = false
MaxLines_0_13_0.maxLines = 10
RowNumbers_0_12_0.displayRowNumbers = true
RowNumbers_1_11_0.displayRowNumbers = true
RowNumbers_2_12_0.displayRowNumbers = true
Segmentation_0_14_0.segmentation = full
SoftWrap_0_11_0.enable = true

[dashboard:_current]
TimeRangePicker_0_1_0.selected = All time
```

visualizations.conf

The following are the spec and example files for visualizations.conf.

visualizations.conf.spec

```
#   Version 6.5.0
#
# This file contains definitions for visualizations an app makes
avialable
# to the system. An app intending to share visualizations with the
system
# should include a visualizations.conf in
$SPLUNK_HOME/etc/apps/<appname>/default
#
# visualizations.conf should include one stanza for each visualization
to be shared
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

#*******
# The possible attribute/value pairs for visualizations.conf are:
#*******
```

[<stanza name>]

```
[<stanza name>]
* Create a unique stanza name for each visualization. It should match
the name
```

```
  of the visualization
* Follow the stanza name with any number of the following
attribute/value
  pairs.
* If you do not specify an attribute, Splunk uses the default.

allow_user_selection = <bool>
* Optional.
* Whether the visualization should be available for users to select
* Defaults to true

default_height = <int>
* Optional.
* The default height of the visualization in pixels
* Defaults to 250

description = <string>
* Required.
* The short description that will show up in the visualization picker
* Defaults to ""

disabled = <bool>
* Optional.
* Disable the visualization by setting to true.
* If set to true, the visualization is not available anywhere in Splunk
* Defaults to false.

label = <string>
* Required.
* The human-readable label or title of the visualization
* Will be used in dropdowns and lists as the name of the visualization
* Defaults to <app_name>.<viz_name>

search_fragment = <string>
* Required.
* An example part of a search that formats the data correctly for the
viz. Typically the last pipe(s) in a search query.
* Defaults to ""
```

visualizations.conf.example

```
No example
```

web.conf

The following are the spec and example files for web.conf.

web.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attributes and values you can use to
configure
# Splunk's web interface.
#
# There is a web.conf in $SPLUNK_HOME/etc/system/default/.  To set
custom
# configurations, place a web.conf in $SPLUNK_HOME/etc/system/local/.
 For
# examples, see web.conf.example.  You must restart Splunk to enable
# configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles


[settings]
* Set general SplunkWeb configuration options under this stanza name.
* Follow this stanza name with any number of the following
attribute/value
  pairs.
* If you do not specify an entry for each attribute, Splunk will use the
  default value.

startwebserver = [0 | 1]
* Set whether or not to start SplunkWeb.
* 0 disables SplunkWeb, 1 enables it.
* Defaults to 1.

httpport = <port_number>
* Must be present for SplunkWeb to start.
* If omitted or 0 the server will NOT start an http listener.
* If using SSL, set to the HTTPS port number.
* Defaults to 8000.

mgmtHostPort = <IP:port>
* Location of splunkd.
* Don't include http[s]:// -- just the IP address.
* Defaults to 127.0.0.1:8089.

appServerPorts = <one or more port numbers>
* Port number(s) for the python-based application server to listen on.
  This port is bound only on the loopback interface -- it is not
  exposed to the network at large.
```

```
* If set to "0", this will prevent the application server from
  being run from splunkd.  Instead, splunkweb will be started as
  a separate python-based service which directly listens to the
  'httpport'.  This is how Splunk 6.1.X and earlier behaved.
* Generally, you should only set one port number here.  For most
  deployments a single application server won't be a performance
  bottleneck.  However you can provide a comma-separated list of
  port numbers here and splunkd will start a load-balanced
  application server on each one.
* It is recommended that this be set to a non-zero value.  Setting
  this to "0" should only be done if you experience a compatibility
  problem.  The new separate application server configuration is faster
  and supports more configuration options.  Also, setting this to
  "0" may cause problems with new functionality, such as using the
  Search Head Clustering feature.
  (see the [shclustering] stanza in server.conf)
* Defaults to 8065.

splunkdConnectionTimeout = <integer>
* Number of seconds to wait before timing out when communicating with
  splunkd
* Must be at least 30
* Values smaller than 30 will be ignored, resulting in the use of the
  default value
* Defaults to 30

enableSplunkWebSSL = [True | False]
* Toggle between http or https.
* Set to true to enable https and SSL.
* Defaults to False.

privKeyPath = <path>
* The path to the file containing the web server's SSL certificate's
private
  key.
* A relative path is interpreted relative to $SPLUNK_HOME and may not
refer
  outside of $SPLUNK_HOME (e.g., no ../somewhere).
* An absolute path can also be specified to an external key.
* See also 'enableSplunkWebSSL' and 'serverCert'.

serverCert = <path>
* Full path to the PEM format Splunk web server certificate file.
* The file may also contain root and intermediate certificates, if
required.
  They should be listed sequentially in the order:
    [ Server's SSL certificate ]
    [ One or more intermediate certificates, if required ]
    [ Root certificate, if required ]
* Default is $SPLUNK_HOME/etc/auth/splunkweb/cert.pem.
* See also 'enableSplunkWebSSL' and 'privKeyPath'.
```

```
caCertPath = <path>
* DEPRECATED; use 'serverCert' instead.
* A relative path is interpreted relative to $SPLUNK_HOME and may not
refer
  outside of $SPLUNK_HOME (e.g., no ../somewhere).

requireClientCert = [True | False]
* Requires that any HTTPS client that connects to the splunkweb HTTPS
  server has a certificate that was signed by the CA cert installed
  on this server.
* If true, a client can connect ONLY if a certificate created by our
  certificate authority was used on that client.
* When true, it is mandatory to configure splunkd with same root CA in
server.conf.
  This is needed for internal communication between splunkd and
splunkweb.
* Defaults to false.

sslCommonNameToCheck = <commonName1>, <commonName2>, ...
* Optional. Defaults to no common name checking.
* Check the common name of the client's certificate against this list of
names.
* requireClientCert must be set to true for this setting to work.

sslAltNameToCheck = <alternateName1>, <alternateName2>, ...
* If this value is set, and 'requireClientCert' is set to true,
  splunkweb will verify certificates which have a so-called
  "Subject Alternate Name" that matches any of the alternate names in
this list.
  * Subject Alternate Names are effectively extended descriptive
    fields in SSL certs beyond the commonName. A common practice for
    HTTPS certs is to use these values to store additional valid
    hostnames or domains where the cert should be considered valid.
* Accepts a comma-separated list of Subject Alternate Names to consider
valid.
* Optional. Defaults to no alternate name checking

serviceFormPostURL = http://docs.splunk.com/Documentation/Splunk
* This attribute is deprecated since 5.0.3

userRegistrationURL = https://www.splunk.com/page/sign_up
updateCheckerBaseURL = http://quickdraw.Splunk.com/js/
docsCheckerBaseURL = http://quickdraw.splunk.com/help
* These are various Splunk.com urls that are configurable.
* Setting updateCheckerBaseURL to 0 will stop the SplunkWeb from
pinging
  Splunk.com for new versions of itself.

enable_insecure_login = [True | False]
* Indicates if the GET-based /account/insecurelogin endpoint is enabled
* Provides an alternate GET-based authentication mechanism
* If True, the /account/insecurelogin?username=USERNAME&password=PASSWD
```

```
is
  available
* If False, only the main /account/login endpoint is available
* Defaults to False

simple_error_page = [True | False]
   * If set to True a simplified error page will be displayed for
errors (404, 500, etc.) only containing the status
   * If set to False a more verbose error page will be displayed
containing homelink, message, more_results_link, crashes, referrer,
debug output and byline
   * Defaults to False

login_content = <content_string>
* Add custom content to the login page
* Supports any text including html

sslVersions = <list of ssl versions string>
* Comma-separated list of SSL versions to support
* The versions available are "ssl3", "tls1.0", "tls1.1", and "tls1.2"
* The special version "*" selects all supported versions.  The version
"tls"
  selects all versions tls1.0 or newer
* If a version is prefixed with "-" it is removed from the list
* SSLv2 is always disabled; "-ssl2" is accepted in the version list but
does nothing
* When appServerPorts=0 only supported values are  "all", "ssl3, tls"
  and "tls"
* When configured in FIPS mode ssl3 is always disabled regardless
  of this configuration
* Defaults to "ssl3, tls".  (anything newer than SSLv2)
* NOTE: this setting only takes effect when appServerPorts is set to a
  non-zero value

supportSSLV3Only = <bool>
* When appServerPorts is set to a non-zero value (the default mode),
  this setting is DEPRECATED.  SSLv2 is now always disabled.
  The exact set of SSL versions allowed is now configurable via the
  "sslVersions" setting above.
* When appServerPorts is set to zero, this controls whether we disallow
SSLv2
  connections.

cipherSuite = <cipher suite string>
* If set, uses the specified cipher string for the HTTP server.
* If not set, uses the default cipher string provided by OpenSSL.  This
is
  used to ensure that the server does not accept connections using weak
  encryption protocols.
* Must specify 'dhFile' to enable any Diffie-Hellman ciphers.

ecdhCurves = <comma separated list of ec curves>
```

```
* ECDH curves to use for ECDH key negotiation.
* The curves should be specified in the order of preference.
* The client sends these curves as a part of Client Hello.
* The server supports only the curves specified in the list.
* We only support named curves specified by their SHORT names.
  (see struct ASN1_OBJECT in asn1.h)
* The list of valid named curves by their short/long names can be
obtained
  by executing this command:
  $SPLUNK_HOME/bin/splunk cmd openssl ecparam -list_curves
* Default is empty string.
* e.g. ecdhCurves = prime256v1,secp384r1,secp521r1

ecdhCurveName = <string>
* DEPRECATED; use 'ecdhCurves' instead.
* ECDH curve to use for ECDH key negotiation
* We only support named curves specified by their SHORT name.
  (see struct ASN1_OBJECT in asn1.h)
* The list of valid named curves by their short/long names can be
obtained
  by executing this command:
  $SPLUNK_HOME/bin/splunk cmd openssl ecparam -list_curves
* Default is empty string.

dhFile = <path>
* The path to the Diffie-Hellman parameter file.
* Relative paths are interpreted as relative to $SPLUNK_HOME.
* Relative paths may not refer outside of $SPLUNK_HOME (eg. no
../somewhere).
* An absolute path can also be specified to an external key.
* Not set by default.

root_endpoint = <URI_prefix_string>
* Defines the root URI path on which the appserver will listen
* Default setting is '/'
* Ex: if you want to proxy the splunk UI at
http://splunk:8000/splunkui,
  then set root_endpoint = /splunkui

static_endpoint = <URI_prefix_string>
* Path to static content
* The path here is automatically appended to root_endpoint defined
above
* Default is /static

static_dir = <relative_filesystem_path>
* The directory that actually holds the static content
* This can be an absolute url if you want to put it elsewhere
* Default is share/splunk/search_mrsparkle/exposed

rss_endpoint = <URI_prefix_string>
* Path to static rss content
```

```
* The path here is automatically appended to root_endpoint defined
above
* Default is /rss

embed_uri = <URI>
* Optional URI scheme/host/port prefix for embedded content
* This presents an optional strategy for exposing embedded shared
  content that does not require authentication in reverse proxy/SSO
  environment.
* Default is empty and will resolve to the client
  window.location.protocol + "//" + window.location.host

embed_footer = <html_string>
* chunk of html to define the footer for an embedded report.
* Defaults to "splunk>" but can be changed to whatever the user would
like.

tools.staticdir.generate_indexes = [1 | 0]
* Indicates if the webserver will serve a directory listing for static
  directories
* Defaults to 0 (false)

template_dir = <relative_filesystem_path>
* Base path to mako templates
* Defaults to share/splunk/search_mrsparkle/templates

module_dir = <relative_filesystem_path>
* Base path to UI module assets
* Defaults to share/splunk/search_mrsparkle/modules

enable_gzip = [True | False]
* Determines if webserver applies gzip compression to responses
* Defaults to True

use_future_expires = [True | False]
* Determines if the Expires header of /static files is set to a
far-future date
* Defaults to True

flash_major_version = <integer>
flash_minor_version = <integer>
flash_revision_version = <integer>
* Specifies the minimum Flash plugin version requirements
* Flash support, broken into three parts.
* We currently require a min baseline of Shockwave Flash 9.0 r124

override_JSON_MIME_type_with_text_plain = [True | False]
* Specifies whether or not to override the MIME type for JSON data
served up
  by splunkweb endpoints with content-type="text/plain; charset=UTF-8"
* If True, splunkweb endpoints (other than proxy) that serve JSON data
will
```

```
  serve as "text/plain; charset=UTF-8"
* If False, splunkweb endpoints that serve JSON data will serve as
"application/json; charset=UTF-8"

enable_proxy_write = [True | False]
* Indicates if the /splunkd proxy endpoint allows POST operations
* If True, both GET and POST operations are proxied through to splunkd
* If False, only GET operations are proxied through to splunkd
* Setting this to False will prevent many client-side packages (such as
the
  Splunk JavaScript SDK) from working correctly
* Defaults to True

js_logger_mode = [None | Firebug | Server]
* JavaScript Logger mode
* Available modes: None, Firebug, Server
* Mode None: Does not log anything
* Mode Firebug: Use firebug by default if it exists or defer to the
older
  less promiscuous version of firebug lite
* Mode Server: Log to a defined server endpoint
* See js/logger.js Splunk.Logger.Mode for mode implementation details
and if
  you would like to author your own
* Defaults to None

js_logger_mode_server_end_point = <URI_relative_path>
* Specifies the server endpoint to post javascript log messages
* Used when js_logger_mode = Server
* Defaults to util/log/js

js_logger_mode_server_poll_buffer = <integer>
* Specifieds the interval in milliseconds to check, post and cleanse the
javascript log buffer
* Defaults to 1000

js_logger_mode_server_max_buffer = <integer>
* Specifies the maximum size threshold to post and cleanse the
javascript log buffer
* Defaults to 100

ui_inactivity_timeout = <integer>
* Specifies the length of time lapsed (in minutes) for notification when
  there is no user interface clicking, mouseover, scrolling or resizing.
* Notifies client side pollers to stop, resulting in sessions expiring
at
  the tools.sessions.timeout value.
* If less than 1, results in no timeout notification ever being
triggered
  (Sessions will stay alive for as long as the browser is open).
* Defaults to 60 minutes
```

```
js_no_cache = [True | False]
* Toggle js cache control
* Defaults to False

cacheBytesLimit = <integer>
* When appServerPorts is set to a non-zero value, splunkd can keep a
  small cache of static assets in memory.  When the total size of the
  objects in cache grows larger than this, we begin the process of
ageing
  entries out.
* Defaults to 4194304 (i.e. 4 Megabytes)
* If set to zero, this cache is completely disabled

cacheEntriesLimit = <integer>
* When appServerPorts is set to a non-zero value, splunkd can keep a
  small cache of static assets in memory.  When the number of the
objects
  in cache grows larger than this, we begin the process of ageing
  entries out.
* Defaults to 16384
* If set to zero, this cache is completely disabled

staticCompressionLevel = <integer>
* When appServerPorts is set to a non-zero value, splunkd can keep a
  small cache of static assets in memory.  These are stored
  compressed and can usually be served directly to the web browser
  in compressed format.
* This level can be a number between 1 and 9.  Lower numbers use less
  CPU time to compress objects, but the resulting compressed objects
  will be larger.
* Defaults to 9.  Usually not much CPU time is spent compressing these
  objects so there is not much benefit to decreasing this.

enable_autocomplete_login = [True | False]
* Indicates if the main login page allows browsers to autocomplete the
username
* If True, browsers may display an autocomplete drop down in the
username field
* If False, browsers are instructed not to show autocomplete drop down
in the username field
* Defaults to False

verifyCookiesWorkDuringLogin = [True | False]
* Normally, the login page will make an attempt to see if cookies work
  properly in the user's browser before allowing them to log in.  If
  this is set to False, this check is skipped.
* Defaults to True.  It is recommended that this be left on.
* NOTE: this setting only takes effect when appServerPorts is set to a
       non-zero value

minify_js = [True | False]
* Indicates whether the static JS files for modules are consolidated and
```

```
minified
* Enabling improves client-side performance by reducing the number of
HTTP
  requests and the size of HTTP responses

minify_css = [True | False]
* Indicates whether the static CSS files for modules are consolidated
and
  minified
* Enabling improves client-side performance by reducing the number of
HTTP
  requests and the size of HTTP responses
* Due to browser limitations, disabling this when using IE9 and earlier
may
  result in display problems.

trap_module_exceptions = [True | False]
* Toggle whether the JS for individual modules is wrapped in a try/catch
* If True, syntax errors in individual modules will not cause the UI to
  hang, other than when using the module in question
* Set this to False when developing apps.

enable_pivot_adhoc_acceleration = [True | False]
* Toggle whether the pivot interface will use its own ad-hoc
acceleration
  when a data model is not accelerated.
* If True, this ad-hoc acceleration will be used to make reporting in
pivot
  faster and more responsive.
* In situations where data is not stored in time order or where the
majority
  of events are far in the past, disabling this behavior can improve
the
  pivot experience.
* DEPRECATED in version 6.1 and later, use
pivot_adhoc_acceleration_mode
  instead

pivot_adhoc_acceleration_mode = [Elastic | AllTime | None]
* Specify the type of ad-hoc acceleration used by the pivot interface
when a
  data model is not accelerated.
* If Elastic, the pivot interface will only accelerate the time range
  specified for reporting, and will dynamically adjust when this time
range
  is changed.
* If AllTime, the pivot interface will accelerate the relevant data over
all
  time.  This will make the interface more responsive to time-range
changes
  but places a larger load on system resources.
* If None, the pivot interface will not use any acceleration.  This
```

```
means
  any change to the report will require restarting the search.
* Defaults to Elastic

jschart_test_mode = [True | False]
* Toggle whether JSChart module runs in Test Mode
* If True, JSChart module attaches HTML classes to chart elements for
  introspection
* This will negatively impact performance, so should be disabled unless
  actively in use.

#
# JSChart data truncation configuration
# To avoid negatively impacting browser performance, the JSChart library
# places a limit on the number of points that will be plotted by an
# individual chart.  This limit can be configured here either across all
# browsers or specifically per-browser.  An empty or zero value will
disable
# the limit entirely.
#

jschart_truncation_limit = <int>
* Cross-broswer truncation limit, if defined takes precedence over the
  browser-specific limits below

jschart_truncation_limit.chrome = <int>
* Chart truncation limit for Chrome only
* Defaults to 50000

jschart_truncation_limit.firefox = <int>
* Chart truncation limit for Firefox only
* Defaults to 50000

jschart_truncation_limit.safari = <int>
* Chart truncation limit for Safari only
* Defaults to 50000

jschart_truncation_limit.ie11 = <int>
* Chart truncation limit for Internet Explorer 11 only
* Defaults to 50000

dashboard_html_allow_inline_styles = <bool>
* If this setting is set to false styles attributes from inline HTML
elements
* in dashboards will be removed to prevent potential attacks.
* Default is true

max_view_cache_size = <integer>
* Specifies the maximum number of views to cache in the appserver.
* Defaults to 300.

pdfgen_is_available = [0 | 1]
```

```
* Specifies whether Integrated PDF Generation is available on this
search
  head
* This is used to bypass an extra call to splunkd
* Defaults to 1 on platforms where node is supported, defaults to 0
  otherwise

version_label_format = <printf_string>
* Internal config
* Used to override the version reported by the UI to *.splunk.com
resources
* Defaults to: %s

auto_refresh_views = [0 | 1]
* Specifies whether the following actions cause the appserver to ask
splunkd
  to reload views from disk.
  * Logging in via the UI
  * Switching apps
  * Clicking the Splunk logo
* Defaults to 0.

#
# Splunk bar options
#
# Internal config. May change without notice.
# Only takes effect if instanceType is 'cloud'.
#

showProductMenu = [True | False]
    * Used to indicate visibility of product menu.
    * Defaults to False.

productMenuUriPrefix = <string>
    * The domain product menu links to.
    * Required if showProductMenu is set to True.

productMenuLabel = <string>
    * Used to change the text label for product menu.
    * Defaults to 'My Splunk'.

showUserMenuProfile = [True | False]
    * Used to indicate visibility of 'Profile' link within user menu.
    * Defaults to False.


#
# Header options
#
x_frame_options_sameorigin = [True | False]
* adds a X-Frame-Options header set to "SAMEORIGIN" to every response
served
```

```
* by cherrypy
* Defaults to True

#
```

SSO

```
# SSO
#

remoteUser = <http_header_string>
* Remote user HTTP header sent by the authenticating proxy server.
* This header should be set to the authenticated user.
* Defaults to 'REMOTE_USER'.
* Caution: There is a potential security concern regarding Splunk's
  treatment of HTTP headers.
* Your proxy provides the selected username as an HTTP header as
specified
  above.
* If the browser or other http agent were to specify the value of this
  header, probably any proxy would overwrite it, or in the case that the
  username cannot be determined, refuse to pass along the request or set
  it blank.
* However, Splunk (cherrypy) will normalize headers containing the dash,
  and the underscore to the same value.  For example USER-NAME and
  USER_NAME will be treated as the same in SplunkWeb.
* This means that if the browser provides REMOTE-USER and splunk accepts
  REMOTE_USER, theoretically the browser could dictate the username.
* In practice, however, in all our testing, the proxy adds its headers
  last, which causes them to take precedence, making the problem moot.
* See also the 'remoteUserMatchExact' setting which can enforce more
exact
  header matching when running with appServerPorts enabled.

remoteGroups = <http_header_string>
* Remote groups HTTP header name sent by the authenticating proxy
server.
* This value is used by splunk the match against header name.
* The header value format should be set to comma separated groups that
user belongs to.
* Example of header value: Products,Engineering,Quality Assurance
* There is no default value set for this parameter.

remoteGroupsQuoted = [true | false]
* This attribute is considered only when 'remoteGroups' is set.
* When value is set to true, the group header value can be comma
separated
* quoted entries. Note: These entries can contain comma.
* Example of header value with quoted entries:
* "Products","North America, Engineering","Quality Assurance"
```

```
* By default value is set to false, in which case group entries should
be without quotes.

remoteUserMatchExact = [0 | 1]
* IMPORTANT: this setting only takes effect when appServerPorts is set
to a
  non-zero value
* When matching the remoteUser header with value "1", consider dashes
and
  underscores distinct (so "Remote-User" and "Remote_User" will be
  considered different headers)
* Value "0" is for compatibility with older versions of Splunk, but
  setting to "1" is a good idea when setting up SSO with appServerPorts
  enabled
* Defaults to "0"

remoteGroupsMatchExact = [0 | 1]
* IMPORTANT: this setting only takes effect when appServerPorts is set
to a
  non-zero value
* When matching the remoteGroups header with value "1", consider dashes
and
  underscores distinct (so "Remote-Groups" and "Remote_Groups" will be
  considered different headers)
* Value "0" is for compatibility with older versions of Splunk, but
  setting to "1" is a good idea when setting up SSO with appServerPorts
  enabled
* Defaults to "0"

SSOMode = [permissive | strict]
* Allows SSO to behave in either permissive or strict mode.
* Permissive: Requests to Splunk Web that originate from an untrusted IP
  address are redirected to a login page where they can log into Splunk
  without using SSO.
* Strict: All requests to splunkweb will be restricted to those
originating
  from a trusted IP except those to endpoints not requiring
authentication.
* Defaults to "strict"

trustedIP = <ip_address>
* Trusted IP.  This is the IP address of the authenticating proxy.
* Splunkweb verifies it is receiving data from the proxy host for all
  SSO requests.
* Uncomment and set to a valid IP address to enable SSO.
* Disabled by default.  Normal value is '127.0.0.1'
* If appServerPorts is set to a non-zero value, this setting can accept
a
  richer set of configurations, using the same format as the
"acceptFrom"
  setting.
```

```
allowSsoWithoutChangingServerConf = [0 | 1]
* IMPORTANT: this setting only takes effect when appServerPorts is set
to a
  non-zero value
* Usually when configuring SSO, a trustedIP needs to be set both here
  in web.conf and also in server.conf.  If this is set to "1" then we
will
  enable web-based SSO without a trustedIP in server.conf
* Defaults to 0

testing_endpoint = <relative_uri_path>
* Specifies the root URI path on which to serve splunkweb unit and
  integration testing resources.
* Development only setting
* Defaults to '/testing'

testing_dir = <relative_file_path>
* Specifies the path relative to $SPLUNK_HOME that contains the testing
  files to be served at endpoint defined by 'testing_endpoint'.
* Development only setting
* Defaults to 'share/splunk/testing'

ssoAuthFailureRedirect = <scheme>://<URL>
   * Used to set the redirect URL if SSO authentication fails.
     Examples:
     http://www.example.com
     https://www.example.com
   * Defaults to an empty value and will show the default unauthorized
error
     page if SSO authentication fails.

# Results export server config

export_timeout = <integer>
* When exporting results, the number of seconds the server waits before
* closing the connection with splunkd. If you do not set a value for
* export_timeout, the value in splunkdConnectionTimeout is used.
* We recommend that you set export_timeout to a value greater than 30

#
# cherrypy HTTP server config
#

server.thread_pool = <integer>
* Specifies the minimum number of threads the appserver is allowed to
  maintain
* Defaults to 20

server.thread_pool_max = <integer>
* Specifies the maximum number of threads the appserver is allowed to
  maintain
* Defaults to -1 (unlimited)
```

```
server.thread_pool_min_spare = <integer>
* Specifies the minimum number of spare threads the appserver keeps idle
* Defaults to 5

server.thread_pool_max_spare = <integer>
* Specifies the maximum number of spare threads the appserver keeps idle
* Defaults to 10

server.socket_host = <ip_address>
* Host values may be any IPv4 or IPv6 address, or any valid hostname.
* The string 'localhost' is a synonym for '127.0.0.1' (or '::1', if your
  hosts file prefers IPv6). The string '0.0.0.0' is a special IPv4 entry
  meaning "any active interface" (INADDR_ANY), and '::' is the similar
  IN6ADDR_ANY for IPv6.
* Defaults to 0.0.0.0 if listenOnIPv6 is set to no, else ::

server.socket_timeout = <integer>
* The timeout in seconds for accepted connections between the browser
and
  splunkweb
* Defaults to 10

listenOnIPv6 = <no | yes | only>
* By default, splunkweb will listen for incoming connections using
  IPv4 only
* To enable IPv6 support in splunkweb, set this to "yes".  Splunkweb
  will simultaneously listen for connections on both IPv4 and IPv6
* To disable IPv4 entirely, set this to "only", which will cause
splunkweb
* to exclusively accept connections over IPv6.
* You will also want to set server.socket_host (use "::"
  instead of "0.0.0.0") if you wish to listen on an IPv6 address

max_upload_size = <integer>
* Specifies the hard maximum size of uploaded files in MB
* Defaults to 500

log.access_file = <filename>
* Specifies the HTTP access log filename
* Stored in default Splunk /var/log directory
* Defaults to web_access.log

log.access_maxsize = <integer>
* Specifies the maximum size the web_access.log file should be allowed
to
  grow to (in bytes)
* Comment out or set to 0 for unlimited file size
* File will be rotated to web_access.log.0 after max file size is
reached
* See log.access_maxfiles to limit the number of backup files created
* Defaults to unlimited file size
```

```
log.access_maxfiles = <integer>
* Specifies the maximum number of backup files to keep after the
  web_access.log file has reached its maximum size
* Warning: setting this to very high numbers (eg. 10000) may impact
  performance during log rotations
* Defaults to 5 if access_maxsize is set

log.error_maxsize = <integer>
* Specifies the maximum size the web_service.log file should be allowed
to
  grow to (in bytes)
* Comment out or set to 0 for unlimited file size
* File will be rotated to web_service.log.0 after max file size is
reached
* See log.error_maxfiles to limit the number of backup files created
* Defaults to unlimited file size

log.error_maxfiles = <integer>
* Specifies the maximum number of backup files to keep after the
  web_service.log file has reached its maximum size
* Warning: setting this to very high numbers (eg. 10000) may impact
          performance during log rotations
* Defaults to 5 if access_maxsize is set

log.screen = [True | False]
* Indicates if runtime output is displayed inside an interactive tty
* Defaults to True

request.show_tracebacks = [True | False]
* Indicates if a an exception traceback is displayed to the user on
fatal
  exceptions
* Defaults to True

engine.autoreload_on = [True | False]
* Indicates if the appserver will auto-restart if it detects a python
file
  has changed
* Defaults to False

tools.sessions.on = True
* Indicates if user session support is enabled
* Should always be True

tools.sessions.timeout = <integer>
* Specifies the number of minutes of inactivity before a user session is
  expired
* The countdown is effectively reset by browser activity minute until
  ui_inactivity_timeout inactivity timeout is reached.
* Use a value of 2 or higher, as a value of 1 will race with the
browser
```

```
  refresh, producing unpredictable behavior.
  (Low values aren't very useful though except for testing.)
* Defaults to 60

tools.sessions.restart_persist = [True | False]
* If set to False then the session cookie will be deleted from the
browser
  when the browser quits
* Defaults to True - Sessions persist across browser restarts
  (assuming the tools.sessions.timeout limit hasn't been reached)

tools.sessions.httponly = [True | False]
* If set to True then the session cookie will be made unavailable
  to running javascript scripts, increasing session security
* Defaults to True

tools.sessions.secure = [True | False]
* If set to True and Splunkweb is configured to server requests using
HTTPS
  (see the enableSplunkWebSSL setting) then the browser will only
transmit
  the session cookie over HTTPS connections, increasing session security
* Defaults to True

response.timeout = <integer>
* Specifies the number of seconds to wait for the server to complete a
  response
* Some requests such as uploading large files can take a long time
* Defaults to 7200

tools.sessions.storage_type = [file]
tools.sessions.storage_path = <filepath>
* Specifies the session information storage mechanisms
* Comment out the next two lines to use RAM based sessions instead
* Use an absolute path to store sessions outside of the splunk tree
* Defaults to storage_type=file, storage_path=var/run/splunk

tools.decode.on = [True | False]
* Indicates if all strings that come into Cherrpy controller methods are
  decoded as unicode (assumes UTF-8 encoding).
* WARNING: Disabling this will likely break the application, as all
incoming
           strings are assumed to be unicode.
* Defaults to True

tools.encode.on = [True | False]
* Encodes all controller method response strings into UTF-8 str objects
in
  Python.
* WARNING: Disabling this will likely cause high byte character encoding
to
  fail.
```

```
* Defaults to True

tools.encode.encoding = <codec>
* Force all outgoing characters to be encoded into UTF-8.
* This only works with tools.encode.on set to True.
* By setting this to utf-8, Cherrypy's default behavior of observing the
* Accept-Charset header is overwritten and forces utf-8 output. Only
change
  this if you know a particular browser installation must receive some
other
  character encoding (Latin-1 iso-8859-1, etc)
* WARNING: Change this at your own risk.
* Defaults to utf08

tools.proxy.on = [True | False]
* Used for running Apache as a proxy for Splunk UI, typically for SSO
  configuration. See http://tools.cherrypy.org/wiki/BehindApache for
more
  information.
* For Apache 1.x proxies only. Set this attribute to "true". This
  configuration instructs CherryPy (the Splunk Web HTTP server) to look
for
  an incoming X-Forwarded-Host header and to use the value of that
header to
  construct canonical redirect URLs that include the proper host name.
For
  more information, refer to the CherryPy documentation on running
behind an
  Apache proxy. This setting is only necessary for Apache 1.1 proxies.
For
  all other proxies, the setting must be "false", which is the default.
* Defaults to False

tools.proxy.base = <scheme>://<URL>
* Used for setting the proxy base url in Splunk
* Defaults to an empty value

pid_path = <filepath>
* Specifies the path to the PID file
* Equals precisely and only var/run/splunk/splunkweb.pid
* NOTE: Do not change this parameter.

enabled_decomposers = <intention> [, <intention>]...
* Added in Splunk 4.2 as a short term workaround measure for apps which
  happen to still require search decomposition, which is deprecated
  with 4.2.
* Search decomposition will be entirely removed in a future release.
* Comma separated list of allowed intentions.
* Modifies search decomposition, which is a splunk-web internal
behavior.
* Can be controlled on a per-app basis.
* If set to the empty string, no search decomposition occurs, which
```

```
causes
  some usability problems with report builder.
* The current possible values are: addcommand, stats, addterm,
addtermgt,
  addtermlt, setfields, excludefields, audit, sort, plot
* Default is 'plot', leaving only the plot intention enabled.
* When you need a good mulch, we recommend antibeethoven.
* However, for a traditional compost, antimozart is preferred.

simple_xml_perf_debug = [True | False]
* If True, simple xml dashboards will log some performance metrics to
the
  browser console
* Defaults to False

job_min_polling_interval = <integer>
* Minimum polling interval for job in miliseconds (ms)
* The default value is 100
* This is the intial time wait for fetching results
* The poll period increases gradually from min interval to max interval
when
  search is in queued state or parsing state (and not running state) for
a
  some time.
* Set this value between 100 to job_max_polling_interval

job_max_polling_interval = <integer>
* Maximum polling interval for job in miliseconds (ms)
* The default value is 1000
* This is the maximum time wait for fetching results
* The recommended maximum value is 3000

acceptFrom = <network_acl> ...
* IMPORTANT: this setting only takes effect when appServerPorts is set
to a
  non-zero value
* Lists a set of networks or addresses to accept connections from.
 These
  rules are separated by commas or spaces
* Each rule can be in the following forms:
    1. A single IPv4 or IPv6 address (examples: "10.1.2.3", "fe80::4a3")
    2. A CIDR block of addresses (examples: "10/8", "fe80:1234/32")
    3. A DNS name, possibly with a '*' used as a wildcard (examples:
       "myhost.example.com", "*.splunk.com")
    4. A single '*' which matches anything
* Entries can also be prefixed with '!' to cause the rule to reject the
  connection.  Rules are applied in order, and the first one to match is
  used.  For example, "!10.1/16, *" will allow connections from
everywhere
  except the 10.1.*.* network.
* Defaults to "*" (accept from anywhere)
```

```
maxThreads = <int>
* NOTE: this setting only takes effect when appServerPorts is set to a
        non-zero value
* Number of threads that can be used by active HTTP transactions.
  This can be limited to constrain resource usage.
* If set to 0 (the default) a limit will be automatically picked
  based on estimated server capacity.
* If set to a negative number, no limit will be enforced.

maxSockets = <int>
* NOTE: this setting only takes effect when appServerPorts is set to a
        non-zero value
* Number of simultaneous HTTP connections that we accept simultaneously.
  This can be limited to constrain resource usage.
* If set to 0 (the default) a limit will be automatically picked
  based on estimated server capacity.
* If set to a negative number, no limit will be enforced.

forceHttp10 = auto|never|always
* NOTE: this setting only takes effect when appServerPorts is set to a
  non-zero value
* When set to "always", the REST HTTP server will not use some
  HTTP 1.1 features such as persistent connections or chunked
  transfer encoding.
* When set to "auto" it will do this only if the client sent no
  User-Agent header, or if the user agent is known to have bugs
  in its HTTP/1.1 support.
* When set to "never" it always will allow HTTP 1.1, even to
  clients it suspects may be buggy.
* Defaults to "auto"

crossOriginSharingPolicy = <origin_acl> ...
* IMPORTANT: this setting only takes effect when appServerPorts is set
to a
  non-zero value
* List of HTTP Origins for which to return Access-Control-Allow-*
(CORS) headers
* These headers tell browsers that we trust web applications at those
sites
  to make requests to the REST interface
* The origin is passed as a URL without a path component (for example
  "https://app.example.com:8000")
* This setting can take a list of acceptable origins, separated
  by spaces and/or commas
* Each origin can also contain wildcards for any part.  Examples:
  *://app.example.com:*  (either HTTP or HTTPS on any port)
  https://*.example.com  (any host under example.com, including
example.com itself)
* An address can be prefixed with a '!' to negate the match, with
  the first matching origin taking precedence.  For example,
  "!*://evil.example.com:* *://*.example.com:*" to not avoid
  matching one host in a domain
```

```
* A single "*" can also be used to match all origins
* By default the list is empty

allowSslCompression = true|false
* IMPORTANT: this setting only takes effect when appServerPorts is set
  to a non-zero value.  When appServerPorts is zero or missing, this
setting
  will always act as if it is set to "true"
* If set to true, the server will allow clients to negotiate
  SSL-layer data compression.
* Defaults to false.  The HTTP layer has its own compression layer
  which is usually sufficient.

allowSslRenegotiation = true|false
* IMPORTANT: this setting only takes effect when appServerPorts is set
to a
  non-zero value
* In the SSL protocol, a client may request renegotiation of the
connection
  settings from time to time.
* Setting this to false causes the server to reject all renegotiation
  attempts, breaking the connection.  This limits the amount of CPU a
  single TCP connection can use, but it can cause connectivity problems
  especially for long-lived connections.
* Defaults to true

sendStrictTransportSecurityHeader = true|false
* IMPORTANT: this setting only takes effect when appServerPorts is set
to a
  non-zero value
* If set to true, the REST interface will send a
"Strict-Transport-Security"
  header with all responses to requests made over SSL.
* This can help avoid a client being tricked later by a
Man-In-The-Middle
  attack to accept a non-SSL request.  However, this requires a
commitment
  that no non-SSL web hosts will ever be run on this hostname on any
port.
  For example, if splunkweb is in default non-SSL mode this can break
the
  ability of browser to connect to it.  Enable with caution.
* Defaults to false

dedicatedIoThreads = <int>
* If set to zero, HTTP I/O will be performed in the same thread
  that accepted the TCP connection.
* If set set to a non-zero value, separate threads will be run
  to handle the HTTP I/O, including SSL encryption.
* Defaults to "0"
* Typically this does not need to be changed.  For most usage
  scenarios using the same the thread offers the best performance.
```

```
* NOTE: this setting only takes effect when appServerPorts is set to a
  non-zero value

termsOfServiceDirectory = <directory>
  * If set, we will look in this directory for a "Terms Of Service"
document
    which each user must accept before logging into the UI
  * Inside the directory the TOS should have a filename in the format
"<number>.html"
    Where <number> is in the range 1 to 18446744073709551615.  The
active TOS is the
    filename with the larger number.  For instance if there are two
files in the
    directory named "123.html" and "456.html", then 456 will be the
active TOS version.
  * If a user hasn't accepted the current version of the TOS, they'll be
required to
    the next time they try to log in.  The acceptance times will be
recorded inside
    a "tos.conf" file inside an app called "tos"
  * The TOS file can either be a full HTML document or plain text, but
it must have the
    ".html" suffix
  * It is not necessary to restart Splunk when adding files to the TOS
directory
  * Defaults to empty (no TOS)
  * NOTE: this setting only takes effect when appServerPorts is set to
a non-zero value

appServerProcessShutdownTimeout = <nonnegative integer>[smhd]
  * IMPORTANT: this setting only takes effect when appServerPorts is set
to a non-zero value.
  * The amount of time splunkd will wait "politely" for a Python-based
application server process to handle outstanding/existing requests.
  * If a Python-based application server process "outlives" this
timeout, the process is forcibly killed.
  * Defaults to '10m'

enableWebDebug = true|false
* Controls the visibility of the debug endpoints (i.e., /debug/**splat).
* Defaults to false

allowableTemplatePaths =  <directory> [, <directory>]...
* A comma separated list of template paths that may be added tp
template lookup white list.
* Paths are relative to $SPLUNK_HOME.
* Defaults to empty

enable_risky_command_check = <bool>
* Enable checks for data-exfiltrating search commands.
* default true
```

```
customFavicon = <pathToMyFile, myApp:pathToMyFile, or blank for
default>
* Customize the favicon image across the entire application. If no
favicon image file, the favicon defaults to the Splunk favicon.
  * Supported favicon image files are .ico files, and should be square
images.
  * Place favicon image file in default or manual location:
    * Default destination folder:
$SPLUNK_HOME/etc/apps/search/appserver/static/customfavicon.
      * Example: If your favicon image is located at
$SPLUNK_HOME/etc/apps/search/appserver/static/customfavicon/favicon.ico,
type customFavicon = customfavicon/favicon.ico.
    * Manual location:
$SPLUNK_HOME/etc/apps/<myApp>/appserver/static/<pathToMyFile>, and type
customFavicon = <myApp:pathToMyFile>.

loginCustomLogo = <fullUrl, pathToMyFile, myApp:pathToMyFile, or blank
for default>
* Customize the logo image on the login page. If no image file, the logo
defaults to the Splunk logo.
* Supported images are:
  * Full URL image file (secured or not secured), such as
https://www.splunk.com/logo.png or http://www.splunk.com/logo.png.
  * Image file, such as .jpg or .png. All image formats are supported.
    * Place logo image file in default or manual location:
      * Default destination folder:
$SPLUNK_HOME/etc/apps/search/appserver/static/logincustomlogo.
        * Example: If your logo image is located at
$SPLUNK_HOME/etc/apps/search/appserver/static/logincustomlogo/logo.png,
type loginCustomLogo = logincustomlogo/logo.png.
      * Manual location:
$SPLUNK_HOME/etc/apps/<myApp>/appserver/static/<pathToMyFile>, and type
loginCustomLogo = <myApp:pathToMyFile>.
* Logo height limit is 100px. Logo displays original dimensions unless
the image height is greater than 100px.
* If image height exceeds 100px, the image is resized and width
automatically adjusts.

loginBackgroundImageOption = [default| custom | none]
* Controls display of the background image of the login page.
* Defaults to "default".
  * "default" displays the Splunk default background image.
  * "custom" uses the background image defined by the
backgroundImageCustomName setting.
  * "none" removes any background image on the login page. A dark
background color is applied.

loginCustomBackgroundImage = <pathToMyFile or myApp:pathToMyFile>
* Customize the login page background image.
  * Supported image files include .jpg, .jpeg or .png with a maximum
file size of 20Mb.
  * Landscape image is recommended, with a minimum resolution of
```

```
1024x640 pixels.
  * Using Splunk Web:
    * Upload a custom image to a manager page under General Settings.
    * The login page background image updates automatically.
  * Using the CLI or a text editor:
    * Set loginBackgroundImageOption = "custom".
    * Place custom image file in default or manual location:
      * Default destination folder:
$SPLUNK_HOME/etc/apps/search/appserver/static/logincustombg.
        * Example: If your image is located at
$SPLUNK_HOME/etc/apps/search/appserver/static/logincustombg/img.png,
type loginCustomBackgroundImage = logincustombg/img.png.
      * Manual location:
$SPLUNK_HOME/etc/apps/<myApp>/appserver/static/<pathToMyFile>, and type
loginCustomBackgroundImage = <myApp:pathToMyFile>.
    * The login page background image updates automatically.
  * If no custom image is used, the default Splunk background image
displays.

[framework]
# Put App Framework settings here
django_enable = [True | False]
* Specifies whether Django should be enabled or not
* Defaults to True
* Django will not start unless an app requires it

django_path = <filepath>
* Specifies the root path to the new App Framework files,
  relative to $SPLUNK_HOME
* Defaults to etc/apps/framework

django_force_enable = [True | False]
* Specifies whether to force Django to start, even if no app requires it
* Defaults to False


#
# custom cherrypy endpoints
#

[endpoint:<python_module_name>]
* registers a custom python CherryPy endpoint
* The expected file must be located at:
  $SPLUNK_HOME/etc/apps/<APP_NAME>/appserver/controllers/<PYTHON_NODULE_NAME>.py
* This module's methods will be exposed at
/custom/<APP_NAME>/<PYTHON_NODULE_NAME>/<METHOD_NAME>

#
# exposed splunkd REST endpoints
#
[expose:<unique_name>]
* Registers a splunkd-based endpoint that should be made available to
```

```
the UI
  under the "/splunkd" and "/splunkd/__raw" hierarchies
* The name of the stanza doesn't matter as long as it starts with
"expose:"
  Each stanza name must be unique, however

pattern = <url_pattern>
* Pattern to match under the splunkd /services hierarchy.  For instance,
  "a/b/c" would match URIs "/services/a/b/c" and
"/servicesNS/*/*/a/b/c"
* The pattern should not include leading or trailing slashes
* Inside the pattern an element of "*" will match a single path
element.
  For example, "a/*/c" would match "a/b/c" but not "a/1/2/c"
* A path element of "**" will match any number of elements.  For
example,
  "a/**/c" would match both "a/1/c" and "a/1/2/3/c"
* A path element can end with a "*" to match a prefix.  For example,
  "a/elem-*/b" would match "a/elem-123/c"

methods = <method_lists>
* Comma separated list of methods to allow from the web browser
  (example: "GET,POST,DELETE")
* If not included, defaults to "GET"

oidEnabled = [0 | 1]
* If set to 1 indicates that the endpoint is capable of taking an
embed-id
  as a query parameter
* Defaults to 0
* This is only needed for some internal splunk endpoints, you probably
  should not specify this for app-supplied endpoints

skipCSRFProtection = [0 | 1]
* If set to 1, tells splunkweb that it is safe to post to this endpoint
  without applying CSRF protection
* Defaults to 0
* This should only be set on the login endpoint (which already contains
  sufficient auth credentials to avoid CSRF problems)
```

web.conf.example

```
#   Version 6.5.0
#
# This is an example web.conf.  Use this file to configure data web
# settings.
#
# To use one or more of these configurations, copy the configuration
block
```

```
# into web.conf in $SPLUNK_HOME/etc/system/local/. You must restart
Splunk
# to enable configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles



# This stanza heading must precede any changes.
[settings]

# Change the default port number:
httpport = 12800
# Also run the python application server on a non-default port:
appServerPorts = 12801

# Turn on SSL:
enableSplunkWebSSL = true
# absolute paths may be used here.
privKeyPath = /home/user/certs/myprivatekey.pem
caCertPath = /home/user/certs/mycacert.pem
# NOTE: non-absolute paths are relative to $SPLUNK_HOME
```

wmi.conf

The following are the spec and example files for wmi.conf.

wmi.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attribute/value pairs for configuring
Windows
# Management Instrumentation (WMI) access from Splunk.
#
# There is a wmi.conf in $SPLUNK_HOME\etc\system\default\.  To set
custom
# configurations, place a wmi.conf in $SPLUNK_HOME\etc\system\local\.
For
# examples, see wmi.conf.example.
#
# You must restart Splunk to enable configurations.
```

```
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS-----

```
##########################################################################
#----GLOBAL SETTINGS-----
##########################################################################GLOBAL
SETTINGS-----

[settings]
* The settings stanza specifies various runtime parameters.
* The entire stanza and every parameter within it is optional.
* If the stanza is missing, Splunk assumes system defaults.

initial_backoff = <integer>
* How long, in seconds, to wait before retrying the connection to
  the WMI provider after the first connection error.
* If connection errors continue, the wait time doubles until it reaches
  the integer specified in max_backoff.
* Defaults to 5.

max_backoff = <integer>
* The maximum time, in seconds, to attempt to reconnect to the
  WMI provider.
* Defaults to 20.

max_retries_at_max_backoff = <integer>
* Once max_backoff is reached, tells Splunk how many times to attempt
  to reconnect to the WMI provider.
* Splunk will try to reconnect every max_backoff seconds.
* If reconnection fails after max_retries, give up forever (until
restart).
* Defaults to 2.

checkpoint_sync_interval = <integer>
* The minimum wait time, in seconds, for state data (event log
checkpoint)
  to be written to disk.
* Defaults to 2.
```

INPUT-SPECIFIC SETTINGS-----

```
#####################################################################
#----INPUT-SPECIFIC SETTINGS-----
#####################################################################INPUT-SPECIFIC
SETTINGS-----

[WMI:$NAME]
* There are two types of WMI stanzas:
  * Event log: for pulling event logs. You must set the
    event_log_file attribute.
  * WQL: for issuing raw Windows Query Language (WQL) requests. You
    must set the wql attribute.
* Do not use both the event_log_file or the wql attributes.  Use
  one or the other.

server = <comma-separated strings>
* A comma-separated list of servers from which to get data.
* If not present, defaults to the local machine.

interval = <integer>
* How often, in seconds, to poll for new data.
* This attribute is required, and the input will not run if the
attribute is
  not present.
* There is no default.

disabled = [0|1]
* Specifies whether the input is enabled or not.
* 1 to disable the input, 0 to enable it.
* Defaults to 0 (enabled).

hostname = <host>
* All results generated by this stanza will appear to have arrived from
  the string specified here.
* This attribute is optional.
* If it is not present, the input will detect the host automatically.

current_only = [0|1]
* Changes the characteristics and interaction of WMI-based event
  collections.
* When current_only is set to 1:
  * For event log stanzas, this will only capture events that occur
    while Splunk is running.
  * For WQL stanzas, event notification queries are expected.  The
    queried class must support sending events.  Failure to supply
    the correct event notification query structure will cause
    WMI to return a syntax error.
  * An example event notification query that watches for process
creation:
```

```
    * SELECT * FROM __InstanceCreationEvent WITHIN 1 WHERE
      TargetInstance ISA 'Win32_Process'.
* When current_only is set to 0:
  * For event log stanzas, all the events from the checkpoint are
    gathered. If there is no checkpoint, all events starting from
    the oldest events are retrieved.
  * For WQL stanzas, the query is executed and results are retrieved.
    The query is a non-notification query.
  * For example
    * Select * Win32_Process where caption = "explorer.exe"
* Defaults to 0.

batch_size = <integer>
*  Number of events to fetch on each query.
* Defaults to 10.

index = <string>
* Specifies the index that this input should send the data to.
* This attribute is optional.
* When defined, "index=" is automatically prepended to <string>.
* Defaults to "index=main" (or whatever you have set as your default
index).
```

Event log-specific attributes:

```
#####
# Event log-specific attributes:
#####Event log-specific attributes:

event_log_file = <Application, System, etc>
* Tells Splunk to expect event log data for this stanza, and specifies
  the event log channels you want Splunk to monitor.
* Use this instead of WQL to specify sources.
* Specify one or more event log channels to poll.  Multiple event log
  channels must be separated by commas.
* There is no default.

disable_hostname_normalization = [0|1]
* If set to true, hostname normalization is disabled
* If absent or set to false, the hostname for 'localhost' will be
converted
  to %COMPUTERNAME%.
* 'localhost' refers to the following list of strings: localhost,
127.0.0.1,
  ::1, the name of the DNS domain for the local computer, the fully
  qualified DNS name, the NetBIOS name, the DNS host name of the local
  computer
```

WQL-specific attributes:

```
#####
# WQL-specific attributes:
#####WQL-specific attributes:

wql = <string>
* Tells Splunk to expect data from a WMI provider for this stanza, and
  specifies the WQL query you want Splunk to make to gather that data.
* Use this if you are not using the event_log_file attribute.
* Ensure that your WQL queries are syntactically and structurally
correct
  when using this option.
* For example,
  SELECT * FROM Win32_PerfFormattedData_PerfProc_Process WHERE Name =
"splunkd".
* If you wish to use event notification queries, you must also set the
  "current_only" attribute to 1 within the stanza, and your query must
be
  appropriately structured for event notification (meaning it must
contain
  one or more of the GROUP, WITHIN or HAVING clauses.)
* For example,
  SELECT * FROM __InstanceCreationEvent WITHIN 1 WHERE TargetInstance
ISA
  'Win32_Process'
* There is no default.

namespace = <string>
* The namespace where the WMI provider resides.
* The namespace spec can either be relative (root\cimv2) or absolute
  (\\server\root\cimv2).
* If the server attribute is present, you cannot specify an absolute
  namespace.
* Defaults to root\cimv2.
```

wmi.conf.example

```
#   Version 6.5.0
#
# This is an example wmi.conf.  These settings are used to control
inputs
# from WMI providers. Refer to wmi.conf.spec and the documentation at
# splunk.com for more information about this file.
#
# To use one or more of these configurations, copy the configuration
```

```
block
# into wmi.conf in $SPLUNK_HOME\etc\system\local\.  You must restart
Splunk
# to enable configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

# This stanza specifies runtime parameters.

[settings]
initial_backoff = 5
max_backoff = 20
max_retries_at_max_backoff = 2
checkpoint_sync_interval = 2

# Pull events from the Application, System and Security event logs from
the
# local system every 10 seconds. Store the events in the "wmi_eventlog"
# Splunk index.

[WMI:LocalApplication]
interval = 10
event_log_file = Application
disabled = 0
index = wmi_eventlog

[WMI:LocalSystem]
interval = 10
event_log_file = System
disabled = 0
index = wmi_eventlog

[WMI:LocalSecurity]
interval = 10
event_log_file = Security
disabled = 0
index = wmi_eventlog

# Gather disk and memory performance metrics from the local system every
# second.  Store event in the "wmi_perfmon" Splunk index.

[WMI:LocalPhysicalDisk]
interval = 1
wql = select Name, DiskBytesPerSec, PercentDiskReadTime,
PercentDiskWriteTime, PercentDiskTime from
Win32_PerfFormattedData_PerfDisk_PhysicalDisk
disabled = 0
index = wmi_perfmon
```

```
[WMI:LocalMainMemory]
interval = 10
wql = select CommittedBytes, AvailableBytes,
PercentCommittedBytesInUse, Caption from
Win32_PerfFormattedData_PerfOS_Memory
disabled = 0
index = wmi_perfmon

# Collect all process-related performance metrics for the splunkd
process,
# every second.  Store those events in the "wmi_perfmon" index.
[WMI:LocalSplunkdProcess]
interval = 1
wql = select * from Win32_PerfFormattedData_PerfProc_Process where Name
= "splunkd"
disabled = 0
index = wmi_perfmon

# Listen from three event log channels, capturing log events that occur
only
# while Splunk is running, every 10 seconds.  Gather data from three
remote
# servers srv1, srv2 and srv3.

[WMI:TailApplicationLogs]
interval = 10
event_log_file = Application, Security, System
server = srv1, srv2, srv3
disabled = 0
current_only = 1
batch_size = 10

# Listen for process-creation events on a remote machine, once a
second.

[WMI:ProcessCreation]
interval = 1
server = remote-machine
wql = select * from __InstanceCreationEvent within 1 where
TargetInstance isa 'Win32_Process'
disabled = 0
current_only = 1
batch_size = 10

# Receive events whenever someone connects or removes a USB device on
# the computer, once a second.

[WMI:USBChanges]
interval = 1
wql = select * from __InstanceOperationEvent within 1 where
TargetInstance ISA 'Win32_PnPEntity' and
```

```
TargetInstance.Description='USB Mass Storage Device'
disabled = 0
current_only = 1
batch_size = 10
```

workflow_actions.conf

The following are the spec and example files for workflow_actions.conf.

workflow_actions.conf.spec

```
#   Version 6.5.0
#
# This file contains possible attribute/value pairs for configuring
workflow
# actions in Splunk.
#
# There is a workflow_actions.conf in
$SPLUNK_HOME/etc/apps/search/default/.
# To set custom configurations, place a workflow_actions.conf in either
# $SPLUNK_HOME/etc/system/local/ or add a workflow_actions.conf file to
your
# app's local/ directory. For examples, see
workflow_actions.conf.example.
# You must restart Splunk to enable configurations, unless editing them
# through the Splunk manager.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles
```

GLOBAL SETTINGS

```
# GLOBAL SETTINGS
# Use the [default] stanza to define any global settings.
#  * You can also define global settings outside of any stanza, at the
top
#    of the file.
#  * Each conf file should have at most one default stanza. If there
are
#    multiple default stanzas, attributes are combined. In the case of
```

```
#    multiple definitions of the same attribute, the last definition in
the
#    file wins.
#  * If an attribute is defined at both the global level and in a
specific
#    stanza, the value in the specific stanza takes precedence.

###############################################################################
# General required settings:
# These apply to all workflow action types.
###############################################################################

type = <string>
* The type of the workflow action.
* If not set, Splunk skips this workflow action.

label = <string>
* The label to display in the workflow action menu.
* If not set, Splunk skips this workflow action.

###############################################################################
# General optional settings:
# These settings are not required but are available for all workflow
# actions.
###############################################################################

fields = <comma or space separated list>
* The fields required to be present on the event in order for the
workflow
  action to be applied.
* When "display_location" is set to "both" or "field_menu", the
workflow
  action will be applied to the menu's corresponding to the specified
  fields.
* If fields is undefined or set to *, the workflow action is applied to
all
  field menus.
* If the * character is used in a field name, it is assumed to act as a
  "globber". For example host* would match the fields hostname, hostip,
etc.
* Acceptable values are any valid field name, any field name including
the *
  character, or * (e.g. *_ip).
* Defaults to *

eventtypes = <comma or space separated list>
* The eventtypes required to be present on the event in order for the
  workflow action to be applied.
* Acceptable values are any valid eventtype name, or any eventtype name
plus
  the * character (e.g. host*).
```

```
display_location = <string>
* Dictates whether to display the workflow action in the event menu, the
  field menus or in both locations.
* Accepts field_menu, event_menu, or both.
* Defaults to both.

disabled = [True | False]
* Dictates whether the workflow action is currently disabled
* Defaults to False
```

Using field names to insert values into workflow action settings

```
##############################################################################
# Using field names to insert values into workflow action settings
##############################################################################Using
field names to insert values into workflow action settings

# Several settings detailed below allow for the substitution of field
values
# using a special variable syntax, where the field's name is enclosed in
# dollar signs.  For example, $_raw$, $hostip$, etc.
#
# The settings, label, link.uri, link.postargs, and
search.search_string all
# accept the value of any valid field to be substituted into the final
# string.
#
# For example, you might construct a Google search using an error
message
# field called error_msg like so:
# link.uri = http://www.google.com/search?q=$error_msg$.
#
# Some special variables exist to make constructing the settings
simpler.

$@field_name$
* Allows for the name of the current field being clicked on to be used
in a
  field action.
* Useful when constructing searches or links that apply to all fields.
* NOT AVAILABLE FOR EVENT MENUS

$@field_value$
* Allows for the value of the current field being clicked on to be used
in a
  field action.
* Useful when constructing searches or links that apply to all fields.
* NOT AVAILABLE FOR EVENT MENUS
```

```
$@sid$
* The sid of the current search job.

$@offset$
* The offset of the event being clicked on in the list of search
events.

$@namespace$
* The name of the application from which the search was run.

$@latest_time$
* The latest time the event occurred.  This is used to disambiguate
similar
  events from one another. It is not often available for all fields.
```

Field action types

```
################################################################################
# Field action types
################################################################################Field
action types

################################################################################
# Link type:
# Allows for the construction of GET and POST requests via links to
external
# resources.
################################################################################

link.uri = <string>
* The URI for the resource to link to.
* Accepts field values in the form $<field name>$, (e.g $_raw$).
* All inserted values are URI encoded.
* Required

link.target = <string>
* Determines if clicking the link opens a new window, or redirects the
  current window to the resource defined in link.uri.
* Accepts: "blank" (opens a new window), "self" (opens in the same
window)
* Defaults to "blank"

link.method = <string>
* Determines if clicking the link should generate a GET request or a
POST
  request to the resource defined in link.uri.
* Accepts: "get" or "post".
* Defaults to "get".
```

```
link.postargs.<int>.<key/value> = <value>
* Only available when link.method = post.
* Defined as a list of key / value pairs like such that foo=bar becomes:
  link.postargs.1.key = "foo"
  link.postargs.1.value = "bar"
* Allows for a conf compatible method of defining multiple identical
keys (e.g.):
  link.postargs.1.key = "foo"
  link.postargs.1.value = "bar"
  link.postargs.2.key = "foo"
  link.postargs.2.value = "boo"
  ...
* All values are html form encoded appropriately.

###########################################################################
# Search type:
# Allows for the construction of a new search to run in a specified
view.
###########################################################################

search.search_string = <string>
* The search string to construct.
* Accepts field values in the form $<field name>$, (e.g. $_raw$).
* Does NOT attempt to determine if the inserted field values may break
  quoting or other search language escaping.
* Required

search.app = <string>
* The name of the Splunk application in which to perform the
constructed
  search.
* By default this is set to the current app.

search.view = <string>
* The name of the view in which to preform the constructed search.
* By default this is set to the current view.

search.target = <string>
* Accepts: blank, self.
* Works in the same way as link.target. See link.target for more info.

search.earliest = <time>
* Accepts absolute and Splunk relative times (e.g. -10h).
* Determines the earliest time to search from.

search.latest = <time>
* Accepts absolute and Splunk relative times (e.g. -10h).
* Determines the latest time to search to.

search.preserve_timerange = <boolean>
* Ignored if either the search.earliest or search.latest values are
```

```
set.
* When true, the time range from the original search which produced the
  events list will be used.
* Defaults to false.
```

workflow_actions.conf.example

```
#   Version 6.5.0
#
# This is an example workflow_actions.conf.  These settings are used to
# create workflow actions accessible in an event viewer.  Refer to
# workflow_actions.conf.spec and the documentation at splunk.com for
more
# information about this file.
#
# To use one or more of these configurations, copy the configuration
block
# into workflow_actions.conf in $SPLUNK_HOME/etc/system/local/, or into
your
# application's local/ folder.  You must restart Splunk to enable
# configurations.
#
# To learn more about configuration files (including precedence) please
see
# the documentation located at
#
http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles

# These are the default workflow actions and make extensive use of the
# special parameters: $@namespace$, $@sid$, etc.

[show_source]
type=link
fields = _cd, source, host, index
display_location = event_menu
label = Show Source
link.uri =
/app/$@namespace$/show_source?sid=$@sid$&offset=$@offset$&latest_time=$@latest_time$

[ifx]
type = link
display_location = event_menu
label = Extract Fields
link.uri = /ifx?sid=$@sid$&offset=$@offset$&namespace=$@namespace$

[etb]
type = link
display_location = event_menu
label = Build Eventtype
```

```
link.uri = /etb?sid=$@sid$&offset=$@offset$&namespace=$@namespace$

# This is an example workflow action which will be displayed in a
specific
# field menu (clientip).

[whois]
display_location = field_menu
fields = clientip
label = Whois: $clientip$
link.method = get
link.target = blank
link.uri = http://ws.arin.net/whois/?queryinput=$clientip$
type = link

# This is an example field action which will allow a user to search
every
# field value in Google.

[Google]
display_location = field_menu
fields = *
label = Google $@field_name$
link.method = get
link.uri = http://www.google.com/search?q=$@field_value$
type = link

# This is an example post link that will send its field name and field
value
# to a  fictional bug tracking system.

[Create JIRA issue]
display_location = field_menu
fields = error_msg
label = Create JIRA issue for $error_class$
link.method = post
link.postargs.1.key = error
link.postargs.1.value = $error_msg$
link.target = blank
link.uri = http://127.0.0.1:8000/jira/issue/create
type = link

# This is an example search workflow action that will be displayed in
an
# event's menu, but requires the field "controller" to exist in the
event in
# order for the workflow action to be available for that event.

[Controller req over time]
display_location = event_menu
fields = controller
label = Requests over last day for $controller$
```

```
search.earliest = -3d
search.search_string = sourcetype=rails_app controller=$controller$ |
timechart span=1h count
search.target = blank
search.view = charting
type = search
```