Splunk Admin Manual

Version: 4.1.5

Generated: 10/06/2010 09:52 am

Table of Contents

Welcome to Splunk administration 1
- What's in this manual 1
- What is Splunk? 1

What to do first 3
- Start Splunk 3
- Configure Splunk to start at boot time 5
- Find Splunk Manager in Splunk Web 6
- Install your license 6
- Change default values 9
- Bind Splunk to an IP 12

Meet Splunk Web and Splunk apps 13
- What's Splunk Web? 13
- What are apps and add-ons? 13
- Where to get more apps and add-ons 16
- App architecture and object ownership 17
- Manage app and add-on objects 19

How to configure Splunk 21
- Splunk configuration methods 21
- About Splunk Manager 22
- About configuration files 23
- Configuration file precedence 27
- Attribute precedence within a single props.conf file 32

Indexing with Splunk 34
- What's a Splunk index? 34
- How indexing works 34
- Index time versus search time 37
- Advanced indexing strategy 38

Add data and configure inputs 40
- How to get your data into Splunk 40
- Specify input paths with wildcards 42
- Monitor files and directories 44
- Monitor files and directories using the CLI 50
- Monitor network ports 51
- Considerations for deciding how to monitor remote Windows data 56
- Monitor Windows event log data 58
- Monitor Windows Registry data 61
- Monitor WMI data 64
- Monitor Active Directory 68
- Monitor FIFO queues 75
- Monitor changes to your filesystem 77

Table of Contents

Add data and configure inputs
- Find more things to monitor with crawl ... 82
- Send SNMP events to Splunk ... 84
- Set up custom (scripted) inputs ... 85
- Whitelist or blacklist specific incoming data ... 88
- How log file rotation is handled ... 90

Set up forwarding and receiving ... 92
- About forwarding and receiving ... 92
- Enable forwarding and receiving ... 96
- Configure forwarders with outputs.conf ... 101
- Consolidate data from multiple machines ... 105
- Set up load balancing ... 106
- Route and filter data ... 109
- Clone data ... 115
- Forward data to third-party systems ... 117
- Encrypt and authenticate data with SSL ... 120
- More about forwarders ... 123

Configure event processing ... 126
- Overview of event processing ... 126
- Configure character set encoding ... 126
- Configure linebreaking for multi-line events ... 128
- Handle event timestamps ... 131
- Extract default fields automatically ... 132
- Improve data compression with segmentation ... 132
- Assign metadata to events dynamically ... 133
- Anonymize data with sed ... 134
- Anonymize data using configuration files ... 136

Configure event timestamping ... 138
- How timestamp assignment works ... 138
- Configure timestamp recognition ... 139
- Improve Splunk's ability to recognize timestamps ... 143
- Configure timestamp assignment for events with multiple timestamps ... 147
- Specify timezones of timestamps ... 148
- Tune timestamp recognition for better indexing performance ... 149

Configure indexed field extraction ... 151
- About default fields (host, source, sourcetype, and more) ... 151
- Set a default host for a Splunk server ... 155
- Set a default host for a file or directory input ... 157
- Override default host values based on event data ... 160
- Handle incorrectly-assigned host values ... 162
- Override automatic source type assignment ... 162

Table of Contents

Configure indexed field extraction
- Advanced source type overrides ... 165
- Configure rule-based source type recognition ... 167
- Rename source types ... 168
- Train Splunk's source type autoclassifier ... 169
- List of pretrained source types ... 170
- Specify source type settings in props.conf ... 174
- Configure index-time field extractions ... 175
- Extract fields from file headers at index time ... 183

Add and manage users ... 188
- About users and roles ... 188
- Add users and assign roles ... 189
- Set up user authentication with Splunk ... 193
- Set up user authentication with LDAP ... 194
- Set up user authentication with external systems ... 201
- Use single sign-on (SSO) with Splunk ... 206
- Delete user accounts using the CLI ... 209
- User language and locale ... 209
- Configure user session timeouts ... 210

Manage indexes ... 212
- About managing indexes ... 212
- Set up multiple indexes ... 212
- Set limits on disk usage ... 217
- How Splunk stores indexes ... 219
- Configure segmentation to manage disk usage ... 225
- Configure custom segmentation for a host, source, or source type ... 227
- Move an index ... 229
- Remove indexed data from Splunk ... 230
- Optimize indexes ... 232

Define alerts ... 234
- How alerting works ... 234
- Set up alerts in savedsearches.conf ... 235
- Configure scripted alerts ... 242
- Send SNMP traps to other systems ... 244

Set up backups and retention policies ... 246
- What you can back up ... 246
- How much space you will need ... 246
- Back up indexed data ... 247
- Back up configuration information ... 249
- Set a retirement and archiving policy ... 249
- Archive indexed data ... 251

Table of Contents

Set up backups and retention policies
Restore archived indexed data....253

Configure data security....255
What you can secure with Splunk....255
Secure access to Splunk with HTTPS....256
Secure access to your Splunk server with SSL....258
Distribute certificates to your search peers....265
Configure archive signing....265
Configure IT data block signing....267
Cryptographically sign audit events....270
Audit Splunk activity....271
Configure event hashing....273
Hardening standards....276

Deploy to other Splunk instances....280
About deployment server....280
Plan a deployment....282
Define server classes....283
Configure deployment clients....291
Deploy in multi-tenant environments....294
Deploy apps and configurations....296
Example: deploy a light forwarder....297
Extended example: deploy several standard forwarders....298
Example: add an input to forwarders....305

Set up distributed search....307
What is distributed search?....307
Install a dedicated search head....309
Configure distributed search....310
Use distributed search....315

Manage search jobs....316
About jobs and job management....316
Manage jobs in Splunk Web....317
Manage jobs in the OS....318

Use Splunk's command line interface (CLI)....320
About the CLI....320
Get help with the CLI....321
Use the CLI to administer a remote Splunk server....322
CLI admin commands....324

Table of Contents

Configuration file reference 326

admon.conf 326
alert_actions.conf 327
app.conf 330
audit.conf 333
authentication.conf 336
authorize.conf 342
commands.conf 346
crawl.conf 349
default.meta.conf 351
deploymentclient.conf 352
distsearch.conf 354
eventdiscoverer.conf 357
event_renderers.conf 359
eventtypes.conf 360
fields.conf 361
indexes.conf 363
inputs.conf 368
limits.conf 379
literals.conf 389
macros.conf 390
multikv.conf 392
outputs.conf 395
pdf_server.conf 406
procmon-filters.conf 407
props.conf 408
pubsub.conf 419
regmon-filters.conf 421
report_server.conf 422
restmap.conf 426
savedsearches.conf 429
searchbnf.conf 434
segmenters.conf 437
server.conf 439
serverclass.conf 445
serverclass.seed.xml.conf 449
source-classifier.conf 450
sourcetypes.conf 451
sysmon.conf 452
tags.conf 453
tenants.conf 454
times.conf 456
transactiontypes.conf 458
transforms.conf 461
user-seed.conf 467

Table of Contents

Configuration file reference
web.conf....467
wmi.conf....473
workflow_actions.conf....476

Troubleshooting....481
Splunk log files....481
Work with metrics.log....484
Contact Support....486
Anonymize data samples to send to support....490
Not finding the events you're looking for?....492
SuSE Linux: unable to get a properly formatted response from the server....493
Command line tools for use with Support's direction....494
Troubleshooting configurations....496

Welcome to Splunk administration

What's in this manual

What's in this manual

This manual contains information and procedures for the **Splunk administrator.** If you're responsible for configuring, running, and maintaining Splunk as a service for yourself or other users, this manual is for you.

For example, learn how to:

- add data inputs to Splunk
- add users and set up roles
- configure data security
- back up index and user data
- manage the search jobs your users run
- configure and manage mid- to large-scale deployments

and much more.

Where is all the information about event types and source types etc?

For this all-new version of Splunk, we are trying something different--we've broken out all the information about Splunk knowledge into a separate manual just for the person who handles that information. If that person is you, check the **Knowledge Manager Manual**. Let us know what you think!

Looking for help with searching in Splunk?

Check out the **User Manual** and the **Search Reference Manual** for all things search. In particular, you might want to check out the **Search Cheatsheet** if you're looking for a quick list of common examples.

Make a PDF

If you'd like a PDF of any version of this manual, click the **pdf version** link above the table of contents bar on the left side of this page. A PDF version of the manual is generated on the fly for you, and you can save it or print it out to read later.

What is Splunk?

What is Splunk?

Splunk is an IT search engine.

- You can use Splunk to search and navigate IT data from applications, servers, and network devices in real-time.
- Data sources include logs, configurations, messages, alerts, scripts, code, metrics, etc.

- Splunk lets you search, navigate, alert, and report on all your IT data in real-time using Splunk Web.

Learn more about what Splunk is, what it does, and how it's different.

What to do first

Start Splunk

Start Splunk

This topic provides a brief instruction for starting Splunk. If you are new to Splunk, we recommend reviewing the User Manual first.

Start Splunk on Windows

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

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

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

- Server daemon: `splunkd`
- Web interface: `splunkweb`

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

- Server daemon: `splunkd`
- Web interface: `splunkweb`

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

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

Start Splunk on UNIX

Start Splunk

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

```
# splunk start
```

This starts both `splunkd` (indexer and other back-end processes) and `splunkweb` (the Splunk Web interface). To start them individually, type:

```
# splunk start splunkd
```

or

```
# splunk start splunkweb
```

Note: If `startwebserver` is disabled in `web.conf`, manually starting `splunkweb` does not override that setting. If it is disabled in the configuration file, it will not start.

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

```
# splunk restart

# splunk restart splunkd

# splunk restart splunkweb
```

Stop Splunk

To shut down Splunk, run this command:

```
# splunk stop
```

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

```
# splunk stop splunkd
```

or

```
# splunk stop splunkweb
```

Check if Splunk is running

To check if Splunk 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).
splunkweb is running (PID: 3216).
```

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

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

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

Solaris users, type `-ef` instead of `aux`:

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

Configure Splunk to start at boot time

Configure Splunk to start at boot time

On Windows, Splunk starts by default at machine startup. On other platforms, you must configure this manually. To disable this, see the end of this topic.

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
```

Disabling boot-start on Windows

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

Find Splunk Manager in Splunk Web

Find Splunk Manager in Splunk Web

Launch Splunk Web

Navigate to:

```
http://mysplunkhost:8000
```

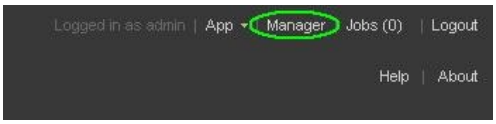
Use whatever host and port you chose during installation.

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.

Find Splunk Manager

Splunk Web provides a convenient interface for managing most aspects of Splunk operations: **Manager**. To access **Manager**, look for the link in the upper right corner of Splunk Web:



Learn more about using Manager to configure and maintain your Splunk installation.

Install your license

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.

Important: You cannot use the same Enterprise license on multiple servers. Each instance of Splunk (including forwarders) must have its own unique license, whether a Free license or an Enterprise license. The only exception to this is the 1 MB/day forward-only license that can be installed on multiple forwarding instances. For more information, read About Splunk licenses.

Access your license

All Splunk servers have a license located in `$SPLUNK_HOME/etc/`, whether it is a Free license (`splunk-free.license`) or an Enterprise license (`splunk.license`).

Example of a Splunk license

```
user@company.com;EQ/GQXW/J7u9VLJShPsW4m8yi+5a+geRrof4Bep70j32xsBpq
JItM5pdntRfl4auply366BAjTMnfTB6JyzJOZLplyBQijk02fQjgKjakl0ol4N5G6Wr
09ufnSe3iOXVAay24hzFfgDkaijOnkoGOPJqnHaVzaWC9dxIuKUvDPt3UcKTkDv0Gka
Q4EZxAvZKAFImvOF4PmDoNaMiBgLLkWibGhezFTTDh10PLl9kyeVThGzAyN23J512pVM
3xqNIg3pFcd2aJf31xspt1HRdSwofkfnuCVpzildy3qMbae4g85KpCfND+aJ6z2LoUu3
RQ4OV4SpxMXEZ4PgSGZ6dwA==
```

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 a new license (or change and update your existing license), replace your existing license with the new license.

You can install and update your licenses from Splunk Web's **Manager > License** page or with the CLI.

Install via Splunk Web

To install or update your license using Splunk Web:

1. Start Splunk and open Splunk Web in a supported browser.

2. On the upper right corner of any of the dashboards, click **Manager**.

3. Click **License**.

The *License & Usage'* page displays your license level, peak usage and license violations.

4. Click **Change License**.

The **Change License** page opens and displays your existing license key or `splunk.license` file.

5. Copy your new license key and paste (overwrite) the existing license.

6. Click **Save**.

7. Restart your Splunk server to apply your new license.

Note: You can restart your server from Splunk Web. On the **Manager > Server controls** page, click **Restart Splunk**.

Install via CLI

To install or update your license using the CLI:

1. Create a new file named `splunk.license`.

2. Copy your new license key and paste it into `splunk.license`.

3. Move your license file, `splunk.license`, into the `$SPLUNK_HOME/etc/` directory:

```
mv splunk.license $SPLUNK_HOME/etc/
```

Note: If a `splunk.license` file already exists in this directory, `mv` will overwrite it without prompting for confirmation of the action. This does not overwrite the Free license, `splunk-free.license`. However, by default Splunk ignores the Free license file if `splunk.license` exists.

4. Restart your Splunk server to apply your new license:

```
$SPLUNK_HOME/bin/splunk restart
```

First login after applying new trial or Enterprise license

To log in for the first time after applying an Enterprise license (converting from free), use the default username "admin" with the password "changeme". If you later clean (reset) your user data, your username/password is reset to this default.

License violations

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 will get a violation warning. The message persists for 14 days. **If you have 5 or more violations on an Enterprise license (3 on a Free license) in a rolling 30-day period, search will be disabled.** Search capabilities return when you have fewer than 5 (3 for Free) violations in the previous 30 days or when you apply a new license with a larger volume limit.

Note: During a license violation period, Splunk does not stop indexing your data. Splunk only blocks access while you exceed your license.

Report on daily indexed volume

For a report on the daily indexed volume, use this search:

```
index=_internal todaysBytesIndexed LicenseManager-Audit NOT
source=*web_service.log | eval Daily_Indexing_Volume_in_MBs =
todaysBytesIndexed/1024/1024 | timechart avg(Daily_Indexing_Volume_in_MBs)
by host
```

Change default values

Change default values

Before you begin configuring Splunk for your environment, check through the following default settings to see if there's anything you'd like to change.

Changing the admin default password

Splunk with an Enterprise license has a default administration account and password, admin/changeme. Splunk recommends strongly that you change the default. You can do this via Splunk's CLI or Splunk Web.

via Splunk Web

- Log into Splunk Web as the admin user.
- Click **Manager** in the top-right of the interface.
- Click **Access controls**.
- Click **Users**.
- Click the **admin** user.
- Update the password and click **Save**.

via Splunk CLI

The Splunk CLI command is:

```
# splunk edit user
```

Note: You must authenticate with the existing password before it can be changed. Log into Splunk via the CLI or use the `-auth` parameter.

For example:

```
# splunk edit user admin -password foo -roles admin -auth admin:changeme
```

This command changes the admin password from *changeme* to *foo*.

NB: Passwords with special characters that would be interpreted by the shell (for example '$' or '!') must be either escaped or single-quoted:

```
./splunk edit user admin -password 'fflanda$' -role admin -auth
admin:changeme
```

or

```
./splunk edit user admin -password fflanda\$ -role admin -auth
admin:changeme
```

Change network ports

Splunk uses two ports. They default to:

- 8000 - HTTP or HTTPS socket for Splunk Web.
- 8089 - Splunkd management port. 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.

Note: You may have changed these ports at install time.

via Splunk Web

- Log into Splunk Web as the admin user.
- Click **Manager** in the top-right of the interface.
- Click the **System Configuration** tab.
- Click **System Settings**.
- Change the value for **Web port** and click **Save**.

via Splunk CLI

To change the port settings via the Splunk CLI, use the CLI command `set`.

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

This command sets the Splunk Web port to 9000.

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

This command sets the splunkd port to 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.

via Splunk Web

- Log into Splunk Web as the admin user.
- Click **Manager** in the top-right of the interface.
- Click the **System Configuration** tab.
- Click **System Settings**.
- Change the value for **Splunk server name** and click **Save**.

via Splunk CLI

To change the server name via the CLI, type the following:

```
# splunk set servername foo
```

This command sets the servername to *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.

via Splunk Web

- Log into Splunk Web as the admin user.
- Click **Manager** in the top-right of the interface.
- Click the **System Configuration** tab.
- Click **System Settings**.
- Change the path in **Path to indexes** and click **Save**.

via Splunk CLI

To change the datastore directory via the CLI, type the following:

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

This command sets the datastore directory to `/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 stops indexing.

Splunk resumes indexing when more space becomes available.

via Splunk Web

- Log into Splunk Web as the admin user.
- Click **Manager** in the top-right of the interface.
- Click the **System Configuration** tab.
- Click **System Settings**.
- Change the value for **Pause indexing if free disk space falls below:** and click **Save**.

via Splunk CLI

To change the minimum free space value via the CLI, type the following:

```
# splunk set minfreemb 2000
```

This command sets the minimum free space to 2000 MB.

Bind Splunk to an IP

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, `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
```

Meet Splunk Web and Splunk apps

What's Splunk Web?

What's Splunk Web?

Splunk Web is Splunk's dynamic and interactive browser-based interface. Splunk Web is the primary interface for investigating problems, reporting on results, and managing Splunk deployments. Refer to the system requirements for a list of supported operating systems and browsers.

To launch Splunk Web, navigate to:

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

Use whatever host and port you chose during installation. The default port is 8000, but if that port was already in use, the installer asked you to pick a different one.

The first time you log into Splunk with an Enterprise license, use username "admin" and password "changeme". Splunk with a free license does not support access controls or multiple user accounts.

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 have changed the default password.

The Launcher

The first time you launch Splunk Web, you'll see the **Launcher**. This interface lets you choose an **app** from the list of apps currently available to you. In particular, you might want to check out the "Getting Started" app, but depending on what OS you're running, you'll also see an app that's specfic to Windows or UNIX. You can also visit the Splunk App Store to browse and download more apps.

Read on for more information about apps.

What are apps and add-ons?

What are apps and add-ons?

Apps give you insight into your IT systems with dashboards, reports, data inputs, and saved searches that work in your environment from the moment they install. Apps can include new views and dashboards that completely reconfigure the way Splunk looks. Or, they can be as complex as an entirely new program using Splunk's REST API.

Add-ons let you tackle specific data problems directly. They are smaller, reusable components that can change the look and feel of Splunk, add data sources, or share information between users. Add-ons can be as simple as a collection of one or more event type definitions or saved searches.

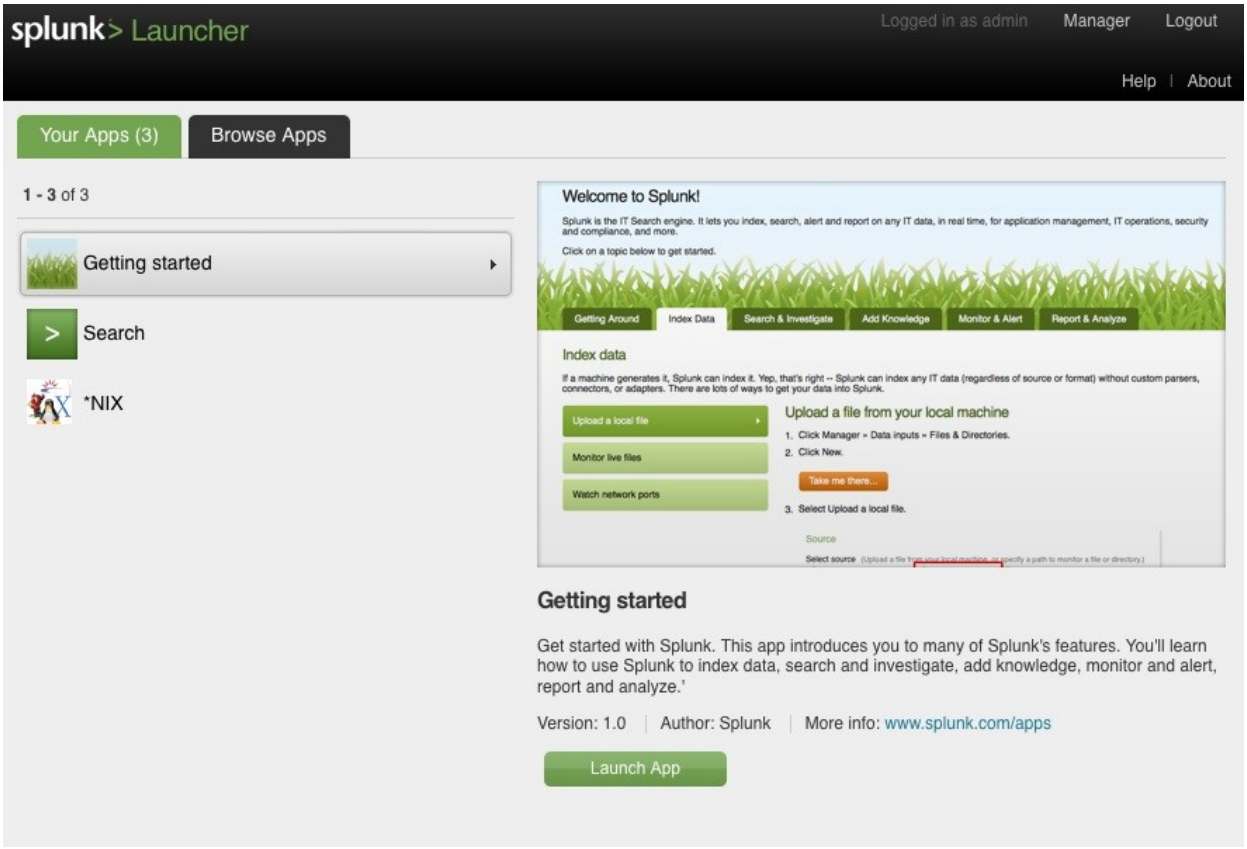
When you're using Splunk, you're almost always using an app; we typically refer to that as being "in" an app. The default app is the Search app.

What are apps and add-ons good for?

Apps and add-ons allow you to build different environments that sit on top of a single Splunk instance. You can create separate interfaces for the different communities of Splunk users within your organization: one app for troubleshooting email servers, another for Web analysis, an add-on that connects a lookup table for the frontline support team to use, and so on. This way, everyone can use the same Splunk instance, but see only data and tools that are relevant to their interests.

What apps and add-ons are there?

The first time you install and log into Splunk, you'll see the app **Launcher**. This interface shows you the list of apps that have been preinstalled for you. By default, one of these apps is the Getting Started app. This app has been developed to introduce new users to Splunk's features. If you're new to Splunk, we recommend you check it out and give us your feedback!



Bypass the Launcher for a single user

If you do not want the Launcher displayed every time you log into Splunk, you can configure a default app to land in. This can be done on a per-user basis. For example, 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 the Launcher 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.

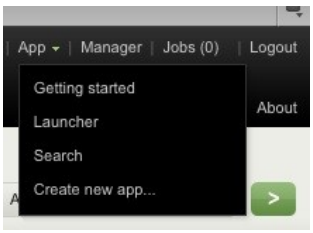
What you get by default

Besides the Getting Started app, Splunk comes with the Search app and another app to support your OS:

- The Search app interface provides the core functionality of Splunk and is designed for general-purpose use. If you've used Splunk before, the Search app replaces the main Splunk Web functionality from earlier versions. In the Search app you see a search bar and a dashboard full of graphs. When you are in the Search app, you change the dashboard or view by selecting new ones from the **Dashboards** and **Views** drop-down menus in the upper left of the window.

- The OS-specific app (Splunk for Windows or Splunk for *NIX) provides dashboards and pre-built searches to help you get the most out of Splunk on your particular platform. They are disabled by default, but you can turn them on from the apps section of Splunk Manager.

If you want to change the app you're in, select a new one from the App drop-down menu at the top right:



You can return to the Launcher later and select another app from there.

Get more apps

You can download a variety of other apps. For example, if the bulk of your data operations work involves tasks related to things like change management or PCI (Payment Card Industry)

compliance, you'll be happy to know that Splunk has apps designed specifically for those application areas.

To find more apps, click the **Browse More Apps** tab in the Launcher. For more information, see "Where to get more apps and add-ons".

How saving and sharing Splunk knowledge relates to apps

Splunk knowledge includes objects like saved searches, event types, tags -- items that enrich your Splunk data and make it easier to find what you need. In Splunk, these are known as **knowledge objects**.

Any user logged into Splunk Web can create and save knowledge objects to the user's directory under the app the user's "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.

Once the user has saved the object in a particular app, it is available to the user only in that app, unless they do one of the following things (and have the correct permissions to do so):

- Share the object with other specific roles or users in that same app
- Promote the object so that it is available to all users who have access to that app
- Promote the object so that it is available globally to all apps (and users)

Read more about App architecture and object ownership in this manual.

Where to get more apps and add-ons

Where to get more apps and add-ons

You can find new apps and add-ons on **http://www.splunkbase.com**.

All the apps and add-ons that are available on Splunkbase also show up in Launcher, so you can download and install them directly within Splunk.

When you log into Splunk Web, you see the Launcher by default. You can always get back to the Launcher from the App menu in the upper right-hand corner of the main page of any Splunk-provided app.

If you are connected to the internet

If your Splunk server or your client machine are connected to the internet, you can download apps directly from the Launcher:

1. From the Launcher, click on the **Browse Apps** tab. This will connect you to Splunkbase, where you can download apps and add-ons available for this version of Splunk.

2. Pick the app or add-on you want and select **Install**.

3. You will be prompted to log in with your splunk.com username and password (note that this is not your Splunk username and password).

4. Your app or add-on will be installed. If it has a Web GUI component (most add-ons contain only knowledge objects like event type definitions and don't have any GUI context), you can navigate to it from the Launcher.

If you are not connected to the internet

If your Splunk 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. Copy this app over to your Splunk server.

4. Put the app in your `$SPLUNK_HOME/etc/apps` directory.

5. Untar and ungzip your app or add-on, using a tool like `tar -xvf`. Note that Splunk apps 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, depending on the contents of the app or add-on.

7. Your app or add-on is now installed and will be available from the Launcher (if it has a Web GUI component).

App architecture and object ownership

App architecture and object ownership

Apps and add-ons are commonly built from Splunk **knowledge objects**. Splunk knowledge objects includes saved searches, event types, tags -- data types that enrich your Splunk deployment and make it easier to find what you need.

Any user logged into Splunk Web can create and save knowledge objects to the user's directory under the app the user's "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 to that app
- Restrict the object to specific roles or users (still within the app's context)
- Mark the object as globally available to all apps and users (unless you've explicitly restricted it by role/user)

Note: Users must have write permissions for an app before they can promote objects to the app 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 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 from within Manager, 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 set in the **App Configuration** tab of Splunk Manager. 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 are also managed through Splunk Manager. 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 Splunk Manager directly from the Launcher, your input will be added to `$SPLUNK_HOME/etc/apps/launcher/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

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, 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 Developer Manual.

View and manage app/add-on objects in Manager

To see and control the objects for all the apps on your your system, use Splunk Manager in Splunk Web. You can use Manager 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: **Manager > All configurations**.
- To see all the saved searches and report objects: **Manager > Saved searches and reports**.
- To see all the event types: **Manager > Event types**.
- To see all the field extractions: **Manager > Field extractions**.
- To see all the Python search command scripts: **Manager > Search commands**.

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**.

Disable an app 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.

How to configure Splunk

Splunk configuration methods

Splunk configuration methods

Splunk maintains its configuration information in a set of configuration files. You can configure Splunk by using any of these methods:

- Editing configuration files directly.
- Filling out fields in Splunk Manager in Splunk Web.
- Specifying Splunk CLI commands.

All of these methods change the contents of the underlying configuration files.

To configure and manage distributed environments, you can use Splunk's deployment server.

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`. You can make changes to these files using a standard text editor. Before you begin editing configuration files, read the material in the topic called About configuration files.

Splunk Manager

You can perform most common configuration tasks with Splunk Manager 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.

To access Splunk Manager, log into Splunk Web and click **Manager** in the upper right hand corner.

Splunk CLI

Many configuration options are available via the CLI. These options are documented in their respective topics, or you can get a complete CLI help reference with the command `help` while Splunk is running:

```
./splunk help
```

For more information about the CLI, refer to "About the CLI" in this manual.

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 "Deploy to other Splunk instances" in this manual.

Restarting after configuration changes

Many changes to configuration files require you to restart Splunk. Check the configuration file or its reference topic to see whether a particular change requires a restart.

When you make changes in the Manager, the Manager will let you know if you have to restart.

These changes require additional or different actions before they will take effect:

- Enable configuration changes made to `transforms.conf` by typing the following search in Splunk Web:

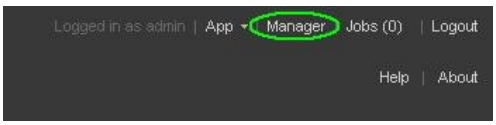
```
| extract reload=T
```

- Reload `authentication.conf` via the **Manager > Authentication** section of Splunk Web. This refreshes the authentication caches, but does not disconnect current users.

About Splunk Manager

About Splunk Manager

To configure Splunk from within Splunk Web, use Splunk Manager. To access Splunk Manager, log into Splunk Web and click **Manager** in the upper right:



Users with admin privileges can access all the areas of the Manager. Other users have limited access to the Manager.

System configurations

From the System configurations area, you can manage:

- System settings: Manage system settings including ports, host name, index path, email server settings (for alerts), and system logging.
- Server controls: Restart Splunk.
- License: View license usage statistics and apply a new license.
- Data inputs: Add data to Splunk from scripts, files, directories, and network ports.
- Forwarding and receiving: Configure this Splunk instance to send or receive data.

- Indexes: Create new indexes and manage index size preferences.
- Access controls: Specify authentication method (Splunk or LDAP), create or modify users, and manage roles.
- Distributed search: Set up distributed search across multiple Splunk instances.
- Deployment: Deploy and manage configuration settings across multiple Splunk instances.
- User options: Manage user settings, including passwords and email addresses.

Apps and knowledge

From the Apps and knowledge area, you can manage:

- Apps: Edit permissions for installed apps, create new apps, or browse Splunkbase for apps created by the community.
- Searches and reports: View, edit, and set permissions on searches and reports. Set up alerts and summary indexing.
- Event types: View, edit, and set permissions on event types.
- Tags: Manage tags on field values.
- Fields: View, edit, and set permissions on field extractions. Define event workflow actions and field aliases. Rename sourcetypes.
- Lookups: Configure lookup tables and lookups.
- User interface: Create and edit views, dashboards, and navigation menus.
- Advanced search: Create and edit search macros. Set permissions on search commands.
- All configurations: See all configurations across all apps.

About configuration files

About configuration files

Splunk's configuration information is stored in configuration files, identified by their `.conf` extension. These files are located under `$SPLUNK_HOME/etc`.

When you make a change to a configuration setting in **Splunk Manager** in Splunk Web, the change gets written to the relevant configuration file. This change is written to a copy of the configuration file in a directory under `$SPLUNK_HOME/etc` (the actual directory depends on a number of factors, discussed later), and the default value of the attribute is left alone in `$SPLUNK_HOME/etc/system/default`.

You can do a lot of configuration from the Manager, but for some more advanced customizations, you must edit the configuration files directly.

The configuration directory structure

The following is the configuration directory structure that exists under `$SPLUNK_HOME/etc`:

- `$SPLUNK_HOME/etc/system/default`
    - This contains the pre-configured configuration files. **Do not** modify the files in this directory.
- `$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.

- $SPLUNK_HOME/etc/apps/<app_name>/local
    - 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 such that the change only applies to a certain app, copy the file to the app's /local directory 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 files contain examples of real-world usage.

A single Splunk instance typically has multiple versions of some configuration files, across several of these directories. For example, you can have configuration files with the same names in your default, local, and app directories. This provides a layering effect that allows Splunk to determine configuration priorities based on factors such as the current user and the current app. Be sure to review the topic "Configuration file precedence" to understand the precedence rules governing Splunk configuration files. That topic explains how Splunk determines which files have priority.

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.

The default directory

When you edit a configuration file, you should not edit the version in $SPLUNK_HOME/etc/system/default. Instead, make a copy of the file and put it in another configuration directory. Since Splunk always looks at the default directory last, the edited version can go into any of the other available directories, according to whether the edit applies at the system, app, or user level. You can layer several versions of a configuration file on top of one-another, with different attribute values filtering through and being used by Splunk as described in "Configuration file precedence", but for most deployments, you can just use the $SPLUNK_HOME/etc/system/local directory to make configuration changes.

Another reason not to edit the copies of the configuration files in $SPLUNK_HOME/etc/system/default is that when you upgrade Splunk, all your changes will be overwritten. Changes you make to files in other directories are not overwritten and will continue to take effect post-upgrade.

Important: Some configuration files are not created by default -- if you want to enable the features they manage, you must create the configuration files from scratch. These configuration files still have .spec and .example files for you to review.

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

Splunk expects configuration files to be in ASCII/UTF-8. If you are editing or creating a configuration file on an operating system that is non-UTF-8, you must ensure that the editor you are using is configured to save in ASCII/UTF-8.

The structure of configuration files

Configuration files consist of one or more **stanzas**, or sections. Each stanza begins with a stanza header, designated by square brackets. Following the header is a series of attribute/value pairs that specify configuration settings. Depending on the stanza type, some of the attributes might be required, while others could be optional.

Here's the basic pattern:

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

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

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

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

[tcpout:my_indexers]
autoLB=true
compressed=true
server=mysplunk_indexer1:9997, mysplunk_indexer2:9996

[tcpout-server://mysplunk_indexer1:9997]
compressed=false
```

This example file has three levels of stanzas:

- The global `[tcpout]`, with settings that affect all tcp forwarding.
- The more specific `[tcpout:my_indexers]`, whose settings affect only the target group of indexers named "my_indexers" (whose members are defined within the stanza).
- The most specific `[tcpout-server://mysplunk_indexer1:9997]`, whose settings affect only one specific indexer in the target group.

The setting for `compressed` in `[tcpout-server://mysplunk_indexer1:9997]` overrides that attribute's setting in `[tcpout:my_indexers]`, *for the indexer "mysplunk_indexer1" only.*

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

List of configuration files, and what's in them

The following is an up-to-date list 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 `$SPLUNK_HOME/etc/system/default/`. Make a copy of the file in `$SPLUNK_HOME/etc/system/local/` or `$SPLUNK_HOME/etc/apps/<app_name>/local` and edit that copy.

File	Purpose
admon.conf	Configure Windows active directory monitoring.
alert_actions.conf	Customize Splunk's global alerting actions.
app.conf	Configure your custom app.
audit.conf	Configure auditing and event hashing.
authentication.conf	Toggle between Splunk's built-in authentication or LDAP, and configure LDAP.
authorize.conf	Configure roles, including granular access controls.
commands.conf	Connect search commands to any custom search script.
crawl.conf	Configure crawl to find new data sources.
default.meta.conf	A template file for use in creating app-specific default.meta files.
deploymentclient.conf	Specify behavior for clients of the deployment server.
distsearch.conf	Specify behavior for distributed search.
eventdiscoverer.conf	Set terms to ignore for typelearner (event discovery).
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.
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 language macros.
multikv.conf	Configure extraction rules for table-like events (ps, netstat, ls).
outputs.conf	Set up forwarding, routing, cloning and data balancing.
pdf_server.conf	Configure the Splunk pdf server.
procmon-filters.conf	Monitor Windows process data.
props.conf	Set indexing property configurations, including timezone offset, custom sourcetype rules, and pattern collision priorities. Also, map transforms to event properties.
pubsub.conf	Define a custom client of the deployment server.

regmon-filters.conf	Create filters for Windows registry monitoring.
report_server.conf	Configure the report server.
restmap.conf	Configure REST endpoints.
savedsearches.conf	Define saved searches and their associated schedules and alerts.
searchbnf.conf	Configure the search assistant.
segmenters.conf	Customize segmentation rules for indexed events.
server.conf	Enable SSL for Splunk's back-end 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 sourcetype.
sourcetypes.conf	Machine-generated file that stores sourcetype learning rules created by sourcetype training.
sysmon.conf	Set up Windows registry monitoring.
tags.conf	Configure tags for fields.
tenants.conf	Configure deployments in multi-tenant environments.
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.
user-seed.conf	Set a default user and password.
web.conf	Configure Splunk Web, enable HTTPS.
wmi.conf	Set up Windows management instrumentation (WMI) inputs.
workflow_actions.conf	Configure workflow actions.

Configuration file precedence

Configuration file precedence

This topic describes how Splunk sifts through its layers of **configuration files** to determine which settings to use. For general information about configuration files, read "About configuration files".

Order of precedence

Splunk uses configuration files to determine nearly every aspect of its behavior. Besides having many different types of configuration files, a single Splunk instance can also have many copies of the same configuration file, layered in directories categorized by user, **app,** and system. When consuming a configuration file, Splunk 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), Splunk uses the value from the file with the highest priority.

Splunk determines the priority of configuration files by their location in its directory structure. At the most general level, it prioritizes according to whether the file is located in a system, app, or user directory.

You need to understand how this works so that you can manage your Splunk configurations intelligently. It's all pretty straightforward when you focus on context. Once you get a feel for the context in which Splunk is consuming a particular file, the way precedence works makes quite a bit of

sense.

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

The app/user context

In determining priority among copies of a configuration file, Splunk uses two different schemes of directory precedence, according to whether that particular configuration relates to an activity with an app/user context -- that is, where the current app and user matter. Some activities, like searching, take place in an app/user context; others, like indexing, take place in a global context, independent of any app or user.

For instance, configuration files that determine indexing or monitoring behavior occur outside of the app/user context; they are global in nature. At the time of data input or event indexing, it does not matter which user or app might later access the data or event. The app/user context, on the other hand, is vital to search-time processing, where certain knowledge objects or actions might be valid only for specific users in specific apps.

How context affects precedence order

When the context is global, where there's no app/user context, directory priority descends from system/local to app to system/default:

- System local directory -- highest priority.
- App directories (local overrides default).
- System default directory -- lowest priority.

When consuming a global configuration, such as `inputs.conf`, Splunk first gets 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.

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

- User directories -- highest priority.
- App directories for currently running app (local, followed by default).
- App directories for all other apps (local, followed by default) -- for exported settings only.
- 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. 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:

In the 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/*
```

In the 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

$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 only looks at configuration data for other apps if that data has been exported globally through the app's `default.meta` file, as described in this topic on setting app permissions.

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, check out "Troubleshooting configurations" in this manual.

Attribute precedence within a single props.conf file

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
```

Indexing with Splunk

What's a Splunk index?

What's a Splunk index?

The **index** is the repository for Splunk data. While processing incoming data, Splunk transforms the raw data into **events**, which it stores in indexes.

Indexes reside in flat files in a datastore on your file system. Splunk manages its index files to facilitate flexible searching and fast data retrieval, eventually archiving them according to a user-configurable schedule. Splunk handles everything with flat files; it doesn't require any third-party database software running in the background.

During indexing, Splunk processes incoming raw data to enable fast search and analysis, storing the result in an index. As part of the indexing process, Splunk adds knowledge to the data in various ways, including by:

- Separating the datastream into individual, searchable events.
- Creating or identifying timestamps.
- Extracting fields such as host, source, and sourcetype.
- Performing user-defined actions on the incoming data, such as identifying custom fields, masking sensitive data, writing new or modified keys, applying breaking rules for multi-line events, filtering unwanted events, and routing events to specified indexes or servers.

To start the indexing process, simply specify the data inputs, using Splunk Web, the CLI, or the `inputs.conf` file. You can add additional inputs at any time, and Splunk will begin indexing them as well. See Add data and configure inputs in this manual.

Splunk, by default, puts all user data into a single, preconfigured index. It also employs several other indexes for internal purposes. You can add new indexes and manage existing ones to meet your data requirements. See Manage indexes in this manual.

How indexing works

How indexing works

Splunk can **index** any type of time-series data (data with **timestamps**). When Splunk indexes data, it breaks it into **events**, based on its timestamps.

Event processing

Event processing occurs in two stages, parsing and indexing. All data that comes into Splunk enters through the **parsing pipeline** as large (10,000 bytes) chunks. During parsing, Splunk breaks these chunks into events which it hands off to the **indexing pipeline**, where final processing occurs.

While parsing, Splunk performs a number of actions, including:

- Extracting sets of default fields for each event, including `host`, `source`, and `sourcetype`.
- Configuring character set encoding.
- Identifying line termination using linebreaking rules. While many events are short and only take up a line or two, others can be long.
- Identifying timestamps or creating them if they don't exist. At the same time that it processes timestamps, Splunk identifies event boundaries.
- Splunk can be set up to mask sensitive event data (such as credit card or social security numbers) during the indexing process. It can also be configured to apply custom metadata to incoming events.

In the indexing pipeline, Splunk performs additional processing, including:

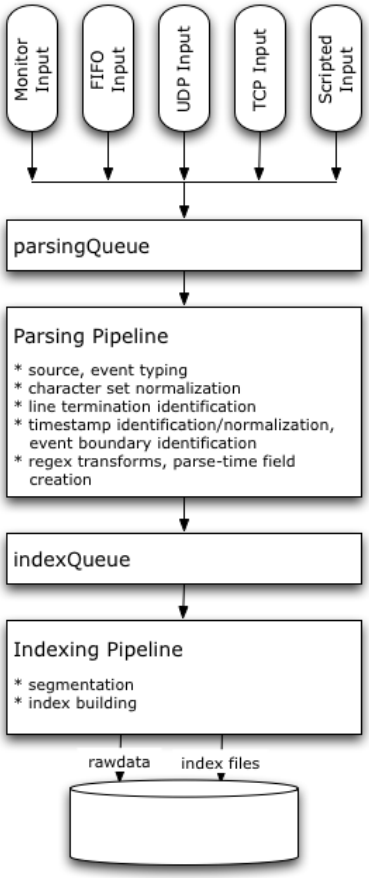
- Breaking all events into segments that can then be searched upon. You can determine the level of segmentation, which affects indexing and searching speed, search capability, and efficiency of disk compression.
- Building the index data structures.
- Writing the raw data and index files to disk, where post-indexing compression occurs.

The breakdown between parsing and indexing pipelines is mainly of relevance for forwarders, which can parse, but not index, data.

For more information about events and what happens to them during the indexing process, see Overview of event processing in this manual.

Note: Indexing is an I/O-intensive process.

This diagram shows the main processes inherent in indexing:



What's in an index?

Splunk stores all of the data it processes in indexes. An index is a collection of databases, which are directories located in `$SPLUNK_HOME/var/lib/splunk`. A database directory is named `db_<starttime>_<endtime>_<seq_num>`.

Splunk comes with the following preconfigured indexes:

- **main**: This is the default Splunk index. All processed data is stored here unless otherwise specified.
- **_internal**: Stores Splunk internal logs and processing metrics.
- **sampledata**: A small amount of sample data is stored here for training purposes.
- **_audit**: Contains events related to the file system change monitor, auditing, and all user search history.

A Splunk administrator can create new indexes, edit index properties, remove unwanted indexes, and relocate existing indexes. Splunk administrators manage indexes through Splunk Manager, the CLI, and configuration files such as `indexes.conf`. For more information, See Managing indexes in this manual.

Answers

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

Index time versus search time

Index time versus search time

Splunk documentation includes many references to the terms **"index time"** and "search time." These terms distinguish between the sorts of event data that are processed by Splunk during indexing, and other kinds of event data that are processed when a search is run.

It is important to consider this distinction when administering Splunk. For example, if you haven't yet started indexing data and you think you're going to have a lot of custom source types and hosts, you might want to get those in place before you start indexing. You can do this by defining custom source types and hosts (through rule-based source type assignation, source type overriding, input-based host assignment, and host overrides), so that these things are handled during the indexing process.

On the other hand, if you have already begun to index your data, you might want to handle the issue at search time. Otherwise, you will need to re-index your data, in order to apply the custom source types and hosts to your existing data as well as new data. After indexing, you can't change the host or source type assignments, but you can tag them with alternate values and manage the issue that way.

As a general rule, it is better to perform most knowledge-building activities, such as field extraction, at search time. Additional, custom field extraction, performed at index time, can degrade performance at both index time and search time. When you add to the number of fields extracted during indexing, the indexing process slows. Later, searches on the index are also slower, because the index has been enlarged by the additional fields, and a search on a larger index takes longer. You can avoid such performance issues by instead relying on search-time field extraction. For details on search-time field extraction, see "About fields" and "Create search-time field extractions" in the Knowledge Manager manual.

At index time

Index-time processes take place just before event data is actually indexed.

The following processes can occur during (or before) index time:

- Default field extraction (such as `host`, `source`, `sourcetype`, and `timestamp`)
- Static or dynamic host assignment for specific inputs
- Default host assignment overrides
- Source type customization
- Index-time field extraction
- Event timestamping
- Event linebreaking
- Event segmentation (also happens at search time)

At search time

Search-time processes take place while a search is run, as events are collected by the search. The following processes occur at search time:

- Event segmentation (also happens at index time)

- Event type matching
- Search-time field extraction (includes automatic and custom field extractions, as well as multivalue field parsing)
- Field aliasing
- Field lookups from external data sources
- Source type renaming
- Tagging

Advanced indexing strategy

Advanced indexing strategy

In a single-machine deployment consisting of just one Splunk instance, the Splunk **indexer** also handles data input and search requests. However, for mid-to-enterprise scale needs, indexing is typically split out from the data input function and sometimes from the search function as well. In these larger, distributed deployments, the Splunk indexer might reside on its own machine and handle only indexing.

For instance, you can have a set of Windows and Linux machines generating interesting events, which need to go to a central Splunk indexer for consolidation. Usually the best way to do this is to install a lightweight instance of Splunk, known as a **forwarder**, on each of the event-generating machines. These forwarders handle data input and send the data across the network to the Splunk indexer residing on its own machine.

Similarly, in cases where you have a large amount of indexed data and numerous concurrent users searching on it, it can make sense to split off the search function from indexing. In this type of scenario, known as **distributed search**, one or more **search heads** distribute search requests across multiple indexers.

To manage a distributed deployment, Splunk's **deployment server** lets you push out configurations and content to sets of Splunk instances, grouped according to any arbitrary criteria, such as OS, machine type, application area, location, and so on.

While the fundamental issues of indexing and event processing remain the same for distributed deployments, it is important to take into account deployment needs when planning your indexing strategy.

Forward data to an indexer

This type of deployment involves the use of forwarders, which are Splunk instances that receive data inputs and then consolidate and send the data to a Splunk indexer. Forwarders come in two flavors:

- Regular forwarders. These retain most of the functionality of a full Splunk instance. They can parse data before forwarding it to the receiving indexer, also known as the **receiver**. (See How indexing works for the distinction between parsing and indexing.) They can also retain indexed data locally, while forwarding the parsed data to the receiver for final indexing on that machine as well.
- **Light forwarders**. These forwarders maintain a small footprint on their host machine. They perform minimal processing on the incoming data streams before forwarding them on to the

receiving indexer.

Both types of forwarders tag data with metadata such as host, source, and source type, before forwarding it on to the indexer.

Forwarders allow you to use resources efficiently while processing large quantities or disparate types of data. They also feature in a number of interesting use case scenarios, to handle key needs such as **load balancing**, **data cloning** for enhanced availability, and **data filtering** and **routing**.

For an extended discussion of forwarders, including configuration and detailed use cases, see About forwarding and receiving.

Search across multiple indexers

In distributed search, Splunk servers send search requests to other Splunk servers and merge the results back to the user. This is useful for a number of purposes, including horizontal scaling, access control, and managing geo-dispersed data.

The Splunk instance that manages search requests is called the search head. The instances that maintain the indexes and perform the actual searching are called search peers or indexer nodes.

For an extended discussion of distributed search, including configuration and detailed use cases, see What is distributed search?.

Manage distributed deployments

When dealing with distributed deployments consisting potentially of multiple forwarders, indexers, and search heads, the Splunk deployment server greatly eases the process of configuring and updating all Splunk instances. With the deployment server, you can group the distributed Splunk instances (referred to as **deployment clients** in this context) into **server classes**.

A server class is a set of Splunk instances that share configurations. Server classes are typically grouped by OS, machine type, application area, location, or other useful criteria. A single deployment client can belong to multiple server classes, so a Linux forwarder residing in the UK, for example, might belong to a Linux server class and a UK server class, and receive configuration settings appropriate to each.

For an extended discussion of deployment management, see About deployment server.

Add data and configure inputs

How to get your data into Splunk

How to get your data into Splunk

No special plugins are required for Splunk to index data any network or local source. Splunk can index any IT data from any source in real time. We call this **universal indexing**.

Seriously, from any data source

A variety of flexible input methods let you index logs, configurations, traps and alerts, messages, scripts, and code and performance data from all your applications, servers and network devices. Monitor file systems for scripts and configuration changes, capture archive files, find and tail live application logs, connect to network ports to receive syslog, SNMP and other network-based instrumentation.

Splunk consumes any data you point it at. Before indexing data, you must add your data source as an **input**. The **source** is then listed as one of Splunk's **default fields** (whether it's a file, directory or network port).

Important: If you add an input, Splunk adds that input to a copy of `inputs.conf` that belongs to the **app** you're currently in. This means that if you navigated to **Splunk Manager** directly from the Launcher and then added an input there, your input will be added to `$SPLUNK_HOME/etc/apps/launcher/local/inputs.conf`. **Make sure you're in the desired app when you add your inputs.**

Ways to get data into Splunk

Specify data inputs via the following methods:

- The Manager in **Splunk Web**.
- Splunk's **CLI**.
- The inputs.conf configuration file.
- Forwarding from other systems

You can add most data sources using Splunk Web. For more extensive configuration options, use `inputs.conf`. Changes you make using Splunk Web or the Splunk CLI are written to `$SPLUNK_HOME/etc/system/local/inputs.conf`.

Sources

Splunk accepts data inputs in a variety of ways. Here's a basic overview of your options.

Files and directories

A lot of the data you may be interested comes directly from files and directories. For the most part, you can use Splunk's **files and directories monitor** input processor to index data in files and directories.

You can also configure Splunk's **file system change monitor** to watch for changes in your file system. However, you shouldn't use both the file and directories monitor and the file system change monitor to follow the same directory or file. If you want to see changes in a directory, use the file system change monitor. If you want to index new events in a directory, use the file and directories monitor.

To monitor files and directories, see "Monitor files and directories".

To enable and configure the file system change monitor, see "Monitor changes to your file system".

TCP network ports

TCP is a reliable, connection-oriented protocol that should be used instead of UDP to transmit and receive data whenever possible. Splunk with an Enterprise license can receive data on any TCP port, allowing Splunk to receive remote data from syslog-ng and any other application that transmits via TCP.

To monitor data via TCP, see "Monitor network ports".

Splunk supports monitoring over UDP, but recommends using TCP instead whenever possible. UDP is generally undesirable as a transport because:

- It doesn't enforce delivery.
- It's not encrypted.
- There's no accounting for lost datagrams.

Refer to "Working with UDP connections" on the Splunk Community Wiki for recommendations if you must use UDP.

Windows sources

Splunk on Windows ships with the Windows inputs, well as pages in Splunk Manager for defining the Windows-specific input types listed below. Because of compatibility issues, you will not see Windows-specific inputs or Splunk Manager pages on non-Windows Splunk instances.

Splunk on Windows can add data from these Window-specific sources:

- Windows Event Log data
- Windows Registry data
- WMI data
- Active Directory data

Important: You can index and search your Windows data on a non-Windows instance of Splunk, but you must first use a Windows instance of Splunk to gather the Windows data. You can easily do this by means of a Splunk forwarder running on Windows, configured to gather Windows inputs and then forward the data to the non-Windows instance of Splunk where searching and indexing will take place. See "Considerations for deciding how to monitor remote Windows data" for the details.

Need some help deciding how to get your Windows data into Splunk? Check out "Considerations for deciding how to monitor remote Windows data" in this manual.

Got custom data? It might need some extra TLC

Splunk can index any time-series data, usually without the need for additional configuration. If you've got logs from a custom application or device, you should try Splunk's defaults first. But if you're not getting the results you want, you can tweak a bunch of different things to make sure your events are indexed correctly.

We recommend you learn a bit about event processing and how Splunk indexes data before proceeding so you can make informed decisions about what TLC your data needs. Some options include:

- Are your events multi-line?
- Is your data in an unusual character set?
- Is Splunk not figuring out the timestamps correctly?

Not finding the events you're looking for?

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 that ship with Splunk, write input data to a specific index (in the case of *nix and Windows, that is the 'os' index). If you're not finding data that you're certain is in Splunk, be sure that you're looking at the right index. You may want to add the 'os' index to the list of default indexes for the role you're using. For more information about roles, refer to the topic about roles in this manual.

Note: Splunk looks for the inputs it is configured to monitor every 24 hours starting from the time it was last restarted. This means that if you add a stanza to monitor a directory or file that doesn't exist yet, it could take up to 24 hours for Splunk to start indexing its contents. To ensure that your input is immediately recognized and indexed, add the input using Splunk Web or by using the `add` command in the CLI.

Specify input paths with wildcards

Specify input paths with wildcards

Important: Input path specifications in inputs.conf don't use REGEX-compliant expressions but rather Splunk-defined wildcards.

A wildcard is a character that you can substitute for one or more unspecified characters when searching text or selecting multiple files or directories. In Splunk, you can use wildcards to specify your input path for monitored input; use `...` for paths and `*` for files.

Wildcard	Description	Regex equivalent	Example(s)
`...`	The ellipsis wildcard recurses through directories and subdirectories to match.	`.*`	`/foo/.../bar` matches the files `/foo/bar`, `/foo/1/bar`, `/foo/1/2/bar`, etc. **Note:** This only works if `bar` is a file.

*	The asterisk wildcard matches anything in that specific directory path segment. **Note:** It cannot be used inside a directory path; must be used in the last segment of the path.	`[^/]*`	`/foo/*.log` matches all files with the `.log` extension, such as `/foo/bar.log`. It does not match `/foo/bar.txt` or `/foo/bar/test.log`.

Note: A single dot (`.`) is not a wildcard, and is the regex equivalent of `\.`.

For more specific matches, combine the `...` and `*` wildcards. For example, `/foo/.../bar/*` matches any file in the `/bar` directory within the specified path.

Input examples

To load anything in `/apache/foo/logs` or `/apache/bar/logs`, etc.

```
[monitor:///apache/.../logs]
```

To load anything in `/apache/` that ends in `.log`.

```
[monitor:///apache/*.log]
```

Wildcards and whitelisting

Important: In Splunk, whitelists and blacklists are defined with standard PCRE REGEX syntax (unlike the file input path syntax described in the previous sections).

Specifying wildcards results in an implicit `whitelist` created for that stanza. The longest fully qualified path is used as the monitor stanza, and the wildcards are translated into regular expressions, as described in the table above.

Note: In Windows, `whitelist` and `blacklist` rules do not support regexes that include blackslashes; you must use two backslashes `\\` to escape wildcards.

Additionally, the converted expression is anchored to the right end of the file path, so that the entire path must be matched.

For example, if you specify

```
[monitor:///foo/bar*.log]
```

Splunk translates this into

```
[monitor:///foo/]
whitelist = bar[^/]*\.log$
```

As a consequence, you can't have multiple stanzas with wildcards for files in the same directory. If you have multiple inputs that only disambiguate after a wildcard, they will collide.

Also, you cannot use a `whitelist` declaration in conjunction with wildcards.

For example:

```
[monitor:///foo/bar_baz*]
[monitor:///foo/bar_qux*]
```

This results in overlapping stanzas indexing the directory `/foo/`. Splunk takes the first one, so only files starting with `/foo/bar_baz` will be indexed. To include both sources, manually specify a `whitelist` using regular expression syntax for "or":

```
[monitor:///foo]
whitelist = (bar_baz[^/]*|bar_qux[^/]*)$
```

Note: To set any additional attributes (such as sourcetype) for multiple whitelisted/blacklisted inputs that may have different attributes, use props.conf.

Monitor files and directories

Monitor files and directories

Splunk has two file input processors: **monitor** and **upload**. For the most part, you can use monitor to add all your data sources from files and directories. However, you may want to use upload when you want to add one-time inputs, such as an archive of historical data.

This topic discusses how to add monitor and upload inputs using Splunk Web and the configuration files. You can also add, edit, and list monitor inputs using the CLI; for more information, read this topic.

How monitor works in Splunk

Specify a path to a file or directory and Splunk's monitor processor consumes any new input. This is how you'd monitor live application **logs** such as those coming from J2EE or .Net applications, Web access logs, and so on. Splunk will continue to index the data in this file or directory as it comes in. You can also specify a mounted or shared directory, including network filesystems, as long as the Splunk server can read from the directory. If the specified directory contains subdirectories, Splunk recursively examines them for new files.

Splunk checks for the file or directory specified in a monitor configuration on Splunk server start and restart. If the file or directory specified is not present on start, Splunk checks for it again in 24 hour intervals from the time of the last restart. Subdirectories of monitored directories are scanned continuously. To add new inputs without restarting Splunk, use Splunk Web or the command line interface. If you want Splunk to find potential new inputs automatically, use **crawl**.

When using monitor, note the following:

- On most file systems, files can be read even as they are being written to. However, Windows file systems have the ability to prevent files from being read while they are being written, and some Windows programs may use these modes, though most do not.
- Files or directories can be included or excluded via **whitelists** and **blacklists**.

- Upon restart, Splunk continues processing files where it left off.
- Splunk decompresses archive files before it indexes them. It can handle these common archive file types: .tar, .gz, .bz2, .tar.bz2 , and .zip.
- Splunk detects log file rotation and does not process renamed files it has already indexed (with the exception of .tar and .gz archives; for more information see "Log file rotation" in this manual).
- The entire `dir/filename` path must not exceed 1024 characters.
- Set the source type for directories to Automatic. If the directory contains multiple files of different formats, do not set a value for the source type manually. Manually setting a source type forces a single source type for all files in that directory.
- Removing an input does not stop the the input's files from being indexed. Rather, it stops files from being checked again, but all the initial content will be indexed. To stop all in-process data, you must restart the Splunk server.

Note: You cannot currently use both **monitor** and **file system change monitor** to follow the same directory or file. If you want to see changes in a directory, use file system change monitor. If you want to index new events in a directory, use monitor.

Note: Monitor input stanzas may not overlap. That is, monitoring `/a/path` while also monitoring `/a/path/subdir` will produce unreliable results. Similarly, monitor input stanzas that watch the same directory with different whitelists, blacklists, and wildcard components are not supported.

Why use upload or batch

Use the **Upload a local file** or **Index a file on the Splunk server** options to index a static file one time. The file will not be monitored on an ongoing basis.

Use the batch input type in `inputs.conf` to load files once and destructively. By default, Splunk's batch processor is located in `$SPLUNK_HOME/var/spool/splunk`. If you move a file into this directory, Splunk indexes it and then deletes it.

Note: For best practices on loading file archives, see "How to index different sized archives" on the Community Wiki.

Configure with Splunk Web

Add inputs from files and directories via Splunk Web.

1. Click **Manager** in the upper right-hand corner of Splunk Web.

2. Under System configurations, click **Data Inputs**.

3. Click **Files and directories**.

4. Click **Add new** to add an input.

5. Select a **Source** radio button:

- **Monitor a file or directory**. Sets up an ongoing input. Whenever data is added to this file or directory, Splunk will index it. See the next section for advanced options specific to this choice.

- **Upload a local file**. Uploads a file from your local machine into Splunk.
- **Index a file on the Splunk server**. Copies a file on the server into Splunk via the batch directory.

6. Specify the **Full path** to the file or directory.

To monitor a shared network drive, enter the following: `<myhost><mypath>` (or `\\<myhost>\<mypath>` on Windows). Make sure Splunk has read access to the mounted drive, as well as to the files you wish to monitor.

7. Under the **Host** section, set the host name value. You have several choices for this setting. Learn more about setting the host value in "About default fields".

Note: Host only sets the **host** field. It does not direct Splunk to look on a specific host on your network.

8. Set the **Source type**. **Source type** is a default field added to events. Source type is used to determine processing characteristics such as timestamps and event boundaries.

9. Set the **Index**. Leave the value as "default", unless you have defined multiple indexes to handle different types of events. In addition to indexes for user data, Splunk has a number of utility indexes, which show up in the dropdown box.

10. Click **Save**.

Advanced options for file/directory monitoring

If your choice for source is **Monitor a file or directory**, the page includes an **Advanced Options** section, which allows you to configure some additional settings:

- **Follow tail.** If checked, monitoring begins at the end of the file (like `tail -f`).
- **Whitelist.** If a path is specified, files from that path are monitored only if they match the specified regex.
- **Blacklist.** If a path is specified, files from that path are not monitored if they match the specified regex.

For detailed information on whitelists and blacklists, see Whitelist or blacklist specific incoming data in this manual.

Configure with inputs.conf

To add an input, add a **stanza** to inputs.conf in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. If you have not worked with Splunk's configuration files before, read "About configuration files" before you begin.

You can set multiple attributes in an input stanza. If you do not specify a value for an attribute, Splunk uses the default that's preset in `$SPLUNK_HOME/etc/system/default/`.

Note: To ensure that new events are indexed when you copy over an existing file with new contents, set `CHECK_METHOD = modtime` in props.conf for the source. This checks the modtime of the file

and re-indexes it when it changes. Be aware that the entire file will be re-indexed, which can result in duplicate events.

Configuration settings

The following are options that you can use in both `monitor` and `batch` input stanzas. See the sections that follow for attributes that are specific to each type of input.

`host = <string>`

- Set the host value of your input to a static value.
- "host=" is automatically prepended to `<string>`.
- Defaults to the IP address or fully qualified domain name of the host where the data originated.

`index = <string>`

- Set the index where events from this input will be stored.
- "index=" is automatically prepended to `<string>`.
- Defaults to `main`, or whatever you have set as your default index.
- For more information about the index field, see "How indexing works" in this manual.

`sourcetype = <string>`

- Set the sourcetype name of events from this input.
- "sourcetype=" is automatically prepended to `<string>`.
- Splunk picks a sourcetype based on various aspects of your data. There is no hard-coded default.
- For more information about the sourcetype field, see "About default fields (host, source, sourcetype, and more)", in this manual.

`source = <string>`

- Set the source name of events from this input.
- Defaults to the file path.
- "source=" is automatically prepended to `<string>`.

`queue = parsingQueue | indexQueue`

- Specifies where the input processor should deposit the events that it reads.
- Set to "parsingQueue" to apply `props.conf` and other parsing rules to your data.
- Set to "indexQueue" to send your data directly into the index.
- Defaults to `parsingQueue`.

`_TCP_ROUTING = <tcpout_group_name>,<tcpout_group_name>,...`

- Specifies a comma-separated list of tcpout group names.
- Using this attribute, you can selectively forward your data to specific indexer(s) by specifying the tcpout group(s) that the forwarder should use when forwarding your data.
- The tcpout group names are defined in `outputs.conf` in

[tcpout:<tcpout_group_name>] stanzas.
- This setting defaults to the groups present in 'defaultGroup' in [tcpout] stanza in outputs.conf.

host_regex = <regular expression>

- If specified, the regex extracts host from the filename of each input.
- Specifically, the first group of the regex is used as the host.
- Defaults to the default "host =" attribute, if the regex fails to match.

host_segment = <integer>

- If specified, a segment of the path is set as host, using <integer> to determine which segment. For example, if host_segment = 2, host is set to the second segment of the path. Path segments are separated by the '/' character.
- Defaults to the default "host =" attribute, if the value is not an integer, or is less than 1.

Monitor syntax and examples

Monitor input stanzas direct Splunk to watch all files in the <path> (or just <path> itself if it represents a single file). You must specify the input type and then the path, so put three slashes in your path if you're starting at root. You can use wildcards for the path. For more information, read how to "Specify input paths with wildcards".

```
[monitor://<path>]
<attrbute1> = <val1>
<attrbute2> = <val2>
...
```

The following are additional attributes you can use when defining monitor input stanzas:

crcSalt = <string>

- If set, this string is added to the CRC.
- Use this setting to force Splunk to consume files that have matching CRCs.
- If set to crcSalt = <SOURCE> (note: This setting is case sensitive), then the full source path is added to the CRC.

followTail = 0|1

- If set to 1, monitoring begins at the end of the file (like tail -f).
- This only applies to files the first time they are picked up.
- After that, Splunk's internal file position records keep track of the file.

whitelist = <regular expression>

- If set, files from this path are monitored only if they match the specified regex.

blacklist = <regular expression>

- If set, files from this path are NOT monitored if they match the specified regex.

`alwaysOpenFile = 0 | 1`

- If set to 1, Splunk opens a file to check if it has already been indexed.
- Only useful for files that don't update modtime.
- Should only be used for monitoring files on Windows, and mostly for IIS logs.
- **Note:** This flag should only be used as a last resort, as it increases load and slows down indexing.

`time_before_close = <integer>`

- Modtime delta required before Splunk can close a file on EOF.
- Tells the system not to close files that have been updated in past `<integer>` seconds.
- Defaults to 3.

`recursive = true|false`

- If set to `false`, Splunk will not go into subdirectories found within a monitored directory.
- Defaults to `true`.

`followSymlink`

- If `false`, Splunk will ignore symbolic links found within a monitored directory.
- Defaults to `true`.

Example 1. To load anything in `/apache/foo/logs` or `/apache/bar/logs`, etc.

```
[monitor:///apache/.../logs]
```

Example 2. To load anything in `/apache/` that ends in `.log`.

```
[monitor:///apache/*.log]
```

Batch syntax and examples

Use batch to set up a one time, destructive input of data from a source. For continuous, non-destructive inputs, use **monitor**. Remember, after the batch input is indexed, Splunk **deletes** the file.

```
[batch://<path>]
move_policy = sinkhole
<attrbute1> = <val1>
<attrbute2> = <val2>
...
```

Important: When defining batch inputs, you must include the setting, `move_policy = sinkhole`. This loads the file destructively. Do not use this input type for files you do not want to consume destructively.

Note: `source = <string>` and `<KEY> = <string>` are not used by batch.

Example: This example batch loads all files from the directory `/system/flight815/`.

```
[batch://system/flight815/*]
move_policy = sinkhole
```

Monitor files and directories using the CLI

Monitor files and directories using the CLI

Monitor files and directories via Splunk's Command Line Interface (CLI). To use Splunk's CLI, navigate to the `$SPLUNK_HOME/bin/` directory and use the `./splunk` command from the UNIX or Windows command prompt.

If you get stuck, Splunk's CLI has built-in help. Access the main CLI help by typing `splunk help`. Individual commands have their own help pages as well -- type `splunk help <command>`.

CLI commands for input configuration

The following commands are available for input configuration via the CLI:

Command	Command syntax	Action
add	`add monitor $SOURCE [-parameter value] ...`	Add inputs from `$SOURCE`.
edit	`edit monitor $SOURCE [-parameter value] ...`	Edit a previously added input for `$SOURCE`.
remove	`remove monitor $SOURCE`	Remove a previously added `$SOURCE`.
list	`list monitor`	List the currently configured monitor.
spool	`spool source`	Copy a file into Splunk via the sinkhole directory.

Change the configuration of each data input type by setting additional parameters. Parameters are set via the syntax: `-parameter value`.

Note: You can only set one `-hostname`, `-hostregex` or `-hostsegmentnum` per command.

Parameter	Required?	Description
`source`	Required	Path to the file or directory to monitor for new input.
`sourcetype`	Optional	Specify a sourcetype field value for events from the input source.
`index`	Optional	Specify the destination index for events from the input source.
`hostname`	Optional	Specify a host name to set as the host field value for events from the input source.
`hostregex`	Optional	Specify a regular expression on the source file path to set as the host field value for events from the input source.
`hostsegmentnum`	Optional	Set the number of segments of the source file path to set as the host field value for events from the input source.

follow-only	Optional	(T/F) True or False. Default False. When set to True, Splunk will read from the end of the source (like the "tail -f" Unix command).

Example 1. monitor files in a directory

The following example shows how to monitor files in `/var/log/`:

Add `/var/log/` as a data input:

```
./splunk add monitor /var/log/
```

Example 2. monitor windowsupdate.log

The following example shows how to monitor the Windows Update log (where Windows logs automatic updates):

Add `C:\Windows\windowsupdate.log` as a data input:

```
.\splunk add monitor C:\Windows\windowsupdate.log
```

Example 3. monitor IIS logging

This example shows how to monitor the default location for Windows IIS logging: Add `C:\windows\system32\LogFiles\W3SVC` as a data input:

```
.\splunk add monitor c:\windows\system32\LogFiles\W3SVC
```

Monitor network ports

Monitor network ports

You can enable Splunk to accept an input on any TCP or UDP port. Splunk consumes any data sent on these ports. Use this method for syslog (default port is UDP 514) or set up netcat and bind to a port.

TCP is the protocol underlying Splunk's data distribution, which is the recommended method for sending data from any remote machine to your Splunk server. Note that the user you run Splunk as must have access to the port. On a Unix system you must run as root to access a port under 1024.

Add a network input using Splunk Web

Add inputs from network ports via Splunk Web.

1. Click **Manager** in the upper right-hand corner of Splunk Web.

2. Under **System configurations** click **Data inputs**.

3. Pick **TCP** or **UDP**.

4. Click **Add new** to add an input.

5. Enter a port number.

6. If this is a TCP input, you can specify whether this port should accept connections from all hosts or one host. If you specify one host, enter the IP address of the host.

7. Enter a new **Source name** to override the default source value, if necessary.

Important: Consult Splunk support before changing this value.

8. Set the **Host** by selecting a radio button:

- **IP.** Sets the input processor to rewrite the host with the IP address of the remote server.
- **DNS.** Sets the host to the DNS entry of the remote server.
- **Custom.** Sets the host to a user-defined label.

9. Now set the **Source type**.

Source type is a default field added to events. Source type is used to determine processing characteristics such as timestamps and event boundaries. Choose:

- **From List.** Select one of the predefined source types from the drop-down list.
- **Manual.** Label your own source type in the text box.

10. Set the **Index**. Leave the value as "default" unless you have defined multiple indexes to handle different types of events. In addition to indexes meant for user data, Splunk has a number of utility indexes, which show up in the dropdown box.

11. Click **Save**.

Add a network input using the CLI

Monitor files and directories via Splunk's Command Line Interface (CLI). To use Splunk's CLI, navigate to the `$SPLUNK_HOME/bin/` directory and use the `./splunk` command.

If you get stuck, Splunk's CLI has built-in help. Access the main CLI help by typing `splunk help`. Individual commands have their own help pages as well -- type `splunk help <command>`.

The following commands are available for input configuration via the CLI:

Command	Command syntax	Action
add	`add tcp | udp $SOURCE [-parameter value] ...`	Add inputs from $SOURCE.
edit	`edit tcp | udp $SOURCE [-parameter value] ...`	Edit a previously added input for $SOURCE.

remove	`remove tcp | udp $SOURCE`	Remove a previously added data input.
list	`list tcp | udp`	List the currently configured monitor.

Change the configuration of each data input type by setting additional parameters. Parameters are set via the syntax: `-parameter value`.

Parameter	Required?	Description
$SOURCE	Require	Port number to listen for data to index.
sourcetype	Optional	Specify a sourcetype field value for events from the input source.
index	Optional	Specify the destination index for events from the input source.
hostname	Optional	Specify a host name to set as the host field value for events from the input source.
remotehost	Optional	Specify an IP address to exclusively accept data from.
resolvehost	Optional	Set True of False (T | F). Default is False. Set True to use DNS to set the host field value for events from the input source.

Example

Configure a network input, then set the source type:

- Configure a UDP input to watch port 514 and set the source type to "syslog".

Check the Best Practices Wiki for information about the best practices for using UDP when configuring Syslog input.

```
./splunk add udp 514 -sourcetype syslog
```

- Set the UDP input's host value via DNS. Use `auth` with your username and password.

```
./splunk edit udp 514 -resolvehost true -auth admin:changeme
```

Note: Splunk must be running as root to watch ports under 1024.

Add a network input using inputs.conf

To add an input, add a stanza for it to inputs.conf in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. If you have not worked with Splunk's configuration files before, read "About configuration files" in this manual before you begin.

You can set any number of attributes and values following an input type. If you do not specify a value for one or more attributes, Splunk uses the defaults that are preset in `$SPLUNK_HOME/etc/system/default/` (noted below).

TCP

```
[tcp://<remote server>:<port>]
<attrbute1> = <val1>
<attrbute2> = <val2>
...
```

This type of input stanza tells Splunk to listen to <remote server> on <port>. If <remote server> is blank, Splunk listens to all connections on the specified port.

`host = <string>`

- Set the host value of your input to a static value.
- `host::` is automatically prepended to the value when this shortcut is used.
- Defaults to the IP address of fully qualified domain name of the host where the data originated.

`index = <string>`

- Set the index where events from this input will be stored.
- `index::` is automatically prepended to the value when this shortcut is used.
- Defaults to `main` (or whatever you have set as your default index).
- For more information about the index field, see "How indexing works" in this manual.

`sourcetype = <string>`

- Set the sourcetype name of events from this input.
- `sourcetype::` is automatically prepended to the value when this shortcut is used.
- Splunk automatically picks a source type based on various aspects of your data. There is no hard-coded default.
- For more information about the sourcetype field, read about source types in this manual.

`source = <string>`

- Set the source name of events from this input.
- Defaults to the file path.
- `source::` is automatically prepended to the value when this shortcut is used.

`queue = <string> (parsingQueue, indexQueue, etc)`

- Specify where the input processor should deposit the events that it reads.
- Can be any valid, existing queue in the pipeline.
- Defaults to `parsingQueue`.

`connection_host = [ip | dns]`

- If set to ip: the TCP input processor rewrites the host with the ip address of the remote server.
- If set to dns: the host is rewritten with the DNS entry of the remote server.
- Defaults to ip.

UDP

```
[udp://<port>]
<attrbute1> = <val1>
<attrbute2> = <val2>
...
```

This type of input stanza is similar to the TCP type, except that it listens on a UDP port.

`host = <string>`

- Set the host value of your input to a static value.
- `host=` is automatically prepended to the value when this shortcut is used.
- Defaults to the IP address of fully qualified domain name of the host where the data originated.

`index = <string>`

- Set the index where events from this input will be stored.
- `index=` is automatically prepended to the value when this shortcut is used.
- Defaults to `main` (or whatever you have set as your default index).
- For more information about the index field, read about how indexing works in this manual.

`sourcetype = <string>`

- Set the sourcetype name of events from this input.
- `sourcetype=` is automatically prepended to the value when this shortcut is used.
- Splunk automatically picks a source type based on various aspects of your data. There is no hard-coded default.
- For more information about the sourcetype field, read about source types in this manual.

`source = <string>`

- Set the source name of events from this input.
- Defaults to the file path.
- `source=` is automatically prepended to the value when this shortcut is used.

`queue = <string> (parsingQueue, indexQueue, etc)`

- Specify where the input processor should deposit the events that it reads.
- Can be any valid, existing queue in the pipeline.
- Defaults to `parsingQueue`.

`_rcvbuf = <int>`

- Specify the receive buffer for the UDP port (in bytes).
- If the value is 0 or negative, it is ignored.
- Defaults to 1,572,864.
- Note: The default in the OS varies.

`no_priority_stripping = true | false`

- If this attribute is set to true, then Splunk does NOT strip the <priority> syslog field from received events.
- Otherwise, Splunk strips syslog priority from events.

```
no_appending_timestamp = true
```

- If this attribute is set to true, then Splunk does NOT append a timestamp and host to received events.
- **Note**: Do NOT include this key at all if you want to append timestamp and host to received events.

Answers

Have questions? Visit Splunk Answers and see what and answers the Splunk community has about questions UDP inputs, TCP inputs, and inputs in general,

Considerations for deciding how to monitor remote Windows data

Considerations for deciding how to monitor remote Windows data

Use forwarders or remote collection?

The best way to get data off of a Windows host is with local Splunk forwarder. Using a local forwarder offers the most types of data sources, minimizes network overhead and reduces operational risk and complexity.

However, there are circumstances ? from organizational boundaries to local performance considerations ? where remote collection is preferred. For these situations, Splunk supports using the native WMI interface on Windows to collect event logs and performance data.

This table offers a list of data sources and their respective trade-offs for you to consider.

Data Source Considerations

Data Source	Local Forwarder	Remote Polling
Event logs	Yes	Yes*
Performance	Yes	Yes
Registry	Yes	No
Log files	Yes	Yes**
Crawl	Yes	No

* For remote event log collection, you *must* know the name of the Event Log you wish to collect. On local forwarders, you have the option to collect all logs, regardless of name.

** Remote log file collection using {\\servername\share\} syntax is supported, however you must use CIFS as your application layer file access protocol and Splunk must have at least read access to both

the share and the underlying file system.

Tradeoffs

Performance

For identical collecting local Event Logs and flat log files, a local forwarder requires less CPU and performs basic pre-compression of the data; it is more memory intensive, mostly owing to the additional data source input options. WMI remote polling is more CPU intensive on the target for the same set of data (either remote Event Logs or remote performance data) and more network intensive.

Note that for highly audited hosts, such as domain controllers, remote polling may not be able to keep up with the volume of data or Event Log events. Remote polling is best-effort by design of the WMI API, and is throttled to prevent unintentional denial of service attacks.

Deployment

Local forwarders are easier where you have control of the base OS build, and/or if you have many data sources, especially if transformation of data is required. Remote polling works well when you want a limited set of data from a large number of hosts (for example, just process CPU data for usage billing). Remote polling may be your only option where you don?t have either build control or local administrator privileges on the target host.

A common usage scenario is to first test using remote polling, then add successful or useful polls to your local forwarding configuration later, or at mass deployment time.

Management

Both mechanisms offer logging and, potentially, alerting to let you know if a host is coming on or offline or is no longer connected. However, to prevent an unintentional denial of service attack the WMI polling service in Splunk will start to poll less frequently over time if it is unable to contact a host for a period of time. Therefore remote polling is not advised for machines that are frequently offline, such as laptops or dynamically provisioned virtual machines.

Search Windows data on a non-Windows Splunk

You can index and search your Windows data on a non-Windows instance of Splunk, but you must first use a Windows instance of Splunk to gather the Windows data. You can easily do this by means of a Splunk forwarder running on Windows, configured to gather Windows inputs and then forward the data to the non-Windows instance of Splunk where searching and indexing will take place.

There are two main ways to proceed:

- Set up light forwarders locally on each Windows machine that's generating data. These forwarders can send the Windows data to the non-Windows receiving instance of Splunk.
- Set up a regular forwarder on a separate Windows machine. The fowarder can perform remote polling on all the Windows machines in the environment and then forward the combined data to a non-Windows receiving instance of Splunk.

You must specially configure the non-Windows Splunk to handle the Windows data. For details, see "Searching data received from a forwarder running on a different operating system".

For information on setting up forwarders, see "Set up forwarding and receiving".

Monitor Windows event log data

Monitor Windows event log data

This topic discusses ways to configure Splunk to monitor Windows event logs. You can configure this via Splunk Web or via configuration files.

You can monitor two types of event log collections:

- Local
- Remote

Note: To add another log channel to monitor on localhost, edit the existing input. To monitor a remote machine, add a new input.

Use Splunk Web to configure event log monitoring

Configure local event log monitoring

1. Click **Manager** in the upper right-hand corner of Splunk Web.

2. Under System configurations, click **Data Inputs**.

3. Click **Local event log collections**.

4. Click **Add new** to add an input.

5. Select one or more logs from the list of **Available Logs** and click to add to the list of **Selected Logs**.

Note 1: Select up to 63 logs from the list of Available Logs. Selecting more than 63 can cause Splunk to become unstable.

Note 2: Certain Windows Event Log channels (known as **direct channels**) do not allow for users to access - or subscribe to - them in order to monitor them. This is because events sent via these log channels are not actually processed by the Windows Event Log framework, and thus can't be forwarded or collected remotely. Attempts to monitor these log channels will generate the error: "The caller is trying to subscribe to a direct channel which is not allowed."

6. Click **Save**.

The input is added and enabled.

Configure remote event log monitoring

1. Click **Manager** in the upper right-hand corner of Splunk Web.

2. Under System configurations, click **Data Inputs**.

3. Click **Remote event log collections**.

4. Click **Add new** to add an input.

5. Enter a unique name for this collection.

6. Specify a hostname or IP address for the host from which to pull logs, and click **Find logs...** to get a list of logs from which to choose.

Note: Windows Vista offers many channels; depending on the CPU available to Splunk, selecting all or a large number of them can result in high load.

7. Optionally, provide a comma-separated list of additional servers from which to pull data.

8. Click **Save**.

The input is added and enabled.

Use inputs.conf to configure event log monitoring

To edit `inputs.conf`:

1. Copy inputs.conf from `%SPLUNK_HOME%\etc\system\default` to `etc\system\local`.

2. Use Explorer or the `ATTRIB` command to remove the file's "Read Only" flag.

3. Open the file and edit it to enable Windows event log inputs.

4. Restart Splunk.

The next section describes the specific configuration values for event log monitoring.

Event log monitoring configuration values

Windows event log (*.evt) files are in binary format. As such, they cannot be monitored like a flat file. The settings for which event logs to index are in the following stanza in `inputs.conf`:

```
# Windows platform specific input processor.
[WinEventLog:Application]
disabled = 0
[WinEventLog:Security]
disabled = 0
[WinEventLog:System]
disabled = 0
```

You can configure Splunk to read non-default Windows event logs as well, but you must first import them to the Windows Event Viewer first, and then add them to your local copy of `inputs.conf`, (usually in `%SPLUNK_HOME%\etc\system\local\inputs.conf`) as follows:

```
[WinEventLog:DNS Server]
disabled = 0
[WinEventLog:Directory Service]
disabled = 0
[WinEventLog:File Replication Service]
disabled = 0
```

To disable indexing for an event log, add `disabled = 1` below its listing in the stanza in `%SPLUNK_HOME%\etc\system\local\inputs.conf`.

If you've added some non-standard event log channels and you want to specify whether Active Directory objects like GUIDs and SIDs are resolved for a given Windows event log channel, you can turn on the `evt_resolve_ad_obj` setting (1=enabled, 0=disabled) for that channel's stanza in your local copy of `inputs.conf`. **`evt_resolve_ad_obj` is on by default for the Security channel.**

To specify the Domain Controller name and/or DNS name of the domain to bind to for Splunk to use to resolve the AD objects, use the `evt_dc_name` and/or `evt_dns_name` settings along with `evt_resolve_ad_obj`. This name can be the 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 are correctly formatted domain controller names:

- "FTW-DC-01"
- "\\FTW-DC-01"
- "FTW-DC-01.splunk.com"
- "\\FTW-DC-01.splunk.com"

Specify whether to index starting at earliest or most recent event

Use these settings to specify in which chronological order you want to index the events, from oldest->newest or newest->oldest, and whether you want to index all pre-existing events, or just new events.

```
start_from = oldest
current_only = 1
```

- `start_from`: By default, Splunk starts with the oldest data and indexes forward. You can set it to `newest`, telling Splunk to start with the newest data and index backward. We don't recommend changing this setting, as it results in a highly inefficient indexing process.
- `current_only`: This option allows you to only index new events that appear from the moment Splunk was started. When set to 1, it is enabled. When set to 0, it is disabled and all events are indexed.

Index exported event log (.evt or .evtx) files

To index exported Windows event log files, use the instructions for monitoring files and directories to monitor the directory into which you place these exported files.

Constraints

- As a result of API and log channel processing constraints on Windows XP and 2003 systems, imported .evt files will not contain the message field. This means that the message field will not appear in your Splunk index.
- Splunk running on Windows 2000/2003/XP cannot index Vista/2008/Windows 7 .evtx files.
- Splunk running on Vista/2008/Windows 7 can index both .evt and .evtx files.
- If your .evt/.evtx file is not from a standard event log channel, you must make sure that any DLL files required by that channel are present on the computer on which you are indexing.
- The language that a .evt/.evtx file will be indexed as is the primary locale/language of the Splunk computer that collects the file.

Caution: Do not attempt to monitor a .evt or .evtx file that is currently being written to; Windows will not allow read access to these files. Use the event log monitoring feature instead.

Note: When producing .evt/.evtx files on one system, and monitoring them on another, it's possible to not have all fields expanded as they would be on the producing system. This is caused by variations in DLL availability and APIs. Differences in OS version, language, patch level, installed third party DLLs, etc. can have this effect.

Answers

Have questions? Visit Splunk Answers and see what questions and answers the Splunk community has around Windows event logs.

Monitor Windows Registry data

Monitor Windows Registry data

Splunk supports the capture of Windows registry settings and lets you monitor changes to the registry. You can know when registry entries are added, updated, and deleted. When a registry entry is changed, Splunk captures the name of the process that made the change and the key path from the hive to the entry being changed.

The Windows registry input monitor application runs as a process called `splunk-regmon.exe`.

Warning: Do not stop or kill the `splunk-regmon.exe` process manually; this could result in system instability. To stop the process, stop the Splunkd server process from the Services control panel.

Enable Registry monitoring in Splunk Web

Splunk on Windows comes with Registry monitoring configured but disabled by default. You can perform a one-time baseline index and then separately enable ongoing monitoring for machine and/or user keys. To do this:

1. In Splunk Web, click **Manager** in the upper right corner.

2. Click **Data inputs > Registry Monitoring**

3. Choose **Machine keys** or **User keys** and enable the baseline and ongoing monitoring as desired.

4. Click **Save**.

How it works: the details

Windows registries can be extremely dynamic (thereby generating a great many events). Splunk provides a two-tiered configuration for fine-tuning the filters that are applied to the registry event data coming into Splunk.

Splunk Windows registry monitoring uses two configuration files to determine what to monitor on your system, sysmon.conf and the filter rules file referenced by it. By default, the filter rules file is named regmon-filters.conf, but you can define its name within `sysmon.conf` by using the `filter_file_name` attribute. Both of these files need to reside in `$SPLUNK_HOME\etc\system\local\`.

The two configuration files work as a hierarchy:

- `sysmon.conf` contains global settings for which event types (adds, deletes, renames, and so on) to monitor, which regular expression filters from the filter rules file to use, and whether or not Windows registry events are monitored at all.
- The filter rules file (by default named `regmon-filters.conf`) contains the specific regular expressions you create to refine and filter the Registry hive key paths you want Splunk to monitor.

`sysmon.conf` contains only one stanza, where you specify:

- `event_types`: the superset of registry event types you want to monitor. Can be any of `delete, set, create, rename, open, close, query`.
- `filter_file_name`: the file that Splunk should access for filter rules for this monitor. For example, if the attribute is set to `regmon-filters`, then Splunk looks in `regmon-filters.conf` for filter rule information.
- `inclusive`: whether the filter rules listed in the file specified by `filter_file_name` are inclusive (meaning Splunk should only monitor what is listed there) or exclusive (meaning Splunk should monitor everything except what is listed there). Set this value to 1 to make the filter rules inclusive, and 0 to make them exclusive.
- `disabled`: whether to monitor registry settings changes or not. Set this to 1 to disable Windows registry monitoring altogether.

Each stanza in `regmon-filters.conf` represents a particular filter whose definition includes:

- `proc`: a regular expression containing the path to the process or processes you want to monitor
- `hive`: a regular expression containing the hive path to the entry or entries you want to monitor. Splunk supports the root key value mappings predefined in Windows:
  - `\\REGISTRY\\USER\\` maps to `HKEY_USERS` or `HKU`
  - `\\REGISTRY\\USER\\_Classes` maps to `HKEY_CLASSES_ROOT` or `HKCR`
  - `\\REGISTRY\\MACHINE` maps to `HKEY_LOCAL_MACHINE or HKLM`
  - `\\REGISTRY\\MACHINE\\SOFTWARE\\Classes` maps to `HKEY_CLASSES_ROOT` or `HKCR`
  - `\\REGISTRY\\MACHINE\\SYSTEM\\CurrentControlSet\\Hardware`

Profiles\\Current maps to HKEY_CURRENT_CONFIG or HKCC
    - **Note:** There is no direct mapping for HKEY_CURRENT_USER or HKCU, as the Splunk Registry monitor runs in kernel mode. However, using \\REGISTRY\\USER\\.* (note the period and asterisk at the end) will generate events that contain the logged-in user's SID.
    - Alternatively, you can specify the user whose registry keys you wish to monitor by using \\REGISTRY\\USER\\<SID>, where SID is the SID of the desired user.
- type: the subset of event types to monitor. Can be delete, set, create, rename, open, close, query. The values here must be a subset of the values for event_types that you set in sysmon.conf.
- baseline: whether or not to capture a baseline snapshot for that particular hive path. Set to 0 for no, and 1 for yes.
- baseline interval: how long Splunk has to have been down before re-taking the snapshot, in seconds. The default value is 24 hours.
- disabled: whether or not a filter is enabled. 0 means it is, 1 means it is not.

Get a baseline snapshot

When you enable Registry monitoring, you're given the option of recording a baseline snapshot of your registry hives the next time Splunk starts. By default, the snapshot covers the entirety of the user keys and machine keys hives. It also establishes a timeline for when to retake the snapshot; by default, if Splunk has been down for more than 24 hours since the last checkpoint, it will retake the baseline snapshot. You can customize this value for each of the filters in regmon-filters.conf by setting the value of baseline_interval.

Note: The baseline_interval attribute is expressed in seconds.

Note: Executing a splunk clean all -f from the CLI deletes the current baseline snapshot.

What to consider

When you install Splunk on a Windows machine and enable registry monitoring, you specify which major hive paths to monitor: key users (HKEY) and/or key local machine (HKLM). Depending on how dynamic you expect the registry to be on this machine, checking both could result in a great deal of data for Splunk to monitor. If you're expecting a lot of registry events, you may want to specify some filters in regmon-filters.conf to narrow the scope of your monitoring immediately after you install Splunk and enable registry event monitoring but before you start Splunk up.

Similarly, you have the option of capturing a baseline snapshot of the current state of your Windows registry when you first start Splunk, and again every time a specified amount of time has passed. The baselining process can be somewhat processor-intensive, and may take several minutes. You can postpone taking a baseline snapshot until you've edited regmon-filters.conf and narrowed the scope of the registry entries to those you specifically want Splunk to monitor.

Change the default Windows registry input values

Look at inputs.conf to see the default values for Windows registry input. They are also shown below.

To make changes to the default values, edit **a copy** of inputs.conf in $SPLUNK_HOME\etc\system\local\. Provide new values for only the parameters you want to

change within the `[script://$SPLUNK_HOME\bin\scripts\splunk-regmon.path]` stanza. There's no need to edit the other values. For more information about how to work with Splunk configuration files, refer to "About configuration files".

```
[script://$SPLUNK_HOME\bin\scripts\splunk-regmon.path]
interval = 60
sourcetype = WinRegistry
source = WinRegistry
disabled = 0
```

- `source`: labels these events as coming from the registry.
- `sourcetype`: assigns these events as registry events.
- `interval`: specifies how frequently to poll the registry for changes, in seconds.
- `disabled`: indicates whether the feature is enabled. Set this to 1 to disable this feature.

Note: The Splunk registry input monitoring script (`splunk-regmon.path`) is configured as a **scripted input**. Do not change this value.

Note: You must use two backslashes `\\` to escape wildcards in stanza names in `inputs.conf`. Regexes with backslashes in them are not currently supported when specifying paths to files.

Monitor WMI data

Monitor WMI data

Splunk supports WMI (Windows Management Instrumentation) data input for agentless access to Windows performance data and event logs. This means you can pull event logs from all the Windows servers and desktops in your environment without having to install anything on those machines.

The Splunk WMI data input can connect to multiple WMI providers and pull data from them. The WMI data input runs as a separate process (`splunk-wmi.exe`) on the Splunk server. It is configured as a **scripted input** in `%SPLUNK_HOME%\etc\system\default\inputs.conf`. Do not make changes to this file.

Note: This feature is only available on the Windows versions of Splunk.

Security and remote access considerations

Splunk requires privileged access to index many Windows data sources, including WMI, Event Logs, and the registry. This includes both the ability to connect to the computer you wish to poll, as well as permissions to read the appropriate data once connected. To access WMI data, Splunk must run as a user with permissions to perform remote WMI connections. This user name must be a member of an Active Directory domain, and must have appropriate privileges to query WMI. Both the Splunk server making the query and the target systems being queried must be part of this Active Directory domain.

Note: If you installed Splunk as the LOCAL SYSTEM user, WMI remote authentication will not work; this user has null credentials and Windows servers normally disallow such connections.

There are several things to consider:

- For remote data collection via WMI, the Splunk service must run as a user who has sufficient OS privileges to access the WMI resources you wish to poll. At a minimum, Splunk requires access to the following privileges on every machine you poll:
    - **Profile System Performance**
    - **Access this Computer from the Network**
    - The simplest way to ensure Splunk has access to these resources is to add Splunk's user to the **Performance Log Users** and **Distributed COM Users** Domain groups. If these additions fail to provide sufficient permissions, add Splunk's user to the remote machine's **Administrators** group.
- You must enable DCOM for remote machine access, and it must be accessible to Splunk's user. See the Microsoft topic about Securing a Remote WMI Connection for more information. Adding Splunk's user to the **Distributed COM Users** local group is the fastest way to enable this permission. If this fails to provide sufficient permissions, add Splunk's user to the remote machine's **Administrators** group.
- The WMI namespace that Splunk accesses (most commonly `root\cimv2`) must have proper permissions set. Enable the following permissions on the WMI tree at root for the Splunk user:
    - **Execute Methods, Enable Account, Remote Enable**, and **Read Security**.
    - See the Microsoft HOW TO: Set WMI Namespace Security in Windows Server 2003 for more information.
- If you have a firewall enabled, you must allow access for WMI. If you are using the Windows Firewall, the exceptions list explicitly lists WMI. You must set this exception for both the originating and the remote machine. See the Microsoft topic about Connecting to WMI Remotely Starting with Vista for more details.

Test access to WMI

Follow these steps to test the configuration of the Splunk server and the remote machine:

1. Log into the machine Splunk runs on as the user Splunk runs as.

2. Open a command prompt window (click **Start -> Run** and type `cmd`).

3. Go to the `bin` sub-directory under your Splunk installation (for example, `cd c:\Program Files\Splunk\bin`).

4. Run the following command: `splunk cmd splunk-wmi -wql "select * from win32_service" -namespace \\<server>\root\cimv2`, replacing `<server>` with the name of the remote server.

5. If you see data streaming back and no error message, that means your were able to connect and query successfully. If there was an error, there would be a message with a reason on why it failed (look for the `error="<msg>"` string).

Configure WMI input

You can configure WMI input either in Splunk Web or by editing configuration files. More options are available when using the configuration file option.

Configure WMI with Splunk Web

1. Click **Manager** in the upper right-hand corner of Splunk Web.

2. Under System configurations, click **Data Inputs**.

3. Click **WMI collections**.

4. Click **Add new** to add an input.

5. Enter a unique name for this collection.

6. Enter a target host and click **Query...** to get a list of the available classes of properties to choose from.

Note: Only classes that are prefixed with Win32_PerfFormattedData_* are displayed in the list. If the class you wish to index does not start with Win32_PerfFormattedData_* prefix, you must add them by editing `wmi.conf`.

7. Optionally, provide a comma-separated list of additional servers from which to pull data.

8. Specify an interval in seconds between polls.

9. Ensure the **Enabled?** radio button is set to **Yes** and click **Save**.

The input is added and enabled.

Disabling or deleting WMI inputs

To disable a WMI input, navigate to it and select the **No** radio button under **Enabled?**.

It's not possible to delete a WMI input from within Splunk Web. To delete a WMI input, use the information in the section below titled "Configure WMI with configuration files" to edit the wmi.conf file and remove the [WMI:<input_name>] stanza you'd like to delete.

Configure WMI with configuration files

Look at wmi.conf to see the default values for the WMI input. If you want to make changes to the default values, edit a copy of `wmi.conf` in `%SPLUNK_HOME%\etc\system\local\`. Only set values for the attributes you want to change for a given type of data input. Refer to About configuration files for more information about how Splunk uses configuration files.

```
[settings]
initial_backoff = 5
max_backoff = 20
max_retries_at_max_backoff = 2
checkpoint_sync_interval = 2

[WMI:AppAndSys]
server = foo, bar
interval = 10
event_log_file = Application, System, Directory Service
```

```
disabled = 0

[WMI:LocalSplunkWmiProcess]
interval = 5
wql = select * from Win32_PerfFormattedData_PerfProc_Process where Name = "splunk-wmi"
disabled = 0

# Listen from three event log channels, capturing log events that occur only
# while Splunk is running.  Gather data from three servers.
[WMI:TailApplicationLogs]
interval = 10
event_log_file = Application, Security, System
server = srv1, srv2, srv3
disabled = 0
current_only = 1

# Listen for process-creation events on a remote machine
[WMI:ProcessCreation]
interval = 1
server = remote-machine
wql = select * from __InstanceCreationEvent within 1 where TargetInstance isa 'Win32_Process'
disabled = 0
current_only = 1

# Receive events whenever someone plugs/unplugs a USB device to/from the computer
[WMI:USBChanges]
interval = 1
wql = select * from __InstanceOperationEvent within 1 where TargetInstance ISA 'Win32_PnPEntity
disabled = 0
current_only = 1
```

The `[settings]` stanza specifies runtime parameters. The entire stanza and every parameter within it are optional. If the stanza is missing, Splunk assumes system defaults.

- The following attributes control how the agent reconnects to a given WMI provider when an error occurs. All times are in seconds:
  - `initial_backoff`: how much time to wait the first time after an error occurs before trying to reconnect. Thereafter, if errors keep occurring, the wait time doubles, until it reaches `max_backoff`.
  - `max_backoff`: the maximum amount of time to wait before invoking `max_retries_at_max_backoff`.
  - `max_retries_at_max_backoff` : if the wait time reaches `max_backoff`, try this many times at this wait time. If the error continues to occur, Splunk will not reconnect to the WMI provider in question until the Splunk services are restarted.
- `checkpoint_sync_interval`: minimum wait time for state data (event log checkpoint) to be written to disk. In seconds.

You can specify two types of data input: event log, and raw WQL (WMI query language) The event log input stanza contains the `event_log_file` parameter, and the WQL input stanza contains `wql`.

The common parameters for both types are:

- `server`: a comma-separated list of servers from which to pull data. If this parameter is missing, Splunk assumes the local machine.

- `interval` : how often to poll for new data, in seconds. Required.
- `disabled`: indicates whether this feature is enabled or disabled. Set this parameter to 1 to disable WMI input into Splunk.

WQL-specific parameters:

- `namespace`: specifies the path to the WMI provider. The local machine must be able to connect to the remote machine using delegated authentication. This attrbitue is optional. If you don't specify a path to a remote machine, Splunk will connect to the default local namespace (`\root\cimv2`), which is where most of the providers you are likely to query reside. Microsoft provides a list of namespaces for Windows XP and later versions of Windows.
- `wql`: provides the WQL query. The example above polls data about a running process named `splunkd` every 5 seconds.

Event log-specific parameters:

- `event_log_file`: specify a comma-separated list of log files to poll in the event_log_file parameter. File names that include spaces are supported, as shown in the example.
- `current_only`: If enabled (or set to `1`), this option allows you to collect events that occur only while Splunk is running. It behaves like a tail to a file.

You can also use the `current_only` parameter in raw WQL stanzas to collect WMI event notification data. Check the `wmi.conf` configuration file reference for examples.

Fields for WMI data

All events received from WMI have the source set to `wmi`.

- For event log data, the source type is set to `WinEventLog:<name of log file>` (for example `WinEventLog:Application`).
- For WQL data, the the source type is set to the name of the config stanza (for example, for a stanza named `[WMI:LocalSplunkdProcess]`, the field is set to `WMI:LocalSplunkdProcess`).

The host is identified automatically from the data received.

Monitor Active Directory

Monitor Active Directory

Configure Active Directory monitoring as an input to monitor changes to portions of, or all of, your AD forest and collect user and machine metadata.

Once you've enabled this feature and restarted Splunk it will take a baseline snapshot of your AD data and the AD schema. It'll use this data to get a starting point against which to monitor. This process might take a little time before it is complete.

Powerful lookups from your AD data

You can use this feature combined with dynamic list lookups to decorate or modify events with any information available in AD. Read an overview of how in this topic on the Splunk Community Wiki.

Things to know

- This feature is only available on Windows platforms.
- The admon.exe process can run under a full Splunk install or within a forwarder.
- The machine the admon.exe process is running on must belong to the domain you want to monitor.
- The user Splunk is running as must be part of the domain too; whatever rights that user has to query to AD will filter the results Splunk can see.
- You can use the Windows permissions of the user admon.exe is running as to control the level of access Splunk should have and what it should be allowed to see. Note that the AD user rights policy set in Group Policy Manager can further restrict access.

For more details, see this topic about choosing the user Splunk should run as in the Installation Manual.

Configure monitoring

You can configure AD monitoring either in Splunk Web or by editing configuration files.

Configure AD monitoring with Splunk Web

1. Click **Manager** in the upper right-hand corner of Splunk Web.

2. Under System configurations, click **Data Inputs**.

3. Click **Active Directory monitoring**.

4. Click **Add new** to add an input.

5. Enter a unique **Name** for the AD monitor.

6. Either enter a **Target domain controller** or let Splunk discover the nearest domain controller automatically.

7. Either select a **Starting node** or Splunk will start monitoring from the highest available part of the tree.

8. Select **Monitor subtree**, if you want Splunk to monitor all child nodes.

10. Click **Save**.

Configure AD monitoring in inputs.conf and admon.conf

Be sure to edit copies of these configuration files in a \local directory. If you edit them in the default directory, any changes you make will be overwritten when you upgrade Splunk. For more information about configuration file precedence, refer to "About configuration files" in this manual.

1. Make a copy of `$SPLUNK_HOME\etc\system\default\inputs.conf` and put it in `$SPLUNK_HOME\etc\system\local\inputs.conf`.

2. Edit the copy and enable the **scripted input** `[script://$SPLUNK_HOME\bin\scripts\splunk-admon.path]` by setting the value of `disabled` to 0.

3. Next, make a similar copy of `$SPLUNK_HOME\etc\system\default\admon.conf` and put it in `$SPLUNK_HOME\etc\system\local\admon.conf`.

4. Edit it using the information later in this topic. By default, when enabled, it will index the first domain controller that the admon.exe process can attach to. If that is acceptable, no further configuration is necessary; it will just work.

Settings in admon.conf

`monitorSubtree` tells Splunk how much of the target container to index. A value of 0 will tell Splunk to only index the target container. A value of of 1 (the default) will tell Splunk to enumerate all sub-containers and domains it has access to.

`targetDC` sets the unique name of the domain controller host you want to monitor. Specify a unique name if:

- you have a very large AD and you only want to monitor information in a particular branch (ou), subdomain, etc.
- you want to limit your scope to only a certain subdomain of your tree.
- you have a specific (read-only) domain controller that is offered for this purpose in a high security environment.
- if you have multiple domain forests in a trusted configuration, you can use this to target a different tree than the one where Splunk resides.
- if you want to run multiple instances of admon.exe to target multiple Domain Controllers, for example, to monitor replication health across a distributed environment.

If you want to target multiple DCs, add another `[<uniquename>TargetDC]` stanza for a target in that tree.

`startingNode` is a fully qualified LDAP name (for example `"LDAP://OU=Computers,DC=ad,DC=splunk,DC=com"`) where Splunk will begin its indexing. Splunk starts there and enumerates down to sub-containers, depending on the configuration of `monitorSubtree` above. If you don't specify something, it will start at the highest root domain in the tree it can access.

The `startingNode` must be within the scope of the DC you are targeting to be successful.

Example AD monitoring configurations

You can monitor monitor a target DC that is a higher root level than an OU you want to target, for example:

The OU = computers in the eng.ad.splunk.com subdomain.

Target your DC to be one of the controllers in ad.splunk.com. The reason one might do this is if you want the schema for the entire tree, not just a sub-domain. Then set the starting node to be an OU in eng.ad.splunk.com to audit machines being added and removed from that OU.

```
[default]
monitorSubtree = 1
disabled = 0

[DefaultTargetDC]
targetDC = pri01.eng.ad.splunk.com
startingNode = OU=Computers,DC=eng,DC=ad,DC=splunk,DC=com
```

You can monitor multiple DCs, for example:

```
[default]
monitorSubtree = 1
disabled = 0

[DefaultTargetDC]
targetDC = pri01.eng.ad.splunk.com
startingNode = OU=Computers,DC=eng,DC=ad,DC=splunk,DC=com

[SecondTargetDC]
targetDC = pri02.eng.ad.splunk.com
startingNode = OU=Computers,DC=eng,DC=ad,DC=splunk,DC=com
```

Sample admon output

The following are examples of the types of admon events that Splunk indexes. Some of the content of these events has been obscured/altered for publication purposes.

Delete event

An object has been marked for deletion. Even though `admonEventType=Update`, notice the `isDeleted=True` at the end of the event.

```
2/1/10
3:11:16.095 PM

02/01/2010 15:11:16.0954
dcName=stuff.splunk.com
admonEventType=Update
Names:
                name=SplunkTest
DEL:blah
                distinguishedName=OU=SplunkTest\0ADEL:blah,CN=Deleted Objects
DEL:blah
Object Details:
                objectGUID=blah
                whenChanged=20100128233113.0Z
                whenCreated=20100128232712.0Z
                objectClass=top|organizationalUnit
Event Details:
                uSNChanged=2922895
                uSNCreated=2922846
                instanceType=4
```

```
Additional Details:
                dSCorePropagationData=20100128233113.0Z|20100128233113.0Z|20100128233113.0Z|160
                lastKnownParent=stuff
                isDeleted=TRUE
```

Update event

Update type event: admonEventType=Update An object has been changed, this includes a change to any of the object's fields.

```
2/1/10
3:17:18.009 PM

02/01/2010 15:17:18.0099
dcName=stuff.splunk.com
admonEventType=Update
Names:
                objectCategory=CN=Computer,CN=Schema,CN=Configuration
                name=stuff2
                displayName=stuff2
                distinguishedName=CN=stuff2,CN=Computers
Object Details:
                sAMAccountType=805306369
                sAMAccountName=stuff2
                logonCount=4216
                accountExpires=9223372036854775807
                objectSid=S-1-5-21-3436176729-1841096389-3700143990-1190
                primaryGroupID=515
                pwdLastSet=129091141316250000
                lastLogon=129095398380468750
                lastLogoff=0
                badPasswordTime=0
                countryCode=0
                codePage=0
                badPwdCount=0
                userAccountControl=4096
                objectGUID=blah
                whenChanged=20100128010211.0Z
                whenCreated=20081125172950.0Z
                objectClass=top|person|organizationalPerson|user|computer
Event Details:
                uSNChanged=2921916
                uSNCreated=1679623
                instanceType=4
Additional Details:
                isCriticalSystemObject=FALSE
                servicePrincipalName=TERMSRV/stuff2|TERMSRV blah
                dNSHostName=stuff2.splunk.com
                operatingSystemServicePack=Service Pack 2
                operatingSystemVersion=6.0 (6002)
                operatingSystem=Windows Vista? Ultimate
localPolicyFlags=0
```

Sync event

Sync type event: admonEventType=Sync Represents the instance of one object, and its field values. Splunk syncs up to the very beginning, trying to capture all of the objects from the last recorded USN.

```
2/1/10
3:11:09.074 PM

02/01/2010 15:11:09.0748
dcName=ftw.ad.splunk.com
admonEventType=Sync
Names:
                name=NTDS Settings
                distinguishedName=CN=NTDS Settings,CN=stuff,CN=Servers,CN=Default-First-Site-Na
                cn=NTDS Settings
                objectCategory=CN=NTDS-DSA,CN=Schema,CN=Configuration,DC=ad,DC=splunk,DC=com
                fullPath=LDAP://stuff.splunk.com/<GUID=bla bla bla>
                CN=NTDS Settings
Object Details:
                whenCreated=10:15.04 pm, Tue 02/12/2008
                whenChanged=10:23.00 pm, Tue 02/12/2008
                objectGUID=bla bla bla
                objectClass=top|applicationSettings|nTDSDSA
                classPath=nTDSDSA
Event Details:
                instanceType=4
Additional Details:
                systemFlags=33554432
                showInAdvancedViewOnly=TRUE
                serverReferenceBL=CN=stuff,CN=Domain System Volume (SYSVOL share),CN=File Repli
                options=1
                msDS-hasMasterNCs=DC=ForestDnsZones|DC=DomainDnsZones|CN=Schema,CN=Configuratio
                msDS-HasInstantiatedNCs=
                msDS-HasDomainNCs=blah
                msDS-Behavior-Version=2
                invocationId=bla bla bla
                hasMasterNCs=CN=Schema,CN=Configuration|CN=Configuration
                dSCorePropagationData=
                dMDLocation=CN=Schema,CN=Configuration
                nTSecurityDescriptor=NT AUTHORITY\Authenticated Users
SchemaName=LDAP://stuff.splunk.com/schema/nTDSDSA
```

Schema event

Schema type event: admonEventType=schema The definitions of every object in the Active Directory structure. Listed for each object: which fields are available, required, and optional.

```
02/01/2010 15:11:16.0518
dcName=LDAP://stuff.splunk.com/
admonEventType=schema
className=msExchProtocolCfgSMTPIPAddress
classCN=ms-Exch-Protocol-Cfg-SMTP-IP-Address
instanceType=MandatoryProperties
nTSecurityDescriptor=MandatoryProperties
objectCategory=MandatoryProperties
objectClass=MandatoryProperties
adminDescription=OptionalProperties
adminDisplayName=OptionalProperties
```

```
allowedAttributes=OptionalProperties
allowedAttributesEffective=OptionalProperties
allowedChildClasses=OptionalProperties
allowedChildClassesEffective=OptionalProperties
bridgeheadServerListBL=OptionalProperties
canonicalName=OptionalProperties
cn=OptionalProperties
createTimeStamp=OptionalProperties
description=OptionalProperties
directReports=OptionalProperties
displayName=OptionalProperties
displayNamePrintable=OptionalProperties
distinguishedName=OptionalProperties
dSASignature=OptionalProperties
dSCorePropagationData=OptionalProperties
extensionName=OptionalProperties
flags=OptionalProperties
fromEntry=OptionalProperties
frsComputerReferenceBL=OptionalProperties
fRSMemberReferenceBL=OptionalProperties
fSMORoleOwner=OptionalProperties
heuristics=OptionalProperties
isCriticalSystemObject=OptionalProperties
isDeleted=OptionalProperties
isPrivilegeHolder=OptionalProperties
lastKnownParent=OptionalProperties
legacyExchangeDN=OptionalProperties
managedObjects=OptionalProperties
masteredBy=OptionalProperties
memberOf=OptionalProperties
modifyTimeStamp=OptionalProperties
mS-DS-ConsistencyChildCount=OptionalProperties
mS-DS-ConsistencyGuid=OptionalProperties
msCOM-PartitionSetLink=OptionalProperties
msCOM-UserLink=OptionalProperties
msDFSR-ComputerReferenceBL=OptionalProperties
msDFSR-MemberReferenceBL=OptionalProperties
msDS-Approx-Immed-Subordinates=OptionalProperties
msDs-masteredBy=OptionalProperties
msDS-MembersForAzRoleBL=OptionalProperties
msDS-NCReplCursors=OptionalProperties
msDS-NCReplInboundNeighbors=OptionalProperties
msDS-NCReplOutboundNeighbors=OptionalProperties
msDS-NonMembersBL=OptionalProperties
msDS-ObjectReferenceBL=OptionalProperties
msDS-OperationsForAzRoleBL=OptionalProperties
msDS-OperationsForAzTaskBL=OptionalProperties
msDS-ReplAttributeMetaData=OptionalProperties
msDS-ReplValueMetaData=OptionalProperties
msDS-TasksForAzRoleBL=OptionalProperties
msDS-TasksForAzTaskBL=OptionalProperties
msExchADCGlobalNames=OptionalProperties
msExchALObjectVersion=OptionalProperties
msExchHideFromAddressLists=OptionalProperties
msExchInconsistentState=OptionalProperties
msExchIPAddress=OptionalProperties
msExchTurfList=OptionalProperties
msExchUnmergedAttsPt=OptionalProperties
msExchVersion=OptionalProperties
```

```
msSFU30PosixMemberOf=OptionalProperties
name=OptionalProperties
netbootSCPBL=OptionalProperties
nonSecurityMemberBL=OptionalProperties
objectGUID=OptionalProperties
objectVersion=OptionalProperties
otherWellKnownObjects=OptionalProperties
ownerBL=OptionalProperties
partialAttributeDeletionList=OptionalProperties
partialAttributeSet=OptionalProperties
possibleInferiors=OptionalProperties
proxiedObjectName=OptionalProperties
proxyAddresses=OptionalProperties
queryPolicyBL=OptionalProperties
replicatedObjectVersion=OptionalProperties
replicationSignature=OptionalProperties
replPropertyMetaData=OptionalProperties
replUpToDateVector=OptionalProperties
repsFrom=OptionalProperties
repsTo=OptionalProperties
revision=OptionalProperties
sDRightsEffective=OptionalProperties
serverReferenceBL=OptionalProperties
showInAddressBook=OptionalProperties
showInAdvancedViewOnly=OptionalProperties
siteObjectBL=OptionalProperties
structuralObjectClass=OptionalProperties
subRefs=OptionalProperties
subSchemaSubEntry=OptionalProperties
systemFlags=OptionalProperties
unmergedAtts=OptionalProperties
url=OptionalProperties
uSNChanged=OptionalProperties
uSNCreated=OptionalProperties
uSNDSALastObjRemoved=OptionalProperties
USNIntersite=OptionalProperties
uSNLastObjRem=OptionalProperties
uSNSource=OptionalProperties
wbemPath=OptionalProperties
wellKnownObjects=OptionalProperties
whenChanged=OptionalProperties
whenCreated=OptionalProperties
wWWHomePage=OptionalProperties
```

Answers

Have questions? Visit Splunk Answers and see what questions and answers the Splunk community has around monitoring AD with Splunk.

Monitor FIFO queues

Monitor FIFO queues

This topic describes how to configure a FIFO input using `inputs.conf`. Defining FIFO inputs is not currently supported in Splunk Web/Manager.

Caution: Data sent via FIFO is not persisted in memory and can be an unreliable method for data sources. To ensure your data is not lost, use monitor instead.

Add a FIFO input to inputs.conf

To add a FIFO input, add a stanza for it to inputs.conf in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. If you have not worked with Splunk's configuration files before, read about configuration files before you begin.

You can set any number of attributes and values following an input type. If you do not specify a value for one or more attributes, Splunk uses the defaults that are preset in `$SPLUNK_HOME/etc/system/default/` (noted below).

`[fifo://<path>]`

This input stanza type directs Splunk to read from a FIFO at the specified path.

`host = <string>`

- Set the host value of your input to a static value.
- `host=` is automatically prepended to the value when this shortcut is used.
- Defaults to the IP address of fully qualified domain name of the host where the data originated.
- For more information about the host field, read "About default fields" in this manual.

`index = <string>`

- Set the index where events from this input will be stored.
- `index=` is automatically prepended to the value when this shortcut is used.
- Defaults to `main` (or whatever you have set as your default index).
- For more information about the index field, read "About fields" in the Knowledge Manager Manual.

`sourcetype = <string>`

- Set the sourcetype name of events from this input.
- `sourcetype=` is automatically prepended to the value when this shortcut is used.
- Splunk automatically picks a source type based on various aspects of your data. There is no hard-coded default.
- For more information about the sourcetype field, read"About default fields" in this manual.

`source = <string>`

- Set the source name of events from this input.
- Defaults to the file path.
- `source=` is automatically prepended to the value when this shortcut is used.

`queue = <string> (parsingQueue, indexQueue, etc)`

- Specify where the input processor should deposit the events that it reads.
- Can be any valid, existing queue in the pipeline.

- Defaults to `parsingQueue`.

Monitor changes to your filesystem

Monitor changes to your filesystem

Splunk's **file system change monitor** is useful for tracking changes in your file system. The file system change monitor watches any directory you specify and generates an event (in Splunk) when that directory undergoes any change. It is completely configurable and can detect when any file on the system is edited, deleted or added (not just Splunk-specific files). For example, you can tell the file system change monitor to watch `/etc/sysconfig/` and alert you any time the system's configurations are changed.

Configure the file system change monitor in inputs.conf.

Note: If you're interested in auditing file reads on Windows, check out this topic on the Splunk Community best practices Wiki. Some users might find it more straightforward to use Windows native auditing tools.

How the file system change monitor works

The file system change monitor detects changes using:

- modification date/time
- group ID
- user ID
- file mode (read/write attributes, etc.)
- optional SHA256 hash of file contents

You can configure the following features of the file system change monitor:

- white listing using regular expressions
    - ♦ specify files that will be checked no matter what
- black listing using regular expressions
    - ♦ specify files to skip
- directory recursion
    - ♦ including symbolic link traversal
    - ♦ scanning multiple directories, each with their own polling frequency
- cryptographic signing
    - ♦ creates a distributed audit-trail of file system changes
- indexing entire file as an event on add/change
    - ♦ size cutoffs for sending entire file and/or hashing
- all change events indexed by and searchable through Splunk

Caution: Do not configure the file system change monitor to monitor your root filesystem. This can be dangerous and time-consuming if directory recursion is enabled.

Configure the file system change monitor

By default, the file system change monitor will generate **audit events** whenever the contents of `$SPLUNK_HOME/etc/` are changed, deleted, or added to. When you start Splunk for the first time, an `add` audit event will be generated for each file in the `$SPLUNK_HOME/etc/` directory and all sub-directories. Any time after that, any change in configuration (regardless of origin) will generate an audit event for the affected file(s). If you have `signedaudit=true` , the file system change audit event will be indexed into the **audit index** (`index=_audit`). If signedaudit is not turned on, by default, the events are written to the **main** index unless you specify another index.

Note: The file system change monitor does not track the user name of the account executing the change, only that a change has occurred. For user-level monitoring consider using native operating system audit tools, which have access to this information.

You can use the file system change monitor to watch any directory by adding a stanza to `inputs.conf`.

Create your own `inputs.conf` in `$SPLUNK_HOME/etc/system/local/`. Edit this files in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For more information on configuration files in general, see "About configuration files".

Edit the `[fschange]` stanza to configure the file system change monitor. Every setting is optional except the stanza name `fschange:<directory or file to monitor>`.

Note: You must restart Splunk any time you make changes to the `[fschange]` stanza.

```
[fschange:<directory or file to monitor>]
index=<indexname>
recurse=<true | false>
followLinks=<true | false>
pollPeriod=N
hashMaxSize=N
fullEvent=<true | false>
sendEventMaxSize=N
signedaudit=<true | false>
filters=<filter1>,<filter2>,...<filterN>
```

Possible attribute/value pairs

`[fschange:<directory or file to monitor>]`

- The system will monitor all adds/updates/deletes to this directory and sub-directories.
- Any changes will generate an event that is indexed by Splunk.
- Defaults to `$SPLUNK_HOME/etc/`.

`index=<indexname>`

- The index to store all events generated.
- Defaults to main (unless you have turned on audit event signing).

`recurse=<true | false>`

- If true, recurse directories within the directory specified in `[fschange]`.
- Defaults to true.

`followLinks=<true | false>`

- If true, the file system change monitor will follow symbolic links.
- Defaults to false.

Caution: If you are not careful with setting `followLinks`, file system loops may occur.

`pollPeriod=N`

- Check this directory for changes every N seconds.
- Defaults to 3600.
    - ♦ If you make a change, the file system audit events could take anywhere between 1 and 3600 seconds to be generated and become available in audit search.

`hashMaxSize=N`

- Calculate a SHA1 hash for every file that is less than or equal to N size in bytes.
- This hash can be used as an additional method for detecting change in the file/directory.
- Defaults to -1 (no hashing used for change detection).

`signedaudit=<true | false>`

- Send cryptographically signed add/update/delete events.
- Defaults to false.
- Setting to true will generate events in the _audit index.
- This should be deliberately set to false if you wish to set the index.

Note: When setting `signedaudit` to true, make sure auditing is enabled in `audit.conf`.

`fullEvent=<true | false>`

- Send the full event if an add or update change is detected.
- Further qualified by the `sendEventMaxSize` attribute.
- Defaults to false.

`sendEventMaxSize=N`

- Only send the full event if the size of the event is less than or equal to N bytes.
- This limits the size of indexed file data.
- Defaults to -1, which is unlimited.

`sourcetype = <string>`

- Set the sourcetype for events from this input.

- "sourcetype=" is automatically prepended to <string>.
- sourcetype = fs_notification by default.

`filesPerDelay = <integer>`

- Injects a delay specified by 'delayInMills' after processing <integer> files.
- This is used to throttle file system monitoring so it doesn't consume as much CPU.

`delayInMills = <integer>`

- The delay in milliseconds to use after processing every <integer> files as specified in 'filesPerDelay'.
- This is used to throttle file system monitoring so it doesn't consume as much CPU.

`filters=<filter1>,<filter2>,...<filterN>`

Each of these filters will apply from left to right for each file or directory that is found during the monitors poll cycle.

To define a filter, add a `[filter...]` stanza as follows:

```
[filter:blacklist:backups]
regex1 = .*bak
regex2 = .*bk
[filter:whitelist:code]
regex1 = .*\.c
regex2 = .*\.h

[fschange:/etc]
filters = backups,code
```

Fschange white/blacklist logic is handed similarly to typical firewalls. The events run down through the list of filters until they reach their first match. If the first filter to match an event is a whitelist, the event will be indexed. If the first filter to match an event is a blacklist, the event will not be indexed. If an event reaches the end of the chain with no matches, it will be indexed. This means that there is an implicit "all pass" built in. To default to a situation where events are *not* indexed if they don't match a whitelist explicitly, end the chain with a blacklist that will match all remaining events.

For example:

```
...
filters = <filter1>, <filter2>, ... terminal-blacklist

[filter:blacklist:terminal-blacklist]
regex1 = .?
```

Important: If a directory is ever blacklisted *including via a terminal blacklist at the end of a series of whitelists*, then *all* its subfolders and files are automatically blacklisted and will not pass any whitelist. To accommodate this, whitelist all desired folders and subfolders explicitly ahead of the blacklist items in your filters.

Examples

Monitor files with specific extensions

This configuration monitors files in the specified directory with the extensions config, xml, properties, and log and ignores all others.

Note: In this example, a directory could be blacklisted. If this is the case, *all* its subfolders and files would automatically blacklisted as well--only files in the specified directory would be monitored.

```
[filter:whitelist:configs]
regex1 = .*\.config
regex2 = .*\.xml
regex3 = .*\.properties
regex4 = .*\.log

[filter:blacklist:terminal-blacklist]
regex1 = .?

[fschange:/var/apache]
index = sample
recurse = true
followLinks = false
signedaudit = false
fullEvent = true
sendEventMaxSize = 1048576
delayInMills = 1000
filters = configs,terminal-blacklist
```

Monitor the Windows system directory for changes

This Windows-specific example monitors the Windows system directory for changes:

```
[filter:whitelist:binaries]
regex1 = .*\.exe
regex2 = .*\.dll
regex3 = .*\.msc
regex4 = .*\.ocx

[filter:whitelist:dirstoindex]
regex1 = .*catroot*
regex2 = .*dllcache
regex3 = .*dns
regex4 = .*grouppolicy
regex5 = .*inetsrv
regex6 = .*logfiles
regex7 = .*ras
regex8 = .*spool
regex9 = .*wins

[filter:blacklist:terminal-blacklist]
regex1 = .?

[fschange:C:\Windows\System32]
index = sample
recurse = true
followLinks = false
```

```
signedaudit = false
fullEvent = true
sendEventMaxSize = 1048576
delayInMills = 1000
filters = binaries,dirstoindex,terminal-blacklist
```

Find more things to monitor with crawl

Find more things to monitor with crawl

Use crawl to search your filesystem for new data sources to add to your index. Configure one or more types of crawlers in crawl.conf to define the type of data sources to include in or exclude from your results.

Configuration

Edit `$SPLUNK_HOME/etc/system/local/crawl.conf` to configure one or more crawlers that browse your data sources when you run the `crawl` command. Define each crawler by specifying values for each of the crawl attributes. Enable the crawler by adding it to `crawlers_list`.

Crawl logging

The `crawl` command produces a log of crawl activity that's stored in `$SPLUNK_HOME/var/log/splunk/crawl.log`. Set the logging level with the `logging` key in the [default] stanza of `crawl.conf`:

```
[default]
logging = <warn | error | info | debug>
```

Enable crawlers

Enable a crawler by listing the crawler specification stanza name in the `crawlers_list` key of the `[crawlers]` stanza.

Use a comma-separated list to specify multiple crawlers.

Enable crawlers that are defined in the stanzas: `[file_crawler]`, `[port_crawler]`, and `[db_crawler]`.

```
[crawlers]
crawlers_list = file_crawler, port_crawler, db_crawler
```

Define crawlers

Define a crawler by adding a definition stanza in `crawl.conf`. Add additional crawler definitions by adding additional stanzas.

Example crawler stanzas in `crawl.conf`:

```
[Example_crawler_name]
....
[Another_crawler_name]
```

....

Add key/value pairs to crawler definition stanzas to set a crawler's behavior. The following keys are available for defining a `file_crawler`:

Argument	Description
`bad_directories_list`	Specify directories to exclude.
`bad_extensions_list`	Specify file extensions to exclude.
`bad_file_matches_list`	Specify a string, or a comma-separated list of strings that filenames must contain to be excluded. You can use wildcards (examples: foo*.*,foo*bar, *baz*).
`packed_extensions_list`	Specify extensions of common archive filetypes to include. Splunk unpacks compressed files before it reads them. It can handle tar, gz, bz2, tar.gz, tgz, tbz, tbz2, zip, and z files. Leave this empty if you don't want to add any archive filetypes.
`collapse_threshold`	Specify the minimum number of files a source must have to be considered a directory.
`days_sizek_pairs_list`	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.
`big_dir_filecount`	Set the maximum number of files a directory can have in order to be crawled. crawl excludes directories that contain more than the maximum number you specify.
`index`	Specify the name of the index to which you want to add crawled file and directory contents.
`max_badfiles_per_dir`	Specify how far to crawl into a directory for files. If Splunk crawls a directory and doesn't find valid files within the specified max_badfiles_per_dir, then Splunk excludes the directory.
`root`	Specify directories for a crawler to crawl through.

Example

Here's an example crawler called `simple_file_crawler`:

```
[simple_file_crawler]
bad_directories_list= bin, sbin, boot, mnt, proc, tmp, temp, home, mail, .thumbnails, cache, ol
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
```

Send SNMP events to Splunk

Send SNMP events to Splunk

This topic covers ways to receive and index SNMP traps at the Splunk indexer. SNMP traps are alerts fired off by remote devices; these devices need to be configured to send their traps to Splunk's IP address. The default port for SNMP traps is udp:162. This topic does not cover SNMP polling, which is a way to query remote devices.

On UNIX

The most effective way to index SNMP traps is to use `snmptrapd` to write them to a file. Then, configure the Splunk server to add the file as an input.

snmptrapd itself is part of the net-snmp project. If you're installing this on your system, refer first to any local documentation for your distribution's packaging of the tool, and after that, the documentation here: http://net-snmp.sourceforge.net/docs/man/snmptrapd.html

The simplest configuration is:

```
# snmptrapd -Lf /var/log/snmp-traps
```

Note: Previously, snmptrapd would accept all incoming notifications, and log them automatically (even if no explicit configuration was provided). Starting with snmptrapd release 5.3 (check with snmptrapd --version), access control checks will be applied to all incoming notifications. If snmptrapd is run without suitable access control settings, then such traps WILL NOT be processed. You can avoid this by specifying:

```
    # snmptrapd -Lf /var/log/snmp-traps --disableAuthorization=yes
```

Troubleshooting:

- If you keep the default listening port of 161, which is a privileged port, you will have to run snmptrapd as root.
- Use the -f flag to keep snmptrapd in the foreground while testing. Use -Lo instead of -Lf to log to standard output
- You can use the snmptrap command to generate an example trap, as in: # snmptrap -v2c -c public localhost 1 1

On Windows

To log SNMP traps to a file on Windows:

1. Install NET-SNMP from http://www.net-snmp.org/

2. Register snmptrapd as service using the script included in the NET-SNMP install.

3. Edit C:\usr\etc\snmp\snmptrapd.conf

```
snmpTrapdAddr [System IP]:162
```

```
authCommunity log [community string]
```

4. The default log location is C:\usr\log\snmptrapd.log

MIBs

MIBs, or *Management Information Bases*, provide a map between numeric OIDs reported by the SNMP trap and a textual human readable form. Though snmptrapd will work quite happily without any MIB files at all, the results won't be displayed in quite the same way. The vendor of the device you are receiving traps from should provide a specific MIB. For example, all Cisco device MIBs can be located using the online Cisco SNMP Object Navigator

There are two steps required to add a new MIB file:

1. Download and copy the MIB file into the MIB search directory. The default location is /usr/local/share/snmp/mibs, although this can be set using the -M flag to snmptrapd.

2. Instruct snmptrapd to load the MIB or MIBs by passing a colon separated list to the -m flag. There are two important details here:

- Adding a leading '+' character will load the MIB in addition to the default list, instead of overwriting the list; and
- The special keyword ALL is used to load all MIB modules in the MIB directory. The safest argument seems to be -m +ALL

Set up custom (scripted) inputs

Set up custom (scripted) inputs

Splunk can accept events from scripts that you provide. Scripted input is useful in conjunction with command-line tools, such as vmstat, iostat, netstat, top, etc. You can use scripted input to get data from APIs and other remote data interfaces and message queues. You can then use that data to generate metrics and status data through commands like vmstat, iostat, etc.

Lots of apps on Splunkbase provide scripted inputs for specific applications. -- You can find them on the **Browse more apps** tab in the **Launcher**.

You configure custom scripted inputs from Splunk Manager or by editing inputs.conf.

Note: On Windows platforms, you can enable text-based scripts, such those in perl and python, with an intermediary Windows batch (.bat) file.

Caution: Scripts launched through scripted input inherit Splunk's environment, so be sure to clear environment variables that can affect your script's operation. The only environment variable that's likely to cause problems is the library path (most commonly known as `LD_LIBRARY_PATH` on linux/solaris/freebsd).

Add a scripted input in Splunk Web

To add a scripted input in Splunk Web:

1. Click **Manager** in the upper right-hand corner of Splunk Web.

2. Under System configurations, click **Data Inputs**.

3. Click **Scripts**.

4. Click **Add new** to add an input.

5. In the **Command** text box, specify the script command, including the path to the script.

6. In **Interval**, specify the interval in seconds between script runtimes. The default is 60 (seconds).

7. Enter a new **Source name** to override the default source value, if necessary.

Important: Consult Splunk support before changing this value.

8. Change the **Host** value, if necessary.

9. Set the **Source type**.

Source type is a default field added to events. Source type is used to determine processing characteristics such as timestamps and event boundaries. Choose:

- **From List.** Select one of the predefined source types from the drop-down list.
- **Manual.** Label your own source type in the text box.

10. Set the **Index**. Leave the value as "default" unless you have defined multiple indexes to handle different types of events. In addition to indexes meant for user data, Splunk has a number of utility indexes, which show up in the dropdown box.

11. Click **Save**.

Add a scripted input via inputs.conf

Configure `inputs.conf` using the following attributes:

```
[script://$SCRIPT]
interval = <integer>|<cron schedule>
index = <index>
sourcetype = <iostat, vmstat, etc>  OPTIONAL
source = <iostat, vmstat, etc> OPTIONAL
disabled = <true | false>
```

- `script` is the fully-qualified path to the location of the script.
  - As a best practice, put your script in the `bin/` directory nearest the `inputs.conf` where your script is specified. So if you are configuring `$SPLUNK_HOME/etc/system/local/inputs.conf`, place your script in

`$SPLUNK_HOME/etc/system/bin/`. If you're working on an application in `$SPLUNK_HOME/etc/apps/$APPLICATION/`, put your script in `$SPLUNK_HOME/etc/apps/$APPLICATION/bin/`.

- `interval` indicates how often to execute the specified command. Specify either an integer value representing seconds or a valid cron schedule.
  - Defaults to 60 seconds.
  - When a cron schedule is specified, the script is not executed on start up.
  - Splunk keeps one invocation of a script per instance. Intervals are based on when the script completes. So if you have a script configured to run every 10 minutes and the script takes 20 minutes to complete, the next run will occur 30 minutes after the first run.
  - For constant data streams, enter 1 (or a value smaller than the script's interval).
  - For one-shot data streams, enter -1. Setting `interval` to -1 will cause the script to run each time the splunk daemon restarts.
- `index` can be any index in your Splunk instance.
  - Default is `main`.
- `disabled` is a boolean value that can be set to true if you want to disable the input.
  - Defaults to `false`.
- `sourcetype` and `source` can be any value you'd like.
  - The value you specify is appended to data coming from your script in the `sourcetype=` or `source=` fields.
  - These are optional settings.

If you want the script to run continuously, write the script to never exit and set it on a short interval. This helps to ensure that if there is a problem the script gets restarted. Splunk keeps track of scripts it has spawned and will shut them down upon exit.

Example using inputs.conf

This example shows the use of the UNIX `top` command as a data input source.

- Start by creating a new application directory. This example uses `scripts/`:

```
$ mkdir $SPLUNK_HOME/etc/apps/scripts
```

- All scripts should be run out of a `bin/` directory inside your application directory:
- `$ mkdir $SPLUNK_HOME/etc/apps/scripts/bin`
- This example uses a small shell script `top.sh`:

```
$ #!/bin/sh
 top -bn 1  # linux only - different OSes have different paramaters
```

- Make sure the script is executable:

```
chmod +x $SPLUNK_HOME/etc/apps/scripts/bin/top.sh
```

- Test that the script works by running it via the shell:

```
$SPLUNK_HOME/etc/apps/scripts/bin/top.sh
```

- The script should have sent one `top` output.

- Add the script entry to `inputs.conf` in `$SPLUNK_HOME/etc/apps/scripts/default/`:

```
[script:///opt/splunk/etc/apps/scripts/bin/top.sh]
interval = 5                # run every 5 seconds
sourcetype = top        # set sourcetype to top
source = script://./bin/top.sh   # set source to name of script
```

props.conf

You may need to modify props.conf:

- By default Splunk breaks the single `top` entry into multiple events.
- The easiest way to fix this problem is to tell the Splunk server to break only before something that does not exist in the output.

For example, adding the following to `$SPLUNK_HOME/etc/apps/scripts/default/props.conf` forces all lines into a single event:

```
[top]
BREAK_ONLY_BEFORE = <stuff>
```

Since there is no timestamp in the top output we need to tell Splunk to use the current time. This is done in `props.conf` by setting:

```
DATETIME_CONFIG = CURRENT
```

Whitelist or blacklist specific incoming data

Whitelist or blacklist specific incoming data

Use **whitelist** and **blacklist** rules to explicitly tell Splunk which files to consume when **monitoring** directories. You can also apply these settings to `batch` inputs. When you define a whitelist, Splunk indexes ONLY the files in that list. Alternately, when you define a blacklist, Splunk ignores the files in that list and consumes everything else. You don't have to define both a whitelist and a blacklist, they are independent settings. If you happen to have both, and a file that matches both of them, that file WILL NOT be indexed, for example `blacklist` will override `whitelist`.

Whitelist and blacklist rules use regular expression syntax to define the match on the file name/path. Also, your rules must be contained within a configuration stanza, for example `[monitor://<path>]`; those outside a stanza (global entries) are ignored.

Instead of whitelisting or blacklisting your data inputs, you can filter specific events and send them to different queues or indexes. Read more about routing and filtering data. You can also use the crawl feature to predefine files you want Splunk to index or not index automatically when they are added to your filesystem.

Define whitelist and blacklist entries with exact regex syntax; the "..." wildcard is not supported.

Whitelist (allow) files

To define the files you want Splunk to exclusively index, add the following line to your `monitor` stanza in the `/local/inputs.conf` file **for the App this input was defined in**:

```
whitelist = $YOUR_CUSTOM_REGEX
```

For example, if you want Splunk to monitor only files with the `.log` extension:

```
[monitor:///mnt/logs]
    whitelist = \.log$
```

You can whitelist multiple files in one line, using the "|" (OR) operator. For example, to whitelist filenames that contain `query.log OR my.log`:

```
whitelist = query\.log$|my\.log$
```

Or, to whitelist exact matches:

```
whitelist = /query\.log$|/my\.log$
```

Note: The "$" anchors the regex to the end of the line. There is no space before or after the "|" operator.

Blacklist (ignore) files

To define the files you want Splunk to exclude from indexing, add the following line to your `monitor` stanza in the `/local/inputs.conf` file **for the App this input was defined in**:

```
blacklist = $YOUR_CUSTOM_REGEX
```

Important: If you create a `blacklist` line for each file you want to ignore, Splunk activates only the last filter.

If you want Splunk to ignore and not monitor only files with the `.txt` extension:

```
[monitor:///mnt/logs]
    blacklist = \.(txt)$
```

If you want Splunk to ignore and not monitor all files with either the `.txt` extension `OR` the `.gz` extension (note that you use the "|" for this):

```
[monitor:///mnt/logs]
    blacklist = \.(txt|gz)$
```

If you want Splunk to ignore entire directories beneath a monitor input refer to this example:

```
[monitor:///mnt/logs]
    blacklist = (archive|historical|\.bak$)
```

The above example tells Splunk to ignore all files under /mnt/logs/ within the archive directory, within historical directory and to ignore all files ending in *.bak.

If you want Splunk to ignore files that contain a specific string you could do something like this:

```
[monitor:///mnt/logs]
   blacklist = 2009022[89]file\.txt$
```

The above example will ignore the webserver20090228file.txt and webserver20090229file.txt files under /mnt/logs/.

How log file rotation is handled

How log file rotation is handled

Splunk recognizes when a file that it is monitoring (such as `/var/log/messages`) has been rolled (`/var/log/messages1`) and will not read the rolled file in a second time.

Note: Splunk does not recognize compressed files produced by logrotate (such as bz2 or gz) as the same as the uncompressed originals. This can lead to a duplication of data if these files are then monitored by Splunk. You can configure logrotate to move these files into a directory you have not told Splunk to read, or you can explicitly set **blacklist** rules for archive filetypes to prevent Splunk from reading these files as new logfiles.

Example:

```
blacklist = \.(gz|bz2|z|zip)$
```

Splunk recognizes the following archive filetypes: tar, gz, bz2, tar.gz, tgz, tbz, tbz2, zip, and z.

For more information on setting blacklist rules see "Whitelist and blacklist specific incoming data" in this manual.

How log rotation works

The monitoring processor picks up new files and reads the first and last 256 bytes of the file. This data is hashed into a begin and end cyclic redundancy check (CRC). Splunk checks new CRCs against a database that contains all the CRCs of files Splunk has seen before. The location Splunk last read in the file is also stored.

There are three possible outcomes of a CRC check:

1. There is no begin and end CRC matching this file in the database. This is a new file and will be picked up and consumed from the start. Splunk updates the database with new CRCs and seekptrs as the file is being consumed.

2. The begin CRC is present and the end CRC are present but the size of the file is larger than the seekPtr Splunk stored. This means that, while Splunk has seen the file before, there has been information added to it since it was last read. Splunk opens the file and seeks to the previous end of the file and starts reading from there (so Splunk will only grab the new data and not anything it has read before).

3. The begin CRC is present but the end CRC does not match. This means the file has been changed since Splunk last read it and some of the portions it has read in already are different. In this case there is evidence that the previous data Splunk read from has been changed. In this case Splunk has no choice but to read the whole file again.

Set up forwarding and receiving

About forwarding and receiving

About forwarding and receiving

You can forward data from a Splunk instance to another Splunk server or even to a non-Splunk system. The Splunk instance that performs the **forwarding** is typically a smaller footprint version of Splunk, called a **forwarder**. The forwarder functions as a lightweight, all-purpose agent.

A Splunk server that **receives** data from a forwarder is called a **receiver**. The receiver is either a Splunk **indexer** or another forwarder, configured to receive data from one or more forwarders.

The forwarding and receiving capability makes possible all sorts of interesting Splunk topologies to handle functions like data consolidation, **load balancing**, **data cloning**, and **data routing**.

Forwarders vs. light forwarders

Splunk forwarders come in two flavors: regular and light. These differ according to their functionality and the corresponding size of their footprints.

A **regular forwarder**, also referred to as just a **forwarder**, has a smaller footprint than a Splunk server but retains most of the capability, except that it lacks the ability to do distributed searches. Much of its default functionality, such as Splunk Web, can be disabled, if necessary, to further reduce the size of its footprint. A forwarder parses data before forwarding it and can route data based on criteria such as source or type of event.

A **light forwarder** has a small footprint with limited functionality. Its size makes it ideal for forwarding data from workstations or non-Splunk production servers to a Splunk server for consolidation. It forwards only unparsed data and, therefore, cannot perform content-based routing. In addition, it does not include Splunk Web and its throughput is limited to 256kbs.

For detailed information on the capabilities of regular and light forwarders, see More about forwarders in this manual.

Both types of forwarders can perform automatic load balancing, with the regular forwarder also offering round-robin load balancing. Forwarders represent a much more robust solution for data forwarding than raw network feeds, with their capabilities for:

- Tagging of metadata (source, sourcetype, and host)
- Configurable buffering
- Data compression
- SSL security
- Use of any available network ports

Types of data

Forwarders can transmit three types of data:

- Raw
- Unparsed
- Parsed

A light forwarder can send raw or unparsed data. A regular forwarder can send raw or parsed data.

With raw data, the data stream is forwarded as raw TCP; it is not converted into Splunk's communications format. The forwarder just collects the data and forwards it on. This is particularly useful for sending data to a non-Splunk system.

With unparsed data, a light forwarder performs only minimal processing. It does not examine the data stream, but it does tag the entire stream with metadata to identify source, sourcetype, and host. It also divides the data stream into 32K blocks and performs some rudimentary timestamping on the stream, for use by the receiving indexer in case the events themselves have no discernable timestamps. The light forwarder does not identify, examine, or tag individual events.

With parsed data, a regular forwarder breaks the data into individual events, which it tags and then forwards to a Splunk server. It can also examine the events. Because the data has been parsed, the forwarder can perform conditional routing based on event data, such as field values.

The parsed and unparsed formats are both referred to as *cooked* data, to distinguish them from raw data. By default, forwarders send cooked data — in the light forwarder's case, unparsed data, and in the regular forwarder's case, parsed data. To send raw data instead, set the `sendCookedData=false` atribute/value pair in outputs.conf.

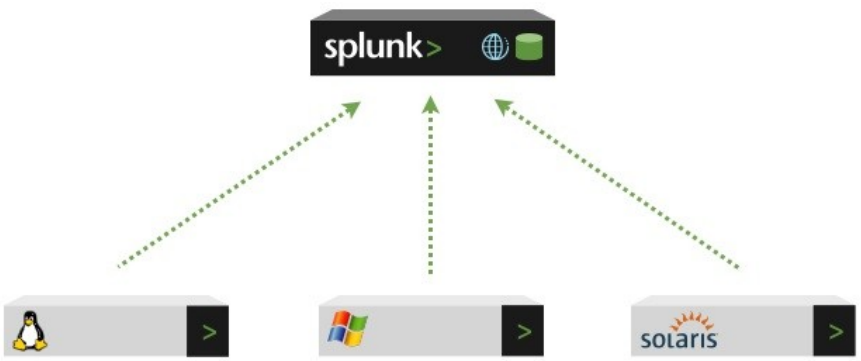
Deployment topologies

You can deploy Splunk forwarders in a wide variety of scenarios. These are some typical topologies.

Data consolidation

Data consolidation represents one of the most common topologies, with multiple forwarders sending data to a single Splunk server. The scenario typically involves light forwarders forwarding unparsed data from workstations or production non-Spunk servers to a central Splunk server for consolidation and indexing. With a lighter footprint, these forwarders have minimal impact on the performance of the systems they reside on. In other scenarios, regular forwarders send parsed data to a central Splunk server.

Here, three light forwarders are sending data to a single Splunk server:



Load balancing

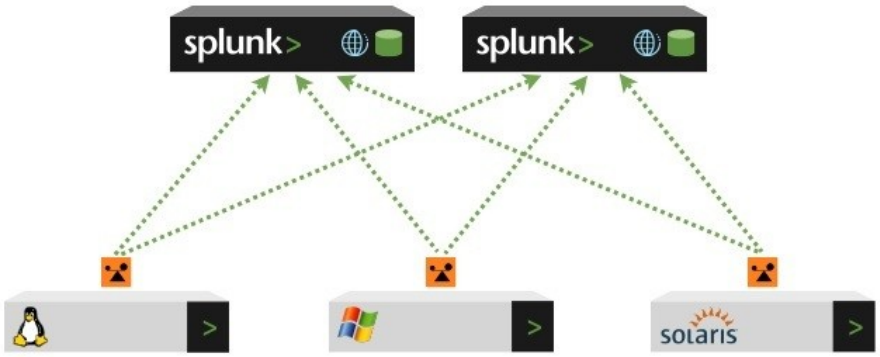
Load balancing simplifies the process of distributing data across several Splunk servers to handle considerations such as high data volume, horizontal scaling for enhanced search performance, and fault tolerance. In load balancing, the forwarder routes data sequentially to different servers at specified intervals.

Splunk load balancing comes in two flavors:

- Automatic load balancing
- Round-robin load balancing

For most needs, automatic load balancing, in which the forwarder switches receivers at set time intervals, offers the better solution. For details on the relative advantages of automatic vs. round-robin load balancing, see Load balancing in this manual.

In this diagram, three light forwarders are each performing automatic load balancing between two receivers:



Routing and filtering

In routing, a forwarder routes events to specific Splunk or third-party servers, based on criteria such as source, sourcetype, or patterns in the events themselves. Routing at the event level requires a regular forwarder.

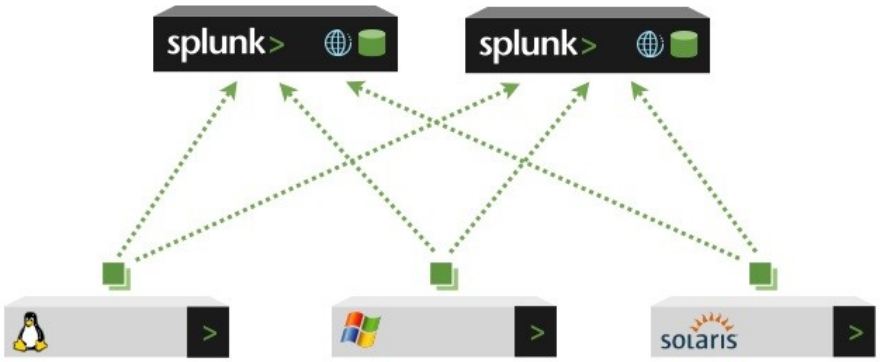
A forwarder can also filter and route events to specific queues, or discard them altogether by routing to the null queue.

Here, a regular forwarder routes data to three Splunk servers based on event patterns:

Cloning

With cloning, the forwarder sends duplicate copies of data to multiple Splunk servers. If cloning is combined with load balancing, the forwarder sends duplicate copies of data to multiple *groups* of servers. This second scenario is particularly useful for situations requiring data redundancy to promote data availability. If any of the servers in a load-balanced group goes down while receiving data, another server in the group automatically takes over the receiver function, ensuring that each group of servers still contains a clone of the data.

In this simple scenario, three forwarders are sending duplicate copies of data to two Splunk servers:



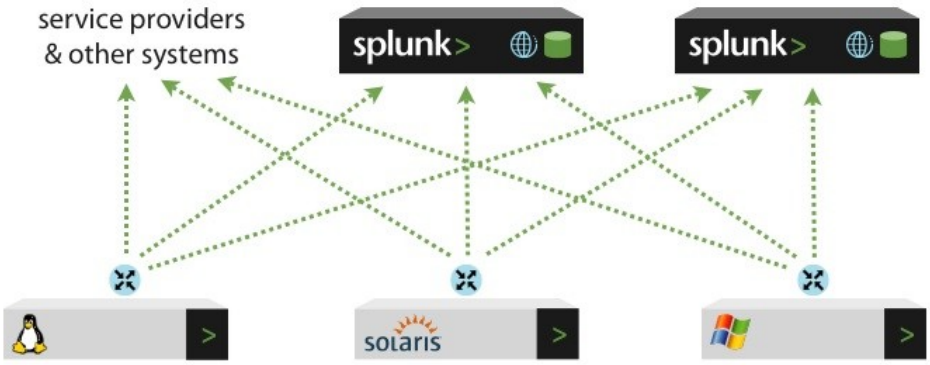
You can also clone data from one full Splunk instance to another. This scenario can be useful if you need to keep complete sets of indexed data at separate locations to ensure fast local access; for instance, the North American office in San Francisco and the European office in London.

In another type of cloning, a regular forwarder can retain indexed data locally while also forwarding parsed data to a Splunk server.

Forwarding to non-Splunk systems

You can send raw data to a third-party system such as a syslog aggregator. You can combine this with data routing, sending some data to a non-Splunk system and other data to one or more Splunk servers.

Here, three forwarders are routing data to two Splunk servers and a non-Splunk system:



Key set-up steps

Once you've determined your Splunk deployment topology and what sort of data forwarding is necessary to implement it, the steps for setting up forwarding are simple:

1. Install the Splunk instances that will serve as forwarders and receivers. See the *Installation Manual* for details.

2. Use Splunk Web or the CLI to enable receiving on the instances designated as receivers. See Set up receiving in this manual.

3. Use Splunk Web or the CLI to enable forwarding on the instances designated as forwarders. See Set up forwarding in this manual.

4. Specify data inputs for the Splunk forwarders in the usual manner. See Add data and configure inputs in this manual.

5. Perform any advanced configuration on each forwarder by editing the outputs.conf file. See Configure forwarders with outputs.conf in this manual.

6. Test the results to confirm that forwarding, along with any configured behaviors like load balancing or routing, is occurring as expected.

In large environments with muliple forwarders, you might find it helpful to use the deployment server to manage your forwarders. See Deploy to other Splunk instances in this manual.

Enable forwarding and receiving

Enable forwarding and receiving

To enable forwarding and receiving, you configure both a receiver and a forwarder. The receiver is the Splunk instance receiving the data; the forwarder is the Splunk instance forwarding the data. Depending on your needs, you might have multiple receivers or forwarders.

You must first set up the receiver. You can then set up forwarder(s) to send data to that receiver.

Important: The receiver must be running the same (or later) version of Splunk as its forwarder. A 4.0 receiver can receive data from a 3.4 forwarder, but a 3.4 receiver cannot receive from a 4.0 forwarder.

Set up receiving

You enable receiving in Splunk Web or through the Splunk CLI.

Set up receiving with Splunk Web

Use the Manager interface to set up a receiver:

1. Log into Splunk Web as admin on the server that will be receiving data from a forwarder.

2. Click the **Manager** link in the upper right corner.

3. Select **Forwarding and receiving** under **System configurations.**

4. Click **Add new** in the **Receive data** section.

5. Specify which TCP port you want the receiver to listen on. For example, if you enter "9997," the receiver will receive data on port 9997. By convention, receivers listen on port 9997, but you can specify any unused port. You can use a tool like `netstat` to determine what ports are available on your system. Make sure the port you select is not in use by splunkweb or splunkd.

6. Click **Save.** You must restart Splunk to complete the process.

Set up receiving with Splunk CLI

To access the CLI, first navigate to `$SPLUNK_HOME/bin/`. This is unnecessary if you have added Splunk to your path.

To enable receiving, enter:

```
./splunk enable listen <port> -auth <username>:<password>
```

For `<port>`, substitute the port you want the receiver to listen on.

To disable receiving, enter:

```
./splunk disable listen -port <port> -auth <username>:<password>
```

Searching data received from a forwarder running on a different operating system

In most cases, a Splunk instance receiving data from a forwarder on a different OS will need to install the app for that OS. However, there are numerous subtleties that affect this; read on for the details.

Forwarding and indexing are OS-independent operations. Splunk supports any combination of forwarders and receivers, as long as each is running on a certified OS. For example, a Linux receiver can index data from a Windows forwarder.

Once data has been forwarded and indexed, the next step is to search or perform other knowledge-based activities on the data. At this point, the Splunk instance performing such activities might need information about the OS whose data it is examining. You typically handle this by installing the app specific to that OS. For example, if you want a Linux instance to search OS-specific data forwarded from Windows, you will ordinarily want to install the Windows app on the Linux instance.

If the data you're interested in is not OS-specific, such as web logs, then you do not need to install the Splunk OS app.

In addition, if the receiver is only indexing the data, and an external search head is performing the actual searches, you do not need to install the OS app on the receiver, but you might need to install it on the search head. As an alternative, you can use a search head running the OS. For example, to search data forwarded from Windows to a Linux receiver, you can use a Windows search head pointing to the Linux indexer as a remote search peer. For more information on search heads, see "Set up distributed search".

Important: After you have downloaded the relevant OS app, remove its `inputs.conf` file before enabling it, to ensure that its default inputs are not added to your indexer. For the Windows app, the location is: `%SPLUNK_HOME%\etc\apps\windows\default\inputs.conf`.

In summary, you only need to install the app for the forwarder's OS on the receiver (or search head) if it will be performing searches on the forwarded OS data.

Set up forwarding

You can use Splunk Web or the Splunk CLI as a quick way to enable forwarding in a Splunk instance.

You can also enable, as well as configure, forwarding by creating an `outputs.conf` file for the Splunk instance. Although setting up forwarders with `outputs.conf` requires a bit more initial knowledge, there are obvious advantages to performing all forwarder configurations in a single location. Most advanced configuration options are available only through `outputs.conf`. In addition, if you will be enabling and configuring a number of forwarders, you can easily accomplish this by editing a single `outputs.conf` file and making a copy for each forwarder. See the topic Configure fowarders with outputs.conf for more information.

Note: By default, Splunk uses an Enterprise trial license when it is initially installed. When you enable a forwarder, you should also apply either the forwarder license or the free license to avoid any subsequent license issues. Instructions on how to do this can be found here.

Set up regular forwarding with Splunk Web

Use the Manager interface to set up a forwarder. To set up a regular forwarder:

1. Log into Splunk Web as admin on the server that will be forwarding data.

2. Click the **Manager** link in the upper right corner.

3. Select **Forwarding and receiving** under **System configurations.**

4. Click **Add new** in the **Forward data** section.

5. Enter the hostname or IP address for the receiving Splunk instance, along with the port specified when the receiver was configured. For example, you might enter: `receivingserver.com:9997`.

6. Click **Save.** You must restart Splunk to complete the process.

You can use Splunk Web to perform one other configuration (for regular forwarders only). To store a copy of indexed data local to the forwarder:

1. From **Forwarding and receiving,** select **Forwarding defaults.**

2. Select **Yes** to store and maintain a local copy of the indexed data on the forwarder.

All other configuration must be done in `outputs.conf`.

Set up light forwarding with Splunk Web

To enable light forwarding, you must first enable regular forwarding on the Splunk instance. Then you separately enable light fowarding. This procedure combines the two processes:

1. Log into Splunk Web as admin on the server that will be forwarding data.

2. Click the **Manager** link in the upper right corner.

3. Select **Forwarding and receiving** under **System configurations.**

4. Click **Add new** in the **Forward data** section.

5. Enter the hostname or IP address for the receiving Splunk instance, along with the port specified when the receiver was configured. For example, you might enter: `receivingserver.com:9997`.

6. Click **Save.**

7. Return to **Manager>>Forwarding and receiving**.

8. Click **Enable lightweight forwarding** in the **Forward data** section. You must restart Splunk to complete the process.

Important: When you enable a light forwarder, Splunk Web is immediately disabled. You will then need to use the Splunk CLI or `outputs.conf` to perform any further configuration on the forwarder. Therefore, if you want to use Splunk Web to configure your forwarder, do so before you enable light forwarding.

Set up forwarding with the Splunk CLI

With the CLI, setting up forwarding is a two step process. First you enable forwarding on the Splunk instance. Then you start forwarding to a specified receiver.

To access the CLI, first navigate to `$SPLUNK_HOME/bin/`. This is unnecessary if you have added Splunk to your path.

To enable the forwarder mode, enter:

```
./splunk enable app [SplunkForwarder|SplunkLightForwarder] -auth <username>:<password>
```

Important: After this step, make sure you restart your Splunk instance as indicated! Attempting to start forwarding activity using the CLI before restarting splunkd will not work!

To disable the forwarder mode, enter:

```
./splunk disable app [SplunkForwarder|SplunkLightForwarder] -auth <username>:<password>
```

By disabling forwarding, this command reverts the Splunk instance to a full server.

Start forwarding activity from the Splunk CLI

To access the CLI, first navigate to `$SPLUNK_HOME/bin/`. This is unnecessary if you have added Splunk to your path.

To start forwarding activity, enter:

```
./splunk add forward-server <host>:<port> -auth <username>:<password>
```

To end forwarding activity, enter:

```
./splunk remove forward-server <host>:<port> -auth <username>:<password>
```

Note: Although this command ends forwarding activity, the Splunk instance remains configured as a forwarder. To revert the instance to a full Splunk server, use the `disable` command:

```
./splunk disable app [SplunkForwarder|SplunkLightForwarder] -auth <username>:<password>
```

Important: Make sure you restart your Splunk instance as indicated by the CLI to take these changes into account.

Troubleshoot forwarding and receiving

Confusing the receiver's listening and management ports

As part of setting up a fowarder, you specify the receiver (`hostname/IP_address` and `port`) that the forwarder will send data to. When you do so, be sure to specify the port that was designated as the receiver's *listening port* at the time the receiver was configured. See "Set up receiving with Splunk". Do **not** specify the receiver's management port. If you do mistakenly specify the receiver's management port, the receiver will generate an error similar to this:

```
splunkd.log:03-01-2010 13:35:28.653 ERROR TcpInputFd - SSL Error = error:140760FC:SSL routines:
splunkd.log:03-01-2010 13:35:28.653 ERROR TcpInputFd - ACCEPT_RESULT=-1 VERIFY_RESULT=0
splunkd.log:03-01-2010 13:35:28.653 ERROR TcpInputFd - SSL Error for fd from HOST:localhost.loc
splunkd.log:03-01-2010 13:35:28.653 ERROR TcpInputFd - SSL Error = error:140760FC:SSL routines:
splunkd.log:03-01-2010 13:35:28.653 ERROR TcpInputFd - ACCEPT_RESULT=-1 VERIFY_RESULT=0
```

```
splunkd.log:03-01-2010 13:35:28.653 ERROR TcpInputFd - SSL Error for fd from HOST:localhost.loc
splunkd.log:03-01-2010 13:35:28.653 ERROR TcpInputFd - SSL Error = error:140760FC:SSL routines:
splunkd.log:03-01-2010 13:35:28.654 ERROR TcpInputFd - ACCEPT_RESULT=-1 VERIFY_RESULT=0
splunkd.log:03-01-2010 13:35:28.654 ERROR TcpInputFd - SSL Error for fd from HOST:localhost.loc
splunkd.log:03-01-2010 13:35:28.654 ERROR TcpInputFd - SSL Error = error:140760FC:SSL routines:
splunkd.log:03-01-2010 13:35:28.654 ERROR TcpInputFd - ACCEPT_RESULT=-1 VERIFY_RESULT=0
```

Closed receiving socket

If a receiving indexer's queues become full, it will close the receiving socket, to prevent additional forwarders from connecting to it. If a forwarder with load-balancing enabled can no longer forward to that receiver, it will send its data to another indexer on its list. If the forwarder does not employ load-balancing, it will hold the data until the problem is resolved.

The receiving socket will reopen automatically when the queue gets unclogged.

Typically, a receiver gets behind on the dataflow because it can no longer write data due to a full disk or because it is itself attempting to forward data to another forwarder that is not accepting data.

The following warning message will appear in `splunkd.log` if the socket gets blocked:

```
Stopping all listening ports. Queues blocked for more than N seconds.
```

This message will appear when the socket reopens:

```
Started listening on tcp ports. Queues unblocked.
```

Answers

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

Configure forwarders with outputs.conf

Configure forwarders with outputs.conf

The outputs.conf file is unique to forwarders. It defines the forwarder configuration. Except for a few basic configurations available through Splunk Web or the CLI, all forwarder configuration takes place through `outputs.conf`. The topics describing various topologies, such as load balancing and data routing, provide detailed examples on configuring `outputs.conf`.

Note: Although `outputs.conf` is the critical file for configuring forwarders, it specifically addresses the *outputs* from the forwarder. To specify the *inputs* to a forwarder, you must configure the inputs separately, as you would for any other Splunk instance. For details on configuring inputs, see Add data and configure inputs in this manual.

Create and modify outputs.conf

There is no default `outputs.conf` file. When you enable a forwarder through Splunk Web or the CLI, Splunk creates an `outputs.conf` file in the directory of the currently running app. For example, if you're working in the search app, Splunk places the file in

$SPLUNK_HOME/etc/apps/search/local/. You can then edit it there.

To enable and configure a forwarder without using Splunk Web or the CLI, create an outputs.conf file and place it in this directory: $SPLUNK_HOME/etc/system/local/.

A single forwarder can have multiple outputs.conf files (for instance, one located in an apps directory and another in /system/local). To understand how to manage multiple outputs.conf files, see Configuration file precedence in this manual. No matter where the outputs.conf file resides, it acts globally on the forwarder (bearing in mind the issue of location precedence, as described in Configuration file precedence). For purposes of distribution and management simplicity, you might prefer to maintain just a single outputs.conf file, keeping it resident in the /system/local directory.

After making changes to outputs.conf, you must restart the forwarder for the changes to take effect.

See outputs.conf.spec and outputs.conf.example in $SPLUNK_HOME/etc/system/README/ for guidance and a template to use when creating or modifying outputs.conf.

Configuration levels

You can configure output processors at three levels of stanzas:

- **Global.** Here, you specify default target groups, as well as certain settings only configurable at the system-wide level for the output processor.
- **Target group.** A target group defines settings for one or more receivers. There can be one or more target groups per output processor. Most configuration settings can be specified at the target group level.
- **Single server.** You can specify configuration values for single servers (receivers) within a target group. This stanza type is optional.

Configurations at the more specific level take precedence. For example, if you specify compressed=true for a single receiver, the forwarder will send that receiver compressed data, even if compressed is set to "false" for the receiver's target group.

Target groups

A target group allows you to configure where and how Splunk will send data. Target groups do not control which events will be forwarded. For tcpout routing, events will be sent to all defined tcpout target groups by default, unless defaultGroup is set.

Here's the basic pattern for the target group stanza:

```
[<output_processor>:<target_group>]
server=<server1>, <server2>, ...
<attribute1> = <val1>
<attribute2> = <val2>
...
```

Available output processors are tcpout, syslog, and httpout.

To specify a server in a target group, use the format `<ipaddress_or_servername>:<port>`. For example, `myhost.Splunk.com:9997`.

To perform load balancing, you specify a target group with multiple receivers.

To perform cloning, you specify multiple target groups.

Note: For syslog and other output types, you must explicitly specify routing as described here: Route and filter data.

Set defaultGroup

You must include the `defaultGroup` attribute in your `[tcpout]` stanza:

```
[tcpout]
defaultGroup= <group1>, <group2>, ...
```

The defaultGroup specifies one or more target groups, defined later in `tcpout:<target_group>` stanzas. The forwarder will send all events to the specified defaultGroups. You can use an asterisk (`defaultGroup=*`) to send events to all defined target groups.

If you do **not** want to forward data automatically, you can set "defaultGroup" to a non-existent target group name (for example, "nothing").

Example

The following `outputs.conf` example contains three stanzas for sending tcpout to other Splunk receivers:

- Global settings. In this example, there are two settings: one to specify a defaultGroup, and another to enable local indexing as well as forwarding.
- Settings for a single target group consisting of two receivers. Here, we are specifying automatic load balancing between the two servers. See Set up load balancing in this manual for a detailed description of load balancing. We are also stipulating that the forwarder send the data in compressed form to the targeted receivers.
- Settings for one receiver within the target group. This stanza turns off compression for this particular receiver. The server-specific value for "compressed" takes precedence over the value set at the target group level.

```
[tcpout]
defaultGroup=my_indexers
indexAndForward=true

[tcpout:my_indexers]
autoLB=true
compressed=true
server=mysplunk_indexer1:9997, mysplunk_indexer2:9996

[tcpout-server://mysplunk_indexer1:9997]
compressed=false
```

Forwarding attributes

The `outputs.conf` file provides a large number of configuration options that offer considerable control and flexibility in forwarding. Of the attributes available, several are of particular interest:

Attribute	Default	Value
server	n/a	Required. Specifies the server(s) that will function as receivers for the forwarder. Configured at the target group level. This must be in the format `<ipaddress_or_servername>:<port>`.
defaultGroup	n/a	Required for `[tcpout]`. A comma-separated list of one or more target groups. Sends all events to all specified target groups. Set this toa non-existent group name, if you dont' want events automatically forwarded to a target group. Configurable only at the global level.
disabled	false	Specifies whether the stanza is disabled. If set to "true", it is equivalent to the stanza not being there.
indexAndForward	false	Specifies whether data should be indexed and stored locally, as well as forwarded. It can be specified only at the global level. This setting is not available for light forwarders.
sendCookedData	true	Specifies whether data is cooked before forwarding.
compressed	false	Specifies whether the forwarder sends compressed data.
maxQueueSize	1000	Specifies the maximum number of events queued on the forwarder.
autoLB	false	Specifies load balancing.
ssl....	n/a	Set of attributes for configuring SSL.

The outputs.conf.spec file provides details, including the default settings, for these and all other configuration options. In addition, most of these settings are discussed in topics dealing with specific forwarding scenarios.

HTTP forwarding

HTTP provides an alternative to TCP for forwarding data to a Splunk receiver. In certain situations, when dealing with firewalls, HTTP forwarding can ease network administrative issues. HTTP is available only for forwarding to a Splunk receiver, not for forwarding to third-party systems.

In `outputs.conf`, specify the `httpoutput` target group:

```
[httpoutput:<target_group>]
<attribute1> = <val1>
<attribute2> = <val2>
...
```

The target group stanza has these attributes:

Attribute	Default	Value
server	n/a	

		Required. This must be in the format `<ipaddress_or_servername>:<port>`.
ssl	true	Optional. Set to "true" or "false". If "true", HTTP output uses SSL.

Consolidate data from multiple machines

Consolidate data from multiple machines

One of the most common forwarding use cases is to consolidate data produced across numerous machines. Light forwarders running on machines generating data forward the data to a central Splunk indexer. Such forwarders ordinarily have little impact on their machines' performance. This diagram illustrates a common scenario, where light forwarders residing on machines running diverse operating systems send data to a single Splunk instance, which indexes and provides search capabilities across all the data:

The diagram illustrates a small deployment. In practice, the number of light forwarders in a data consolidation use case could number upwards into the thousands.

This type of use case is simple to configure:

1. Determine what data, originating from which machines, you need to access.

2. Install a Splunk instance, typically on its own server. This instance will function as the receiver. All indexing and searching will occur on it.

3. Enable the receiver through Splunk Web or the CLI. Using the CLI, enter this command from `$SPLUNK_HOME/bin/`:

```
./splunk enable listen <port> -auth <username>:<password>
```

For `<port>`, substitute the port you want the receiver to listen on.

4. If any of the forwarders will be running on a different operating system from the receiver, install the app for the forwarder's OS on the receiver. For example, assume the receiver in the diagram above is running on a Linux box. In that case, you'll need to install the Windows app on the receiver. You don't need to install the *NIX app. Since the receiver is on Linux, that app was already installed along with the rest of the Splunk instance.

After you have downloaded the relevant app, remove its `inputs.conf` file before enabling it, to ensure that its default inputs are not added to your indexer. For the Windows app, the location is: `$SPLUNK_HOME/etc/apps/windows/default/inputs.conf`.

5. Install a Splunk instance on each machine that will be generating data. These will become light forwarders that forward the data to the receiver.

6. Set up inputs for each forwarder. See Add data and configure inputs in this manual.

7. Configure each forwarder through Splunk Web or the CLI. Using the CLI from `$SPLUNK_HOME/bin/`, first enable each Splunk instance as a light forwarder:

```
./splunk enable app SplunkLightForwarder -auth <username>:<password>
```

Next, begin forwarding to the designated receiver:

```
./splunk add forward-server <host>:<port> -auth <username>:<password>
```

For `<host>:<port>`, substitute the host and port number of the receiver. For example, `splunk_indexer.acme.com:9995`.

Alternatively, if you have many forwarders, you can use an `outputs.conf` file to specify the receiver. For example:

```
[tcpout:my_indexers]
server= splunk_indexer.acme.com:9995
```

You can create this file once, then distribute copies of it to the `$SPLUNK_HOME/etc/system/local/` location of each forwarder.

Set up load balancing

Set up load balancing

In load balancing, a Splunk forwarder distributes data across several receiving Splunk instances. Each receiver gets a portion of the total data, and together the receivers hold all the data. To access the full set of forwarded data, you will need to set up distributed searching across all the receivers. For information on distributed search, see What is distributed search? in this manual.

Load balancing enables horizontal scaling for improved performance. In addition, its automatic switchover capability ensures resiliency in the face of machine outages. If a machine goes down, the forwarder simply begins sending data to the next available receiver.

Load balancing can also be of use when monitoring data from network devices like routers. To handle syslog and other data generated across port 514, a single forwarder can monitor port 514 and distribute the incoming data across several Splunk indexers.

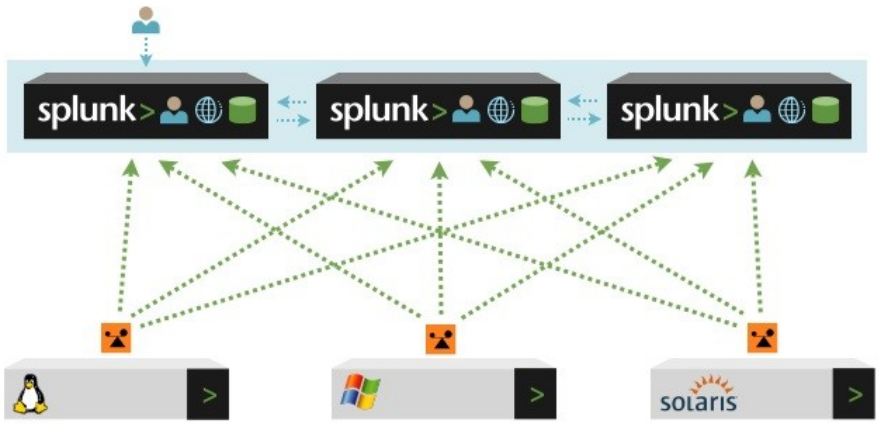
Splunk forwarders can perform two types of load balancing:

- Automatic load balancing: Forwarder routes data to different servers based on a specified time interval, for example, switching the data stream every 30 seconds, from server A to server B to server C and then back to server A.
- Round-robin load balancing: Forwarder routes data to different servers, switching with each new event, for example, event 1 goes to server A, event 2 to server B, event 3 to server C, and event 4 back to server A.

Regular forwarders can perform both types of load balancing. Light forwarders can perform only automatic load balancing.

For most purposes, automatic load balancing is recommended. It provides greater resiliency if a forwarder or receiver goes down. It also provides greater flexibility and easier configuration, because you can combine it with a DNS list. Round-robin load balancing can result in somewhat more even load balancing, because the forwarder switches receivers with each new event, but, in practice, any advantage diminishes at greater data volumes. In addition, round-robin requires that the forwarder perform parsing, consuming more RAM and CPU without improving the overall resiliency of the system. It is recommended only when you intend to distribute pre-indexing activity out to the edge network.

This diagram shows a distributed search scenario, in which three light forwarders are performing load balancing across three receivers:



Targets for automatic load balancing

When configuring the set of target receivers, you can employ either DNS or static lists.

DNS lists provide greater flexibility and simplified scale-up, particularly for large deployments. Through DNS, you can change the set of receivers without needing to re-edit each forwarder's `outputs.conf` file.

The main advantage of a static list is that it allows you to specify a different port for each receiver. This is useful if you need to perform load balancing across multiple receivers running on a single host. Each receiver can listen on a separate port.

Static list target

To use a static list for the target, you simply specify each of the receivers in the target group's `[tcpout]` stanza in the forwarder's `outputs.conf` file. In this example, the target group consists of three receivers, specified by IP address and port number:

```
[tcpout: my_LB_indexers]
autoLB=true
server=10.10.10.1:9997,10.10.10.2:9996,10.10.10.3:9995
```

The forwarder will load balance between the three receivers listed. If one receiver goes down, the forwarder will automatically switch to the next one available.

DNS list target

To use a DNS list, edit your forwarder's `outputs.conf` file to specify a single host in the target group's `[tcpout]` stanza. For example:

```
[tcpout:my_LB_indexers]
autoLB=true
server=splunkreceiver.mycompany.com:9997
```

In your DNS server, create a DNS A record for each host's IP address, referencing the server name you specified in `outputs.conf`. For example:

```
splunkreceiver.mycompany.com   A   10.10.10.1
splunkreceiver.mycompany.com   A   10.10.10.2
splunkreceiver.mycompany.com   A   10.10.10.3
```

The Splunk forwarder will use the DNS list to load balance, sending data in intervals, first to 10.10.10.1, then to 10.10.10.2, then to 10.10.10.3, and then to 10.10.10.1 again. If a receiver is not available, the forwarder skips it and sends data to the next one on the list.

If you have a topology with many forwarders, the DNS list method allows you to update the set of receivers by making changes in just a single location, without touching the forwarders' `outputs.conf` files.

Configure automatic load balancing for horizontal scaling

To configure automatic load balancing, first determine your needs, particularly your horizontal scaling and failover requirements. Then develop a topology based on those needs, possibly including multiple forwarders as well as receivers and a search head to search across the receivers.

Assuming the topology of three forwarders and three receivers illustrated by the diagram at the start of this topic, set up automatic load balancing with these steps:

1. Install and enable a set of three Splunk instances as receivers. This example uses a DNS list to designate the receivers, so they must all listen on the same port. For example, if the port is 9997, enable each receiver by going to its `$SPLUNK_HOME/bin/` location and using this CLI command:

```
./splunk enable listen 9997 -auth <username>:<password>
```

2. Install and enable the set of light forwarders. Once you've installed these Splunk instances, use this CLI command on each of them to enable forwarding:

```
./splunk enable app SplunkLightForwarder -auth <username>:<password>
```

3. Set up a DNS list with an A record for each receiver's IP address:

```
splunkreceiver.mycompany.com   A   10.10.10.1
splunkreceiver.mycompany.com   A   10.10.10.2
splunkreceiver.mycompany.com   A   10.10.10.3
```

4. Create a single `outputs.conf` file for use by all the forwarders. This one specifies the DNS server name used in the DNS list and the port the receivers are listening on:

```
[tcpout]
indexAndForward=false

[tcpout:my_LB_indexers]
disabled=false
autoLB=true
autoLBFrequency=40
server=splunkreceiver.mycompany.com:9997
```

This `outputs.conf` file also uses the autoLB attribute to specify automatic (instead of round-robin) load balancing and the autoLBFrequency attribute to set a frequency of 40 seconds. Every 40 seconds, the forwarders will switch to the next receiver. The default frequency, which rarely needs changing, is 30 seconds.

5. Distribute the `outputs.conf` file to all the forwarders, placing it in each forwarder's `$SPLUNK_HOME/etc/system/local/` directory.

Specify automatic load balancing from the CLI

You can also use the CLI to specify automatic load balancing. You do this when you start forwarding activity to a set of receivers, using this syntax:

```
./splunk add forward-server <host>:<port> -method=autobalance
```

where `<host>:<port>` is the host and port number of the receiver.

This example creates a load-balanced group of four receivers:

```
./splunk add forward-server -method=autobalance indexer1:9991
./splunk add forward-server -method=autobalance indexer2:9991
./splunk add forward-server -method=autobalance indexer3:9991
./splunk add forward-server -method=autobalance indexer4:9991
```

Route and filter data

Route and filter data

Forwarders can filter and route data to specific receivers based on criteria such as source, sourcetype, or patterns in the events themselves. For example, a forwarder can send all data from one group of hosts to one Splunk server and all other data to a second Splunk server. A forwarder can also look inside the events and filter or route accordingly. For example, you might want to inspect WMI event codes to filter or route Windows events. This topic describes a number of typical routing scenarios.

Besides routing to receivers, forwarders can also filter and route data to specific queues or discard the data altogether by routing to the null queue.

Only regular forwarders can route or filter data at the event level. Light forwarders do not have the ability to inspect individual events.

Here's a simple illustration of a forwarder routing data to three Splunk receivers:

This topic describes how to route event data to Splunk instances. See Forward data to third-party systems in this manual for information on routing to non-Splunk systems.

Configure routing

This is the basic pattern for defining most routing scenarios:

1. Determine what criteria to use for routing. How will you identify categories of events, and where will you route them?

2. Edit props.conf to add a TRANSFORMS-routing attribute to determine routing based on event metadata:

```
[<spec>]
TRANSFORMS-routing=<transforms_stanza_name>
```

`<spec>` can be:

- `<sourcetype>`, the sourcetype 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 <transforms_stanza_name> specified here when creating an entry in transforms.conf (below).

Examples later in this topic show how to use this syntax.

3. Edit transforms.conf to specify target groups and to set additional criteria for routing based on event patterns:

```
[<transforms_stanza_name>]
REGEX=<routing_criteria>
DEST_KEY=_TCP_ROUTING
FORMAT=<target_group>,<target_group>,....
```

Note:

- <transforms_stanza_name> must match the name you defined in props.conf.
- Enter the regex rules in <routing_criteria> that determine which events get routed. This line is required. Use REGEX = . if you don't need additional filtering beyond the metadata specified in props.conf.
- DEST_KEY should be set to _TCP_ROUTING to send events via TCP. It can also be set to _SYSLOG_ROUTING or _HTTPOUT_ROUTING for other output processors.
- Set FORMAT to a <target_group> that matches the group name you defined in outputs.conf. A comma separated list will clone events to multiple target groups.

Examples later in this topic show how to use this syntax.

4. Edit outputs.conf to define the target group(s) for the routed data:

```
[tcpout:<target_group>]
server=<ip>:<port>
```

Note:

- Set <target_group> to match the name you specified in transforms.conf.
- Set the IP address and port to match the receiving server.

The use cases described in this topic generally follow this pattern.

Filter and route event data to target groups

In this example, the forwarder filters three types of events, routing them to different target groups. The forwarder filters and routes according to these criteria:

- Events with a sourcetype of "syslog" to a load-balanced target group
- Events containing the word "error" to a second target group
- All other events to a default target group

Here's how you do it:

1. Edit props.conf in $SPLUNK_HOME/etc/system/local to set two TRANSFORMS-routing

attributes — one for syslog data and a default for all other data:

```
[default]
TRANSFORMS-routing=errorRouting

[syslog]
TRANSFORMS-routing=syslogRouting
```

2. Edit `transforms.conf` to set the routing rules for each routing transform:

```
[errorRouting]
REGEX=error
DEST_KEY=_TCP_ROUTING
FORMAT=errorGroup

[syslogRouting]
REGEX=.
DEST_KEY=_TCP_ROUTING
FORMAT=syslogGroup
```

Note: In this example, if a syslog event contains the word "error", it will route to `syslogGroup`, not `errorGroup`. This is due to the settings previously specified in `props.conf`. Those settings dictated that all syslog events be filtered through the `syslogRouting` transform, while all non-syslog (default) events be filtered through the `errorRouting` transform. Therefore, only non-syslog events get inspected for errors.

3. Edit `outputs.conf` to define the target groups:

```
[tcpout]
defaultGroup=everythingElseGroup

[tcpout:syslogGroup]
server=10.1.1.197:9996, 10.1.1.198:9997

[tcpout:errorGroup]
server=10.1.1.200:9999

[tcpout:everythingElseGroup]
server=10.1.1.250:6666
```

syslogGroup and errorGroup receive events according to the rules specified in `transforms.conf`. All other events get routed to the default group, everythingElseGroup.

Replicate a subset of data to a third-party system

This example uses data filtering to route two data streams. It forwards:

- All the data, in cooked form, to a Splunk indexer (10.1.12.1:9997)
- A replicated subset of the data, in raw form, to a third-party server (10.1.12.2:1234)

The example sends both streams as TCP. To send the second stream as syslog data, first route the data through an indexer.

1. Edit `props.conf`:

```
[syslog]
TRANSFORMS-routing = routeAll, routeSubset
```

2. Edit `transforms.conf`:

```
[routeAll]
REGEX=(.)
DEST_KEY=_TCP_ROUTING
FORMAT=Everything

[routeSubset]
REGEX=(SYSTEM|CONFIG|THREAT)
DEST_KEY=_TCP_ROUTING
FORMAT=Subsidiary,Everything
```

3. Edit `outputs.conf`:

```
[tcpout]
defaultGroup=nothing

[tcpout:Everything]
disabled=false
server=10.1.12.1:9997

[tcpout:Subsidiary]
disabled=false
sendCookedData=false
server=10.1.12.2:1234
```

For more information, see Forward data to third party systems in this manual.

Filter event data and send to queues

You can eliminate unwanted data by routing it to `nullQueue`, Splunk's `/dev/null` equivalent. When you filter out data in this way, the filtered data is not forwarded or added to the Splunk index at all, and doesn't count toward your indexing volume.

Although similar to forwarder-based routing, queue routing can be performed by either a forwarder or a full Splunk instance. It does not use the `outputs.conf` file, just `props.conf` and `transforms.conf`.

Discard specific events and keep the rest

This example discards all `sshd` events in `/var/log/messages` by sending them to `nullQueue`:

1. In `props.conf`, set the TRANSFORMS-null attribute:

```
[source::/var/log/messages]
TRANSFORMS-null= setnull
```

2. Create a corresponding stanza in `transforms.conf`. Set `DEST_KEY` to "queue" and `FORMAT` to "nullQueue":

```
[setnull]
```

```
REGEX = \[sshd\]
DEST_KEY = queue
FORMAT = nullQueue
```

That does it.

Keep specific events and discard the rest

Here's the opposite scenario. In this example, you use two transforms to keep *only* the `sshd` events. One transform routes `sshd` events to `indexQueue`, while another routes all other events to `nullQueue`.

Note: Null queue transforms are processed last, even when, as shown here, they appear first in `transforms.conf`.

1. In `props.conf`:

```
[source::/var/log/messages]
TRANSFORMS-set= setnull,setparsing
```

2. In `transforms.conf`:

```
[setnull]
REGEX = .
DEST_KEY = queue
FORMAT = nullQueue

[setparsing]
REGEX = \[sshd\]
DEST_KEY = queue
FORMAT = indexQueue
```

Note: The order of the stanzas doesn't matter in this example. Splunk processes the default, `nullQueue` transform as the last step, after processing all other transforms.

Filter WMI events

To filter on WMI events, you must use the [wmi] sourcetype stanza in `props.conf`. The following example uses regex to filter out two Windows event codes, 592 and 593:

In `props.conf`:

```
[wmi]
TRANSFORMS-wmi=wminull
```

In `transforms.conf`:

```
[wminull]
REGEX=(?m)^EventCode=(592|593)
DEST_KEY=queue
FORMAT=nullQueue
```

Filter data by target index

Splunk provides a **forwardedindex** filter that allows you to specify whether data gets forwarded, based on the data's target index. For example, if you have one data input targeted to "index1" and another targeted to "index2", you can use the filter to forward only the data targeted to index1, while ignoring the index2 data. The forwardedindex filter uses **whitelists** and **blacklists** to specify the filtering. For information on setting up multiple indexes, see the topic "Set up multiple indexes".

Use the `forwardedindex.<n>.whitelist|blacklist` attributes in `outputs.conf` to specify which data should get forwarded on an index-by-index basis. You set the attributes to regexes that filter the target indexes. By default, the forwarder forwards data targeted for all external indexes, as well as the data for the `_audit` internal index. It does not forward data to other internal indexes. The default `outputs.conf` file specifies that behavior with these attributes:

```
[tcpout]
forwardedindex.0.whitelist = .*
forwardedindex.1.blacklist = _.*
forwardedindex.2.whitelist = _audit
```

In most deployments, you will not need to override the default settings. See outputs.conf for more information on how to whitelist and blacklist indexes.

Forward all internal index data

If you want to forward all internal index data (and not just the data in `_audit`), you can override the default forwardedindex filter attributes like this:

```
#Forward everything
[tcpout]
forwardedindex.0.whitelist = .*
# disable these
forwardedindex.1.blacklist =
forwardedindex.2.whitelist =
```

Note: In previous releases you could achieve this result (internal index forwarding) by specifying the `_TCP_ROUTING = *` attribute/value in `inputs.conf`. This attribute/value pair no longer achieves that result. If you wish to reinstate the 4.0.x simpler behavior, set `forwardedindex.filter.disable = true` in `outputs.conf` instead.

Clone data

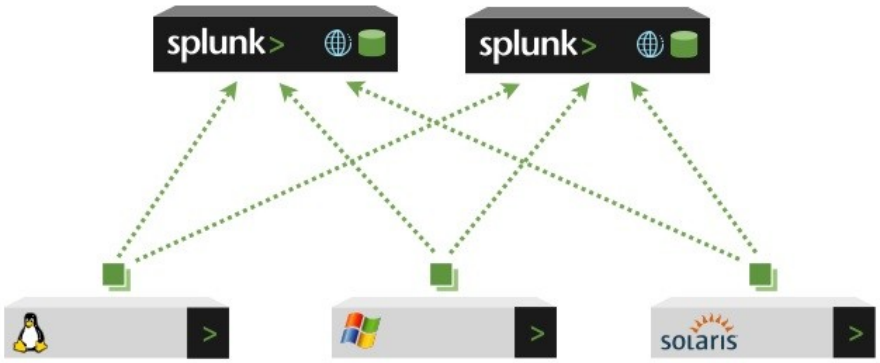
Clone data

In cloning, the forwarder sends duplicate copies of data to multiple target groups of receivers. Each target group can be either a single receiving server or a load-balanced group of receivers.

Cloning has value for enabling a number of key use cases, such as:

- Providing data redundancy to promote data availability
- Geo-diverse dataset replication, for fast local access
- Migration from one Splunk instance to another (not migration of past data)

In this simple scenario, three forwarders send duplicate copies of data to two Splunk servers:



Enable data cloning

The most direct way to set up cloning is by editing outputs.conf. Simply create multiple target groups. Each target group will automatically receive all the forwarder's data. Here is an example of specifying two target groups in a single `outputs.conf` file:

```
[tcpout]
...

[tcpout:indexer1]
server=10.1.1.197:9997

[tcpout:indexer2]
server=10.1.1.200:9999
```

The forwarder will send duplicate data streams to the servers specified in both the `indexer1` and `indexer2` target groups.

Provide data redundancy

Data cloning provides a good solution for situations requiring data redundancy. You can use a forwarder to send all data to two or more target groups. If a server in one target group goes down, users can continue to search their data by switching to another target group.

Although the target groups can each consist of single Splunk receivers, the recommended approach is to set up target groups of multiple load-balanced receivers. That way, if a server within a target group goes down while receiving data, the forwarder will automatically start forwarding data to the next server in the group, ensuring that the target group in total still receives all forwarded data. This provides a measure of protection by helping to ensure that two complete sets of the cloned data will exist in your system.

This example `outputs.conf` file configures a forwarder to clone raw data to two load-balanced target groups of indexers, with the indexing servers specified directly in the target groups. You can also use DNS lists to specify the target group servers, as described in DNS list target in this manual.

```
[tcpout]
indexAndForward=false

[tcpout:cloned_group1]
```

```
sendCookedData=false
autoLB=true
server=10.10.10.1:9997,10.10.10.2:9997,10.10.10.3:9997

[tcpout:cloned_group2]
sendCookedData=false
autoLB=true
server=10.1.1.197:9997, 10.1.1.198:9997, 10.1.1.199:9997, 10.1.1.200:9997
```

The forwarder will send full data streams to both `cloned_group1` and `cloned_group2`. The data will be load-balanced within each group, rotating among receivers every 30 seconds (the default frequency).

Specify cloning from the CLI

You can also use the CLI to specify cloning. You do this when you start forwarding activity to a set of receivers, using this syntax:

```
./splunk add forward-server <host>:<port> -method=clone
```

where `<host>:<port>` is the host and port number of the receiver.

This example sends cloned data to two receivers:

```
./splunk add forward-server -method=clone indexer1:9991
./splunk add forward-server -method=clone indexer2:9991
```

Forward data to third-party systems

Forward data to third-party systems

Splunk can forward raw data to non-Splunk systems. It can send the data over a plain TCP socket or packaged in standard syslog. Because it is forwarding to a non-Splunk system, it can send only raw data.

By editing `props.conf` and `transforms.conf`, you can configure the forwarder to route data conditionally to third-party systems, in the same way that it routes data conditionally to other Splunk instances. You can filter the data by host, source, or sourcetype. You can also use regex to further qualify the data.

TCP data

To forward TCP data to a third-party system, edit the forwarder's outputs.conf file to specify the receiving server and port. You must also configure the receiving server to expect the incoming data stream on that port.

To filter the data first, edit the forwarder's props.conf and transforms.conf files as well.

Edit the configuration files

To forward data, edit `outputs.conf`:

- Specify target groups for the receiving servers.
- Specify the IP address and TCP port for each receiving server.
- Set `sendCookedData` to `false`, so that the forwarder sends raw data.

To filter the data, edit `props.conf` and `transforms.conf`:

- In `props.conf`, specify the host, source, or sourcetype of your data stream. Specify a transform to perform on the input.
- In `transforms.conf`, define the transform and specify `_TCP_ROUTING`. You can also use regex to further filter the data.

Forward all data

This example shows how to send all the data from a Splunk forwarder to a third-party system. Since you are sending all the data, you only need to edit `outputs.conf`:

```
[tcpout]
indexAndForward = true

[tcpout:fastlane]
server = 10.1.1.35:6996
sendCookedData = false
```

Forward a subset of data

This example shows how to filter a subset of data and send the subset to a third-party system:

1. Edit `props.conf` and `transforms.conf` to specify the filtering criteria.

In `props.conf`, apply the `bigmoney` transform to all host names beginning with `nyc`:

```
[host::nyc*]
TRANSFORMS-nyc = bigmoney
```

In `transforms.conf`, configure the `bigmoney` transform to specify `TCP_ROUTING` as the `DEST_KEY` and the `bigmoneyreader` target group as the `FORMAT`:

```
[bigmoney]
REGEX = .
DEST_KEY=_TCP_ROUTING
FORMAT=bigmoneyreader
```

2. In `outputs.conf`, define the `bigmoneyreader` target group for the non-Splunk server, as well as a default target group to receive any other data. If you want to forward only the data specifically identified in `props.conf` and `transforms.conf`, set defaultGroup=nothing:

```
[tcpout]
defaultGroup = default-clone-group-192_168_1_104_9997
```

```
[tcpout:default-clone-group-192_168_1_104_9997]
server = 192.168.1.104:9997

[tcpout:bigmoneyreader]
server=10.1.1.197:7999
sendCookedData=false
```

The forwarder will send all data from host names beginning with `nyc` to the non-Splunk server specified in the `bigmoneyreader` target group. It will send data from all other hosts to the server specified in the `default-clone-group-192_168_1_104_9997` target group.

Syslog data

You can configure a forwarder to send data in standard syslog format. The forwarder sends the data through a separate output processor. You can also filter the data with `props.conf` and `transforms.conf`. You'll need to specify _SYSLOG_ROUTING as the DEST_KEY.

To forward syslog data, identify the third-party receiving server and specify it in a `syslog` target group in the forwarder's `outputs.conf` file.

Forward syslog data

The forwarder sends RFC 3164 compliant events to a TCP/UDP-based server and port, making the payload of any non-compliant data RFC 3164 compliant.

Note: If you have defined multiple event types for syslog data, the event type names must all include the string 'syslog'.

In `outputs.conf`, specify the `syslog` target group:

```
[syslog:<target_group>]
<attribute1> = <val1>
<attribute2> = <val2>
...
```

The target group stanza requires this attribute:

Required Attribute	Default	Value
server	n/a	This must be in the format `<ipaddress_or_servername>:<port>`. This is a combination of the IP address or servername of the syslog server and the port on which the syslog server is listening. Note that syslog servers use port 514 by default.

These attributes are optional:

Optional Attribute	Default	Value
type	udp	The transport protocol. Must be set to "tcp" or "udp".

priority	13	TCP priority. This must be in the format: <ddd>. This value will appear in the syslog header. Compute <ddd> as (<facility> * 8) + <severity>. If facility is 4 (security/authorization messages) and severity is 2 (critical conditions), priority value will be: (4 * 8) + 2 = 34.
syslogSourceType	n/a	This must be in the format `sourcetype::syslog`. The sourcetype for syslog messages.
timestampformat	""	The format used when adding a timestamp to the header. This must be in the format: <%b %e %H:%M:%S>. See Configure timestamps in this manual for details.

Send a subset of data to a syslog server

This example shows how to configure Splunk to forward data from hosts whose names begin with "nyc" to a syslog server named "loghost.example.com" over port 514:

1. Edit `props.conf` and `transforms.conf` to specify the filtering criteria.

In `props.conf`, apply the `send_to_syslog` transform to all host names beginning with `nyc`:

```
[host::nyc*]
TRANSFORMS-nyc = send_to_syslog
```

In `transforms.conf`, configure the `send_to_syslog` transform to specify `_SYSLOG_ROUTING` as the `DEST_KEY` and the `my_syslog_group` target group as the `FORMAT`:

```
[send_to_syslog]
DEST_KEY = _SYSLOG_ROUTING
FORMAT = my_syslog_group
```

2. In `outputs.conf`, define the `my_syslog_group` target group for the non-Splunk server:

```
[syslog:my_syslog_group]
server = loghost.example.com:514
```

Encrypt and authenticate data with SSL

Encrypt and authenticate data with SSL

The communication between forwarder and receiver can use SSL authentication and encryption, or just SSL encryption.

To enable SSL, edit each forwarder's `outputs.conf` file and each receiver's `inputs.conf` file.

Enable SSL on the forwarder

You enable SSL in the forwarder's `outputs.conf`. If you are using SSL just for encryption, you can set SSL attributes at any stanza level: default (`tcpout`), target group, or server. If you are also using SSL for authentication, you must specify SSL attributes at the server level. Each receiving server needs a stanza that specifies its certificate names.

SSL attributes

This table describes the set of SSL attributes:

Attribute	Value
sslCertPath	Full path to client certificate file.
sslPassword	Password for the certificate. Default is "password".
sslRootCAPath	Path to root certificate authority file.
sslVerifyServerCert	Set to true or false. Default is "false", which enables SSL for encryption only. If set to "true", the forwarder will determine whether the receiving server is authenticated, checking `sslCommonNametoCheck` and `altCommonNametoCheck` for a match. If neither matches, authentication fails.
sslCommonNameToCheck	Server's common name. Set only if `sslVerfyServerCert` is "true". The forwarder checks the common name of the server's certificate against this value.
altCommonNameToCheck	Server's alternate name. Set only if `sslVerfyServerCert` is "true". The forwarder checks the alternate name of the server's certificate against this value.

Set SSL for encryption only

Add attribute/value pairs at the appropriate stanza level. Here, the attributes are specified at the `tcpout` level, so they set the SSL defaults for this forwarder:

```
[tcpout]
sslCertPath=<full path to client certificate>
sslPassword=<password for cert>
sslRootCAPath=<optional path to root certificate authority file>
sslVerifyServerCert=false
```

Set up SSL for encryption and authentication

You must enable authentication at the server level of stanza, since each receiving server must have its own certificate:

```
[tcpout-server://<ip address>:<port>]
sslCertPath=<full path to client certificate>
sslPassword=<password for cert>
sslRootCAPath=<optional path to root certificate authority file>
sslVerifyServerCert=true
```

```
sslCommonNameToCheck=<server's common name>
altCommonNameToCheck=<server's alternate name>
```

You need to create a stanza for each receiver that the forwarder authenticates with.

Enable SSL on the receiver

You enable SSL in the receiver's `inputs.conf`. This involves two steps:

- Add an `[SSL]` stanza.
- Add listener stanzas for each port listening for SSL data.

Configure the [SSL] stanza

This table describes the attributes for the `[SSL]` stanza:

Attribute	Value
serverCert	Full path to server certificate file.
password	Password for the certificate, if any. If no password, leave blank or unset.
RootCA	Path to the root certificate authority file.
dhfile	Path to the `dhfile.pem` file. Optional.
requireClientCert	Set to true or false. Default is "false". If set to "true", the receiver will require a valid certificate from the client to complete the connection.

Here's how it looks:

```
[SSL]
serverCert=< path to server certificate>
password=<server certificate password>
rootCA=<certificate authority list>
dhfile=<optional path to dhfile.pem>
requireClientCert=<true|false> - set to "true" if authenticating
```

You can optionally specify certificates at the port level, thus allowing different sets of clients to connect through different ports.

Configure the listener stanzas

You need to add stanzas for each port the receiver will be listening to SSL data on.

For cooked, encrypted data coming from a Splunk forwarder, use this stanza:

```
[splunktcp-ssl:<port>]
```

The `port` must correspond to the port specified by the forwarder in `outputs.conf`.

For raw, encrypted data coming from either a Splunk forwarder or a third-party system, use this stanza instead:

```
[tcp-ssl:<port>]
```

Example SSL configuration

This example shows a configuration that sets up SSL encryption only (no authentication).

Edit the forwarder's `outputs.conf`:

```
[tcpout]
defaultGroup = ssl_group.domain.com_9996

[tcpout:ssl_group.domain.com_9996]
server = 10.1.5.148:9996

[tcpout-server://10.1.5.148:9996]
sslCertPath=$SPLUNK_HOME/etc/auth/server.pem
sslPassword=password
sslRootCAPath=$SPLUNK_HOME/etc/auth/ca.pem
sslVerifyServerCert=false
```

Edit the receiver's `inputs.conf`:

```
[SSL]
serverCert=$SPLUNK_HOME/etc/auth/server.pem
password=password
rootCA=$SPLUNK_HOME/etc/auth/cacert.pem
requireClientCert=false

[splunktcp-ssl:9996]
```

More about forwarders

More about forwarders

Certain capabilities are disabled in forwarders and light forwarders. This section describes forwarder capabilities in detail.

Splunk forwarder details

All functions and modules of the Splunk regular forwarder are enabled by default, with the exception of the distributed search module. The file `$SPLUNK_HOME/etc/apps/SplunkForwarder/default/default-mode.conf` includes this stanza:

```
[pipeline:distributedSearch]
disabled = true
```

For a detailed view of the exact configuration, see the configuration files for the SplunkForwarder application in `$SPLUNK_HOME/etc/apps/SplunkForwarder/default`.

Splunk light forwarder details

Most features of Splunk are disabled in the Splunk light forwarder. Specifically, the Splunk light forwarder:

- Disables event signing and checking whether the disk is full (`$SPLUNK_HOME/etc/apps/SplunkLightForwarder/default/default-mode.conf`).
- Limits internal data inputs to `splunkd` and metrics logs only, and makes sure these are forwarded (`$SPLUNK_HOME/etc/apps/SplunkLightForwarder/default/inputs.conf`).
- Disables all indexing (`$SPLUNK_HOME/etc/apps/SplunkLightForwarder/default/indexes.conf`).
- Does not use `transforms.conf` and does not fully parse incoming data, but the `CHARSET, CHECK_FOR_HEADER, NO_BINARY_CHECK, PREFIX_SOURCETYPE,` and `sourcetype` properties from `props.conf` are used.
- Disables the Splunk Web interface (`$SPLUNK_HOME/etc/apps/SplunkLightForwarder/default/web.conf`).
- Limits throughput to 256KBps (`$SPLUNK_HOME/etc/apps/SplunkLightForwarder/default/limits.conf`).
- Disables the following modules in `$SPLUNK_HOME/etc/apps/SplunkLightForwarder/default/default-mode.conf`:

```
[pipeline:indexerPipe]
disabled_processors= indexandforward, diskusage, signing,tcp-output generic-processor, sy

[pipeline:distributedDeployment]
disabled = true

[pipeline:distributedSearch]
disabled = true

[pipeline:fifo]
disabled = true

[pipeline:merging]
disabled = true

[pipeline:typing]
disabled = true

[pipeline:udp]
disabled = true

[pipeline:tcp]
disabled = true

[pipeline:syslogfifo]
disabled = true

[pipeline:syslogudp]
disabled = true

[pipeline:parsing]
disabled_processors=utf8, linebreaker, header, sendOut
```

```
[pipeline:scheduler]
disabled_processors = LiveSplunks
```

These modules include the deployment server (not the deployment client), distributed search, named pipes/FIFOs, direct input from network ports, and the scheduler.

The defaults for the light forwarder can be tuned to meet your needs by overriding the settings in `$SPLUNK_HOME/etc/apps/SplunkLightForwarder/default/default-mode.conf` on a case-by-case basis.

Configure event processing

Overview of event processing

Overview of event processing

Events are records of activity in log files, stored in Splunk indexes. They are primarily what Splunk indexes. Events provide information about the systems that produce the log files. The term **event data** refers to the contents of a Splunk index.

Here's a sample event:

```
172.26.34.223 - - [01/Jul/2005:12:05:27 -0700] "GET
/trade/app?action=logout HTTP/1.1" 200 2953
```

When Splunk indexes events, it:

- Configures character set encoding.
- Configures linebreaking for multi-line events.
- Identifies event timestamps (and applies timestamps to events if they do not exist).
- Extracts a set of useful standard fields such as `host`, `source`, and `sourcetype`.
- Improves data compression with segmentation.
- Dynamically assigns metadata to events, if specified.
- Anonymizes data if specified through sed or through configuration files.

For an overview of the Splunk indexing process, see the Indexing with Splunk chapter of this manual.

Configure character set encoding

Configure character set encoding

Splunk allows you to configure **character set encoding** for your data sources. Splunk has built-in character set specifications to support internationalization of your Splunk **deployment**. Splunk supports 71 languages (including 20 that aren't UTF-8 encoded). You can retrieve a list of Splunk's valid character encoding specifications by using the `iconv -l` command on most *nix systems.

Splunk attempts to apply UTF-8 encoding to your sources by default. If a source doesn't use UTF-8 encoding or is a non-ASCII file, Splunk will try to convert data from the source to UTF-8 encoding unless you specify a character set to use by setting the `CHARSET` key in `props.conf`.

Note: If a source's character set encoding is valid, but some characters from the specification are not valid in the encoding you specify, Splunk escapes the invalid characters as hex values (for example: "\xF3").

Supported character sets

Language	Code
Arabic	CP1256

Arabic	ISO-8859-6
Armenian i	ARMSCII-8
Belarus	CP1251
Bulgarian	ISO-8859-5
Czech	ISO-8859-2
Georgian	Georgian-Academy
Greek	ISO-8859-7
Hebrew	ISO-8859-8
Japanese	EUC-JP
Japanese	SHIFT-JIS
Korean	EUC-KR
Russian	CP1251
Russian	ISO-8859-5
Russian	KOI8-R
Slovak	CP1250
Slovenian	ISO-8859-2
Thai	TIS-620
Ukrainian	KOI8-U
Vietnamese	VISCII

Manually specify a character set

Manually specify a character set to apply to an input by setting the `CHARSET` key in `props.conf`:

```
[spec]
CHARSET=<string>
```

For example, if you have a host that is generating data in Greek (called "GreekSource" in this example) and that uses ISO-8859-7 encoding, set `CHARSET=ISO-8859-7` for that host in `props.conf`:

```
[host::GreekSource]
CHARSET=ISO-8859-7
```

Note: Splunk will only parse character encodings that have UTF-8 mappings. Some EUC-JP characters do not have a mapped UTF-8 encoding.

Automatically specify a character set

Splunk can automatically detect languages and proper character sets using its sophisticated character set encoding algorithm.

Configure Splunk to automatically detect the proper language and character set encoding for a particular input by setting `CHARSET=AUTO` for the input in `props.conf`. For example, if you want Splunk to automatically detect character set encoding for the host "my-foreign-docs", set `CHARSET=AUTO` for that host in `props.conf`:

```
[host::my-foreign-docs]
CHARSET=AUTO
```

If Splunk doesn't recognize a character set

If you want to use an encoding that Splunk doesn't recognize, train Splunk to recognize the character set by adding a sample file to the following directory:

```
$SPLUNK_HOME/etc/ngram-models/_<language>-<encoding>.txt
```

Once you add the character set specification file, you must restart Splunk. After you restart, Splunk can recognize sources that use the new character set, and will automatically convert them to UTF-8 format at index time.

For example, if you want to use the "vulcan-ISO-12345" character set, copy the specification file to the following path:

```
/SPLUNK_HOME/etc/ngram-models/_vulcan-ISO-12345.txt
```

Configure linebreaking for multi-line events

Configure linebreaking for multi-line events

Overview of multi-line events and event linebreaking

Some events are made up of more than one line. Splunk handles most of these kinds of events correctly by default, but there are cases of multi-line events that Splunk doesn't recognize properly by default. These require special configuration to change Slunk's default linebreaking behavior.

Multi-line event linebreaking and segmentation limitations

Splunk does apply limitations to extremely large events when it comes to linebreaking and segmentation:

- **Lines over 10,000 bytes:** Splunk breaks lines over 10,000 bytes into multiple lines of 10,000 bytes each when it indexes them. It appends the field `meta::truncated` to the end of each truncated section. However, Splunk still groups these lines into a single event.
- **Segmentation for events over 100,000 bytes:** Splunk only displays the first 100,000 bytes of an event in the search results. Segments after those first 100,000 bytes of a very long line are still searchable, however.
- **Segmentation for events over 1,000 segments:** Splunk displays the first 1,000 individual segments of an event as segments separated by whitespace and highlighted on mouseover. It displays the rest of the event as raw text without interactive formatting.

Configuration

Many event logs have a strict one-line-per-event format, but some do not. Usually, Splunk can automatically figure out the event boundaries. However, if event boundary recognition is not working as desired, you can set custom rules by configuring props.conf.

To configure multi-line events, examine the format of the events. Determine a pattern in the events to set as the start or end of an event. Then, edit `$SPLUNK_HOME/etc/system/local/props.conf`, and set the necessary attributes for your data handling.

There are two ways to handle multi-line events:

- Break the event stream into real events. This is recommended, as it increases indexing speed significantly. Use `LINE_BREAKER` (see below).
- Break the event stream into lines and reassemble. This is slower but affords more robust configuration options. Use any linebreaking attribute besides `LINE_BREAKER` (see below).

Linebreaking general attributes

These are the `props.conf` attributes that affect linebreaking:

`TRUNCATE = <non-negative integer>`

- Change the default maximum line length (in bytes).
- 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>`

- If not set, the raw stream will be broken into an event for each line delimited by \r or \n.
- If set, the given regex will be used to break the raw stream into events.
- The regex must contain a matching group.
- Wherever the regex matches, the start of the first matched group is considered the first text NOT in the previous event.
- The end of the first matched group is considered the end of the delimiter and the next character is considered the beginning of the next event.
- For example, "LINE_BREAKER = ([\r\n]+)" is equivalent to the default rule.
- The contents of the first matching group will not occur in either the previous or next events.
- **Note:** There is a significant speed boost by using the LINE_BREAKER to delimit multi-line events rather than using line merging to reassemble individual lines into events.

`LINE_BREAKER_LOOKBEHIND = <integer> (100)`

- Change the default lookbehind for the regex based linebreaker.
- When there is leftover data from a previous raw chunk, this is how far before the end of the raw chunk (with the next chunk concatenated) Splunk begins applying the regex.

`SHOULD_LINEMERGE = <true/false>`

- When set to true, Splunk combines several input lines into a single event, with configuration based on the attributes described below.
- Defaults to true.

Attributes available only when SHOULD_LINEMERGE = true

When SHOULD_LINEMERGE is set to true, these additional attributes have meaning:

`AUTO_LINEMERGE = <true/false>`

- Directs Splunk to use automatic learning methods to determine where to break lines in events.
- Defaults to true.

```
BREAK_ONLY_BEFORE_DATE = <true/false>
```

- When set to true, Splunk will create a new event if and only if it encounters a new line with a date.
- Defaults to false.

```
BREAK_ONLY_BEFORE = <regular expression>
```

- When set, Splunk will create a new event if and 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 always 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 will 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 will not break the last event before the current line.
- Defaults to empty.

```
MAX_EVENTS = <integer>
```

- Specifies the maximum number of input lines that will be added to any event.
- Splunk will break after the specified number of lines are read.
- Defaults to 256.

Examples

Specify event breaks

```
[my_custom_sourcetype]
BREAK_ONLY_BEFORE = ^\d+\s*$
```

This example instructs Splunk to divide events in a file or stream by presuming any line that consists of all digits is the start of a new event, for any source whose source type was configured or determined by Splunk to be *sourcetype::my_custom_sourcetype* .

Merge multiple lines into a single event

The following log event contains several lines that are part of the same request. The differentiator between requests is "Path". For this example, assume that all these lines need to be shown as a single event entry.

```
{{"2006-09-21, 02:57:11.58", 122, 11, "Path=/LoginUser
Query=CrmId=ClientABC&ContentItemId=TotalAccess&SessionId=3A1785URH117BEA&Ticke
Method=GET, IP=209.51.249.195, Content=", ""}}
{{"2006-09-21, 02:57:11.60", 122, 15, "UserData:<User CrmId="clientabc"
UserId="p12345678"><EntitlementList></EntitlementList></User>", ""}}
{{"2006-09-21, 02:57:11.60", 122, 15, "New Cookie:
SessionId=3A1785URH117BEA&Ticket=646A1DA4STF896EE&CrmId=clientabc&UserId=p12345
MANUser:
Version=1&Name=&Debit=&Credit=&AccessTime=&BillDay=&Status=&Language=&Country=&
""}}
```

To index this multiple line event properly, use the `Path` differentiator in your configuration. Add the following to your `$SPLUNK_HOME/etc/system/local/props.conf`:

```
[source::source-to-break]
SHOULD_LINEMERGE = True
BREAK_ONLY_BEFORE = Path=
```

This code tells Splunk to merge the lines of the event, and only break before the term `Path=`.

Answers

Have questions? Visit Splunk Answers and see what questions and answers the Splunk community has around multi-line event processing.

Handle event timestamps

Handle event timestamps

Look carefully at this sample event:

```
172.26.34.223 - - [01/Jul/2005:12:05:27 -0700] "GET
/trade/app?action=logout HTTP/1.1" 200 2953
```

Notice the time information in the event: `[01/Jul/2005:12:05:27 -0700]`. This is what is known as a **timestamp**. Splunk uses timestamps to correlate events by time, create the histogram in Splunk Web, and set time ranges for searches. Most events contain timestamps, and in those cases where an event doesn't contain timestamp information, Splunk attempts to assign a timestamp value to the event at index time.

Most events do not require additional handling of timestamp formatting, but there are situations that require the involvement of a Splunk administrator to help set things right. In the case of some sources and distributed deployments, for example, the Splunk admin may have to reconfigure timestamp recognition and formatting. Other timestamp-handling activities that the admin might undertake

include:

- Configuration of timestamp extraction for events with multiple timestamps
- Application of timestamp offsets (to correlate events from different timezones)
- Training Splunk to recognize a timestamp
- Tuning timestamp extraction performance

For more information about timestamps, see the Configure event timestamping chapter of this manual.

Extract default fields automatically

Extract default fields automatically

When Splunk indexes event data, it extracts by default a set of fields that are common to most events, and which are commonly used in Splunk searches and reports. These default fields include:

- `host`: Identifies the originating hostname or IP address of the network device that generated the event. Used to narrow searches to events that have their origins in a specific host.
- `source`: Identifies the filename or pathname from which the event was indexed. Used to filter events during a search, or as an argument in a data-processing command.
- `sourcetype`: Identifies the type of application, network, or device data that the event represents, such as `access_log` or `syslog`. A Splunk administrator can predefine source types, or they can be generated automatically by Splunk at index time. Use sourcetype to filter events during a search, or as an argument in a data-processing command.

For a full listing of the default fields that Splunk identifies during the indexing process, and examples of how they can be used in a search, see "Use default fields" in the User manual.

For detailed information on default field extraction, see "About default fields" in this manual.

Improve data compression with segmentation

Improve data compression with segmentation

Segmentation is what Splunk uses to break events up into searchable segments at index time, and again at search time. Segments can be classified as **major** or **minor**. To put it simply, minor segments are breaks within major segments. For example, the IP address `172.26.34.223` is, as a whole, a major segment. But this major segment can be broken down into minor segments such as `172` as well as groups of minor segments like `172.26.34`.

Splunk enables a Splunk admin to define how detailed the event segmentation should be. This is important because **index-time segmentation** affects indexing and search speed, impacts disk compression, and affects your ability to use typeahead functionality. **Search-time segmentation**, on the other hand, can also affect search speed as well as your ability to create searches by selecting items from the results displayed in Splunk Web.

Index-time segmentation is set through `segmenters.conf`, while search-time segmentation is set in the search results page in Splunk Web, as described here.

For more information about "index time" and "search time," see Index time versus search time, in this manual.

Levels of event segmentation

There are three levels of segmentation that the Splunk admin can choose from for index time and search time:

- **Inner segmentation** breaks events down into the smallest minor segments possible. For example, when an IP address such as `172.26.34.223` goes through inner segmentation, it is broken down into `172`, `26`, `34`, and `223`. Setting inner segmentation at index time leads to very efficient indexes in terms of search speed, but it also impacts indexing speed and restricts the typeahead functionality (it will only be able to typeahead at the minor segment level).
- **Outer segmentation** is the opposite of inner segmentation. Under outer segmentation only major segments are indexed. In the previous example, the IP address would not be broken down into any components. If you have outer segmentation set at index time you will be unable to search on individual pieces of the IP address without using wildcard characters. Indexes created using outer segmentation tend to be marginally more efficient than those created with full segmentation, but are not quite as efficient as those created through inner segmentation.
- **Full segmentation** is in some respects a combination of inner and outer segmentation. Under full segmentation, the IP address is indexed both as a major segment and as a variety of minor segments, including minor segment combinations like `172.26` and `172.26.34`. This is the least efficient indexing option, but it provides the most versatility in terms of searching.

Note: By default, index-time segmentation is set to a combination of inner and outer segmentation, and search-time segmentation is set to full segmentation.

For more information about changing the segmentation level, see Configure segmentation to manage disk usage in this manual.

Defining segmentation rules for specific hosts, sources, or source types

A Splunk admin can define index time and search time segmentation rules that apply specifically to events with particular hosts, sources, or sourcetypes. If you run searches that involve a particular sourcetype on a regular basis, you could use this to improve the performance of those searches. Similarly, if you typically index a large number of `syslog` events, you could use this feature to help decrease the overall disk space that those events take up.

For details about how to set these special segmentation rules up, see Configure custom segmentation for a host, source, or source type in this manual.

Assign metadata to events dynamically

Assign metadata to events dynamically

Ths feature allows you to dynamically assign metadata to files as they are being consumed by Splunk. Use this feature to specify source type, host, or other metadata dynamically for for incoming data. This feature is useful mainly with scripted data -- either a **scripted input** or a pre-existing file processed by a script.

Important: Splunk does not recommend using dynamic metadata assignment with ongoing monitoring (tail) inputs. For more information about file inputs, refer to Monitor files and directories in this manual.

To use this feature, you append a single dynamic input header to your file and specify the metadata fields you want to assign values to. The metadata fields most likely to be of interest are sourcetype, host, and source. You can see the list of all available pipeline metadata fields in transforms.conf.spec.

You can use this method to assign metadata instead of editing inputs.conf, props.conf and transforms.conf.

Configure a single input file

To use this feature for an existing input file, edit the file (either manually or with a script) to add a single input header:

```
***SPLUNK*** <metadata field>=<string> <metadata field>=<string> ...
```

- Set `<metadata field>=<string>` to a valid metadata/value pair. You can specify mutiple pairs. For example, `sourcetype=log4j host=swan`.
- Add the single header anywhere in your file. Any data following the header will be appended with the attributes and values you assign until the end of the file is reached.
- Add your file to `$SPLUNK_HOME/var/spool/splunk` or any other directory being monitored by Splunk.

Configure with a script

In the more common scenario, you write a script to dynamically add an input header to your incoming data stream. Your script can also set the header dynamically based on the contents of the input file.

Anonymize data with sed

Anonymize data with sed

This utility allows you to anonymize your data by replacing or substituting strings in it at index time using a sed script.

Most UNIX users are familiar with `sed`, a Unix utility which reads a file and modifies the input as specified by a list of commands. Now, you can use `sed`-like syntax to anonymize your data from `props.conf`.

Note: Edit or create a copy of `props.conf` in `$SPLUNK_HOME/etc/system/local`.

Define the sed script in props.conf

In a props.conf stanza, use `SEDCMD` to indicate a `sed script`:

```
[<stanza_name>]
SEDCMD-<class> = <sed script>
```

The `stanza_name` is restricted to the `host`, `source`, or `sourcetype` that you want to modify with your anonymization or transform.

The `sed script` applies only to the `_raw` field at index time. Splunk currently supports the following subset of `sed` commands: replace (`s`) and character substitution (`y`).

Note: You need to restart Splunk to implement the changes you made to `props.conf`

Replace strings with regex match

The syntax for a `sed` replace is:

```
SEDCMD-<class> = s/<regex>/<replacement>/flags
```

- `regex` is a PERL regular expression.
- `replacement` is a string to replace the regex match and uses "\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.

Example

Let's say you want to index data containing social security numbers and credit card numbers. At index time, you want to mask these values so that only the last four digits are evident in your events. Your `props.conf` stanza may look like this:

```
[source::.../accounts.log]
SEDCMD-accounts = s/ssn=\d{5}(\d{4})/ssn=xxxxx\1/g s/cc=(\d{4}-){3}(\d{4})/cc=xxxx-xxxx-xxxx-\2
```

Now, in you accounts events, social security numbers appear as `ssn=xxxxx6789` and credit card numbers will appear as `cc=xxxx-xxxx-xxxx-xxxx-1234`.

Substitute characters

The syntax for a `sed` character substitution is:

```
SEDCMD-<class> = y/<string1>/<string2>/
```

which substitutes each occurrence of the characters in `string1` with the characters in `string2`.

Example

Let's say you have a file you want to index, `abc.log`, and you want to substitute the capital letters "A", "B", and "C" for every lowercase "a", "b", or "c" in your events. Add the following to your `props.conf`:

```
[source::.../abc.log]
SEDCMD-abc = y/abc/ABC/
```

Now, if you search for `source="*/abc.log"`, you should not find the lowercase letters "a", "b", and "c" in your data at all. Splunk substituted "A" for each "a", "B" for each "b", and "C" for each "c'.

Anonymize data using configuration files

Anonymize data using configuration files

You may want to mask sensitive personal data that goes into logs. Credit card numbers and social security numbers are two examples of data that you may not want to index in Splunk. This page shows how to mask part of confidential fields so that privacy is protected but there is enough of the data remaining to be able to use it to trace events.

This example masks all but the last four characters of fields `SessionId` and `Ticket number` in an application server log.

An example of the desired output:

```
SessionId=##########7BEA&Ticket=##########96EE
```

A sample input:

```
"2006-09-21, 02:57:11.58",  122, 11, "Path=/LoginUser Query=CrmId=ClientABC&ContentItemId=Total
Ticket=646A1DA4STF896EE&SessionTime=25368&ReturnUrl=http://www.clientabc.com, Method=GET, IP=20
"2006-09-21, 02:57:11.60",  122, 15, "UserData:<User CrmId="clientabc" UserId="p12345678"><Enti
"2006-09-21, 02:57:11.60",  122, 15, "New Cookie: SessionId=3A1785URH117BEA&Ticket=646A1DA4STF8
p12345678&AccountId=&AgentHost=man&AgentId=man, MANUser: Version=1&Name=&Debit=&Credit=&AccessT
&Language=&Country=&Email=&EmailNotify=&Pin=&PinPayment=&PinAmount=&PinPG=&PinPGRate=&PinMenu=&
```

Configuration

To mask the data, modify the `props.conf` and `transforms.conf` files in your `$SPLUNK_HOME/etc/system/local/` directory.

props.conf

Edit `$SPLUNK_HOME/etc/system/local/props.conf` and add the following:

```
[<spec>]
TRANSFORMS-anonymize = session-anonymizer, ticket-anonymizer
```

`<spec>` can be:

1. <sourcetype>, the source type of an event
2. host::<host>, where <host> is the host for an event
3. source::<source>, where <source> is the source for an event.

`session-anonymizer` and `ticket-anonymizer` are TRANSFORMS class names whose actions are defined in `transforms.conf`. For your data, use the class names you create in `transforms.conf`.

transforms.conf

In `$SPLUNK_HOME/etc/system/local/transforms.conf`, add your TRANSFORMS:

```
[session-anonymizer]
```

```
REGEX = (?m)^(.*)SessionId=\w+(\w{4}[&"].*)$
FORMAT = $1SessionId=########$2
DEST_KEY = _raw
[ticket-anonymizer]
REGEX = (?m)^(.*)Ticket=\w+(\w{4}&.*)$
FORMAT = $1Ticket=########$2
DEST_KEY = _raw
```

`REGEX` should specify the regular expression that will point to the string in the event you want to anonymize.

Note: The regex processor can't handle multi-line events. To get around this you need to specify in `transforms.conf` that the event is multi-line. Use the `(?m)` before the regular expression.

`FORMAT` specifies the masked values. `$1` is all the text leading up to the regex and `$2` is all the text of the event after the regex.

`DEST_KEY = _raw` specifies to write the value from `FORMAT` to the raw value in the log - thus modifying the event.

Configure event timestamping

How timestamp assignment works

How timestamp assignment works

Splunk uses timestamps to correlate events by time, create the timeline histogram in Splunk Web and to set time ranges for searches. Timestamps are assigned to events at index time. Most events get a timestamp value assigned to them based on information in the raw event data. If an event doesn't contain timestamp information, Splunk attempts to assign a timestamp value to the event as it's indexed. Splunk stores timestamp values in the `_time` field (in UTC time format).

Timestamp processing is one of the key steps in event processing. For more information on event processing, see Configure event processing.

Considerations when adding new data

If your data turns out to require timestamp configuration beyond what Splunk does automatically, you must re-index that data once you've configured its timestamp extraction. It's a good idea to test a new data input in a "sandbox" Splunk instance (or just a separate index) before adding it to your production Splunk instance in case you have to clean it out and re-index it a few times to get it just right.

Precedence rules for timestamp assignment

Splunk uses the following precedence to assign timestamps to events:

1. Look for a time or date in the event itself using an explicit TIME_FORMAT if provided.

Use positional timestamp extraction for events that have more than one timestamp value in the raw data.

2. If no TIME_FORMAT is provided, or no match is found, attempt to automatically identify a time or date in the event itself.

Use positional timestamp extraction for events that have more than one timestamp value in the raw data.

3. If an event doesn't have a time or date, use the timestamp from the most recent previous event of the same source.

4. If no events in a source have a date, look in the source (or file) name (Must have time in the event).

5. For file sources, if no time or date can be identified in the file name, use the modification time on the file.

6. If no other timestamp is found, set the timestamp to the current system time (at the event's index time).

Configure timestamps

Most events don't require any special timestamp handling. For some sources and distributed deployments, you may have to configure timestamp formatting to extract timestamps from events. Configure Splunk's timestamp extraction processor by editing props.conf. For a complete discussion of the timestamp configurations available in `props.conf`, see Configure timestamp recognition.

You can also configure Splunk's timestamp extraction processor to:

- Apply timezone offsets.
- Pull the correct timestamp from events with more than one timestamp.
- Improve indexing performance.

Finally, train Splunk to recognize new timestamp formats.

Configure timestamp recognition

Configure timestamp recognition

Splunk uses timestamps to correlate events by time, create the histogram in Splunk Web and to set time ranges for searches. Timestamps are assigned to events at index time.

Splunk assigns a timestamp to most events based on information in the raw event data. If an event doesn't contain timestamp information, Splunk attempts to assign a timestamp value to the event as it's indexed. Splunk stores timestamp values in the `_time` field (in UTC time format).

Most events don't require any special timestamp handling; you can just let Splunk handle it without any configuration.

Precedence rules for timestamp assignment

Splunk uses the following precedence to assign timestamps to events:

1. Look for a time or date in the event itself using an explicit TIME_FORMAT if provided.

Use positional timestamp extraction for events that have more than one timestamp value in the raw data.

2. If no TIME_FORMAT is provided, or no match is found, attempt to automatically identify a time or date in the event itself.

3. If an event doesn't have a time or date, use the timestamp from the most recent previous event of the same source.

4. If no events in a source have a time or date, look in the source (or file) name.

5. For file sources, if no time or date can be identified in the file name, use the modification time on the file.

6. If no other timestamp is found, set the timestamp to the current system time (the time at which the event is indexed by Splunk).

Configure timestamps

Most events don't require any special timestamp handling; you can just let Splunk handle it without any configuration.

For some sources and distributed deployments, you may have to configure timestamp formatting to extract timestamps from events. Configure Splunk's timestamp extraction processor by editing props.conf.

Configure how Splunk recognizes timestamps by editing props.conf. Splunk uses strptime() formatting to identify timestamp values in your events. Specify what Splunk recognizes as a timestamp by setting a `strptime()` format in the `TIME_FORMAT=` key.

Note: If your event has more than one timestamp, set Splunk to recognize the correct timestamp with positional timestamp extraction.

Use `$SPLUNK_HOME/etc/system/README/props.conf.example` as an example, or create your own `props.conf`. Make any configuration changes to a copy of `props.conf` in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`.

Configure any of the following attributes in `props.conf` to set Splunk's timestamp recognition. Refer to `$SPLUNK_HOME/etc/system/README/props.conf.spec` for full specification of the keys.

```
[<spec>]
DATETIME_CONFIG = <filename relative to $SPLUNK_HOME>
MAX_TIMESTAMP_LOOKAHEAD = <integer>
TIME_PREFIX = <regular expression>
TIME_FORMAT = <strptime-style format>
TZ = <posix timezone string>
MAX_DAYS_AGO = <integer>
MAX_DAYS_HENCE = <integer>
```

`[<spec>]`

- `<spec>` indicates what to apply timestamp extraction to. This can be one of the following:
  - `<sourcetype>`, the sourcetype of an event.
  - `host::<host>`, where `<host>` is the host of an event.
  - `source::<source>`, where `<source>` is the source of an event.
- If an event contains data that matches the value of `<spec>`, then the timestamp rules specified in the stanza apply to that event.
- Add additional stanzas to customize timestamp recognition for any type of event.

`DATETIME_CONFIG = <filename relative to $SPLUNK_HOME>`

- Specify a file to use to configure Splunk's timestamp processor (by default Splunk uses `$SPLUNK_HOME/etc/datetime.xml`).

- To use a custom `datetime.xml`, specify the correct path to your custom file in all keys that refer to `datetime.xml`.
- Set `DATETIME_CONFIG = NONE` to prevent the timestamp processor from running.
- Set `DATETIME_CONFIG = CURRENT` to assign the current system time to each event as it's indexed.

`MAX_TIMESTAMP_LOOKAHEAD = <integer>`

- Specify how far (how many characters) into an event Splunk should look for a timestamp.
- Default is 150 characters.
- Set to 0 to assign current system time at an event's index time.

`TIME_PREFIX = <regular expression>`

- Use a regular expression that points to the space exactly before your event's timestamp.
  - For example, if your timestamp follows the phrase `Time=`, your regular expression should match this part of the event.
- The timestamp processor only looks for a timestamp **after** the `TIME_PREFIX` in an event.
- Default is none (empty).

`TIME_FORMAT = <strptime-style format>`

- Specify a strptime() format string to extract the date.
- Set strptime() values in the order that matches the order of the elements in the timestamp you want to extract.
- Splunk's timestamp processor starts processing `TIME_FORMAT` immediately after a matching `TIME_PREFIX` value.
- Doesn't support in-event timezones.
- `TIME_FORMAT` starts reading after a matching `TIME_PREFIX`.
- The `<strptime-style format>` value must contain the hour, minute, month, and day.
- Default is empty.

`TZ = <timezone string>`

- Specify a timezone setting using a value from the zoneinfo TZID database.
- For more details and examples learn how to configure timezone offsets.
- Default is empty.

`MAX_DAYS_AGO = <integer>`

- Specify the maximum number of days in the past (from the current date) for an extracted date to be valid.
- For example, if `MAX_DAYS_AGO = 10` then dates that are older than 10 days ago are ignored.
- Default is 2000.

Note: You must configure this setting if your data is more than 2000 days old.

`MAX_DAYS_HENCE = <integer>`

- Specify the maximum number of days in the future (from the current date) for an extracted date to be valid.
- For example, if `MAX_DAYS_HENCE = 3` then dates that are more than 3 days in the future are ignored.
- The default value (2) allows dates that are tomorrow.

Note: If your machines have the wrong date set or are in a timezone that is one day ahead, set this value to at least 3.

Enhanced strptime() support

Configure timestamp parsing in props.conf with the `TIME_FORMAT=` key. Splunk implements an enhanced version of Unix strptime() that supports additional formats (allowing for microsecond, millisecond, any time width format, and some additional time formats for compatibility). See the table below for a list of the additionally supported strptime() formats.

In previous versions, Splunk parsed timestamps using only the standard Linux strptime() conversion specifications. Now, in addition to standard Unix strptime() formats, Splunk's strptime() implementation supports recognition of the following date-time formats:

%N	For GNU date-time nanoseconds. Specify any sub-second parsing by providing the width: %3N = milliseconds, %6N = microseconds, %9N = nanoseconds.
%Q,%q	For milliseconds, microseconds for Apache Tomcat. %Q and %q can format any time resolution if the width is specified.
%I	For hours on a 12-hour clock format. If %I appears after %S or %s (like "%H:%M:%S.%l") it takes on the log4cpp meaning of milliseconds.
%+	For standard UNIX date format timestamps.
%v	For BSD and OSX standard date format.
%z, %::z, %:::z	GNU libc support.
%o	For AIX timestamp support (%o used as an alias for %Y).
%p	The locale's equivalent of AM or PM. (Note: there may be none.)

strptime() format expression examples

Here are some sample date formats with the `strptime()` expressions that handle them:

1998-12-31	%Y-%m-%d
98-12-31	%y-%m-%d
1998 years, 312 days	%Y years, %j days
Jan 24, 2003	%b %d, %Y
January 24, 2003	%B %d, %Y
q|25 Feb '03 = 2003-02-25|	q|%d %b '%y = %Y-%m-%d|

Note: Splunk does not currently recognize non-English month names in timestamps. If you have an app that's writing non-English month names to log files, reconfigure the app to use numerical months, if possible.

Examples

Your data might contain an easily recognizable timestamp to extract such as:

```
...FOR: 04/24/07 PAGE 01...
```

The entry in `props.conf` is:

```
[host::foo]
TIME_PREFIX = FOR:
TIME_FORMAT = %m/%d/%y
```

Your data might contain other information that Splunk parses as timestamps, for example:

```
...1989/12/31 16:00:00 ed May 23 15:40:21 2007...
```

Splunk extracts the date as Dec 31, 1989, which is not useful. In this case, configure `props.conf` to extract the correct timestamp from events from `host::foo`:

```
[host::foo]
TIME_PREFIX = \d{4}/\d{2}/\d{2} \d{2}:\d{2}:\d{2} \w+\s
TIME_FORMAT = %b %d %H:%M:%S %Y
```

This configuration assumes that all timestamps from `host::foo` are in the same format. Configure your `props.conf` stanza to be as granular as possible to avoid potential timestamping errors.

Configure timestamps in other ways

You can also configure Splunk's timestamp extraction processor to:

- Apply timezone offsets.
- Pull the correct timestamp from events with more than one timestamp.
- Improve indexing performance.
- Train Splunk to recognize new timestamp formats.

In addition, you can use your browser's locale setting to configure how the browser formats Splunk timestamps. For information on the setting the browser locale, see User language and locale.

Answers

Have questions? Visit Splunk Answers and see what questions and answers the Splunk community has around timestamp recognition and configuration.

Improve Splunk's ability to recognize timestamps

Improve Splunk's ability to recognize timestamps

Splunk recognizes most timestamps by default. For more information read How timestamps work. If Splunk doesn't recognize a particular timestamp, you can use the `train dates` command to teach Splunk the pattern. The output of `train dates` is a regular expression that you can add to `datetime.xml` and `props.conf` to configure the unique timestamp extraction.

The `train` command lets you interactively teach Splunk new patterns for timestamps, fields, and sourcetypes. for more information about `train` and the different arguments you can use with it, go to `$SPLUNK_HOME/bin` and refer to the `train` help page:

```
./splunk help train
```

Important: Use `train dates` only when you can't configure the timestamp with props.conf.

Steps to configure timestamps with train dates

To teach Splunk a new timestamp pattern, complete the following steps:

1. Copy a sampling of your timestamp data into a plain text file.

Splunk learns the pattern of the timestamp based on the patterns in this text file.

2. Run the train dates command.

This feature is interactive. When prompted, provide the path to the text file containing your timestamp data. The command produces a regular expression for your timestamp.

3. Create a custom datetime.xml.

Copy the output of the `train` command into a copy of `datetime.xml` file.

Note: The default `datetime.xml` file is located in `$SPLUNK_HOME/etc/datetime.xml`. Do not modify this file; instead, copy the default `datetime.xml` into a custom application directory in `$SPLUNK_HOME/etc/apps/` or `$SPLUNK_HOME/etc/system/local/`. Refer to the topic about applications in this manual for more information.

4. Edit your local props.conf.

Include the path to your custom `datetime.xml` file in the relevant stanzas.

```
./splunk [command]
```

Run the train dates command

The `train` command is an interactive CLI tool. For Splunk to learn a new date format, you need to explicitly provide a file and pattern. Afterwards, Splunk returns a string for you to add to `datetime.xml`.

1. To begin training Splunk to recognize a new timestamp, go to `$SPLUNK_HOME/bin` and type:

```
./splunk train dates
```

Splunk prompts you for an action:

```
------------------------------------------------------
What operation do you want to perform? (default=learn)
------------------------------------------------------
                            Enter choice: [Learn]/Test/Quit >
```

The default action is **Learn**.

2. To perform the training operation, type "L", "l", or "learn". Click enter.

Splunk prompts you to give it the sample file you want to use to train it:

```
Enter full filename from which to learn dates > sampling.txt
```

3. Enter the path of the file on your Splunk server (this step doesn't allow tab-complete).

Splunk displays the first line of your sample and asks you to teach it the values for the timestamp:

```
-----------------------------------
Interactively learning date formats.
-----------------------------------
INSTRUCTIONS: If a sample line does not have a timestamp, hit Enter.
If it does have a timestamp, enter the timestamp separated by commas
in this order: month, day, year, hour, minute, second, ampm, timezone.
Use a comma as a placeholder for missing values.  For example, for a
sample line like this "[Jan/1/08 11:56:45 GMT] login", the input
should be: "Jan, 1, 08, 11, 56, 45, , GMT" (note missing AM/PM).
Spaces are optional.
SAMPLE LINE 1:
        Tue Jul 10 21:23:06 PDT 2007 Received Trade 330 with detail user: user3456 date: date:
        23:06 action: sell 3583 MNAG @ 42
------------------------------------------------------------------------------
Enter timestamp values as: month, day, year, hour, minute, second, ampm, timezone.
   > 7, 10, 2007, 9, 23, 06, pm, PDT
```

4. Enter values for month, day, year, hour, minute, second, ampm, and timezone (as shown above). This trains Splunk to recognize the values you enter as the designated portions of the timestamp.

If the values are sufficient, Splunk displays:

```
Learned pattern.
------------------------------------------------------------------------------
If you are satisfied that the timestamps formats have been learned, hit control-c.
------------------------------------------------------------------------------
```

5. After you hit **control-c**, Splunk displays:

```
Patterns Learned.
It is highly recommended that you make changes to a copy of the default datetime.xml file.
For example, copy "/Applications/splunk/etc/datetime.xml" to "/Applications/splunk/etc/system/l
In that custom file, add the below timestamp definitions, and add the pattern names
to timePatterns and datePatterns list.
For more details, see http://www.splunk.com/doc/latest/admin/TrainTimestampRecognition
------------------------------------------------------------------------------
<define name="trainwreck_1_date" extract="day,litmonth,year,">
        <text><![CDATA[:\d+\s\w+\s(\d+)\s(\w+)\s(\d+)]]></text>
```

```
</define>
<define name="trainwreck_1_time" extract="hour,minute,second,ampm,">
        <text><![CDATA[(\d+):(\d+):(\d+)\s(\w+)]]></text>
</define>
-------------------------------------------------------
What operation do you want to perform? (default=learn)
-------------------------------------------------------
                            Enter choice: [Learn]/Test/Quit > q
```

6. Check the output.

- If it's correct, quit. Then, copy the output and continue to the next section.
- If it's not correct, enter the **Learn** choice to re-train Splunk.

Create a custom datetime.xml

After running `train`, Splunk outputs a string describing the new timestamp pattern.

In your custom `datetime.xml` file:

1. Paste the string returned from `train` before the `<timePatterns>` and `<datePatterns>` stanzas.

2. Add `<use name="define name"/>` for both `<timePatterns>` and `<datePatterns>` with the string defined as the `<define name="string"`.

Example:

For the following `train dates` output:

```
<define name="_utcepoch" extract="utcepoch">
    <text><![CDATA[((?<=^|[\s#,"=\(\[\|\{])(?:1[01]|9)\d{8}|^@[\da-fA-F]{16,24})(?:\d{3})?(?![\
</define>
```

The modified `datetime.xml` file might look something like:

```
<define name="_utcepoch" extract="utcepoch">
    <text><![CDATA[((?<=^|[\s#,"=\(\[\|\{])(?:1[01]|9)\d{8}|^@[\da-fA-F]{16,24})(?:\d{3})?(?![\
</define>
<timePatterns>
      <use name="_time"/>
      <use name="_hmtime"/>
      <use name="_hmtime"/>
      <use name="_dottime"/>
      <use name="_combdatetime"/>
      <use name="_utcepoch"/>
</timePatterns>
<define name="_utcepoch" extract="utcepoch">
    <text><![CDATA[((?<=^|[\s#,"=\(\[\|\{])(?:1[01]|9)\d{8}|^@[\da-fA-F]{16,24})(?:\d{3})?(?![\
</define>
<datePatterns>
      <use name="_usdate"/>
      <use name="_isodate"/>
      <use name="_eurodate"/>
      <use name="_bareurlitdate"/>
```

```
      <use name="_orddate"/>
      <use name="_combdatetime"/>
      <use name="_masheddate"/>
      <use name="_masheddate2"/>
      <use name="_utcepoch"/>
</datePatterns>
```

Edit your local props.conf

To apply your custom timestamp, Splunk needs to know where to find your new `datetime.xml`.

Modify `props.conf` to:

1. Add a `DATETIME_CONFIG` key to the timestamp configuration stanzas.

2. Set the value of `DATETIME_CONFIG` to the path of your custom `datetime.xml`.

Note: See all of the keys you can set in a stanza to configure timestamp recognition.

Example:

This example applies a custom `datetime.xml` to events from the host, "london".

```
[host::london]
DATETIME_CONFIG = /etc/system/local/datetime.xml
```

You can set custom timestamp extraction patterns for any host, source, or sourcetype by editing `props.conf`.

Configure timestamp assignment for events with multiple timestamps

Configure timestamp assignment for events with multiple timestamps

If an event contains more than one recognizable timestamp, you can tell Splunk to use a particular timestamp. This is especially useful when indexing events that contain syslog host-chaining data.

Configure positional timestamp extraction by editing props.conf.

Configure positional timestamp extraction in props.conf

Configure Splunk to recognize a timestamp anywhere in an event by adding `TIME_PREFIX =` and `MAX_TIMESTAMP_LOOKAHEAD =` keys to a `[<spec>]` stanza in `props.conf`. Set a value for `MAX_TIMESTAMP_LOOKAHEAD =` to tell Splunk how far into an event to look for the timestamp. Set a value for `TIME_PREFIX =` to tell Splunk what pattern of characters to look for to indicate the beginning of the timestamp.

Note: Use `$SPLUNK_HOME/etc/system/README/props.conf.example` as an example, or

create your own `props.conf`. Make any configuration changes to a copy of `props.conf` in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`.

Example: If an event looks like:

```
1989/12/31 16:00:00 ed May 23 15:40:21 2007 ERROR UserManager - Exception thrown Ignoring unsup
```

To identify the timestamp: **May 23 15:40:21 2007**

Configure `props.conf`:

```
[source::/Applications/splunk/var/spool/splunk]
TIME_PREFIX = \d{4}/\d{2}/\d{2} \d{2}:\d{2}:\d{2} \w+\s
MAX_TIMESTAMP_LOOKAHEAD = 44
```

Note: Optimize the speed of timestamp extraction by setting the value of `MAX_TIMESTAMP_LOOKAHEAD =` to look only as far into an event as needed for the timestamp you want to extract. In this example `MAX_TIMESTAMP_LOOKAHEAD =` is optimized to look 44 characters into the event .

Specify timezones of timestamps

Specify timezones of timestamps

If you're indexing data from different timezones, use timezone offsets to ensure that they're correctly correlated when you search. You can configure timezones based on the host, source, or source type of an event.

Configure timezones in props.conf. By default, Splunk applies timezones using these rules, in the following order:

1. Use the timezone in raw event data (for example, PST, -0800).

2. Use `TZ` if it is set in a stanza in `props.conf` and the event matches the host, source, or source type specified by a stanza.

3. Use the timezone of the Splunk server that indexes the event.

Specify time zones in props.conf

Use `$SPLUNK_HOME/etc/system/README/props.conf.example` as an example, or create your own `props.conf`. Make any configuration changes to a copy of `props.conf` in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`.

Configure time zones by adding a `TZ =` key to a timestamp configuration stanza for a host, source, or sourcetype in `props.conf`. The Splunk `TZ =` key recognizes zoneinfo TZID's (See all the timezone TZ ID's in the zoneinfo (TZ) database). Set a `TZ =` value to a TZID of the appropriate timezone for any host, source, or source type. The TZ for a host, source, or source type should be set

to the timezone of the events coming from that host, source, or sourcetype.

Note that the timezone of the indexer is not configured in Splunk. As long as the time is set correctly on the host OS of the indexer, offsets to event timezones will be calculated correctly.

Examples

Events are coming to this indexer from New York City (in the US/Eastern timezone) and Mountain View, California (US/Pacific). To correctly handle the timestamps for these two sets of events, the `props.conf` for the indexer needs the timezone to be specified as US/Eastern and US/Pacific respectively.

The first example sets the timezone of events from host names that match the regular expression `nyc.*` with the US/Eastern timezone.

```
[host::nyc*]
TZ = US/Eastern
```

The second example sets the timezone of events from sources in the path `/mnt/ca/...` with the US/Pacific timezone.

```
[source::/mnt/ca/...]
TZ = US/Pacific
```

zoneinfo (TZ) database

The zoneinfo database is a publicly maintained database of timezone values.

- UNIX versions of Splunk rely on a TZ database included with the UNIX distribution you're installing on. Most UNIX distributions store the database in the directory: `/usr/share/zoneinfo`.
- Solaris versions of Splunk store TZ information in this directory: `/usr/share/lib/zoneinfo`.
- Windows versions of Splunk ship with a copy of the TZ database.

Refer to the zoneinfo (TZ) database for values you can set as `TZ =` in `props.conf`.

Tune timestamp recognition for better indexing performance

Tune timestamp recognition for better indexing performance

Tune Splunk's timestamp extraction by editing props.conf. Adjust how far Splunk's timestamp processor looks into events, or turn off the timestamp processor to make indexing faster.

Note: Use `$SPLUNK_HOME/etc/system/README/props.conf.example` as an example, or create your own `props.conf`. Make any configuration changes to a copy of `props.conf` in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For more information on configuration files in general, see how configuration files work.

Adjust timestamp lookahead

Timestamp lookahead determines how far (how many characters) into an event the timestamp processor looks for a timestamp. Adjust how far the timestamp processor looks by setting a value (the number of characters) for the `MAX_TIMESTAMP_LOOKAHEAD =` key in any timestamp stanza.

Note: You can set `MAX_TIMESTAMP_LOOKAHEAD =` to different values for each timestamp stanza.

The default number of characters that the timestamp processor looks into an event is 150. Set `MAX_TIMESTAMP_LOOKAHEAD =` to a lower value to speed up how fast events are indexed. You should do this if your timestamps occur in the first part of your event.

If your events are indexed in real time, increase Splunk's overall indexing performance by turning off timestamp lookahead (set `MAX_TIMESTAMP_LOOKAHEAD = 0`). This causes Splunk to not look into event's for a timestamp, and sets an event's timestamp to be its indexing time (using current system time).

Example:

This example tells the timestamp processor to look 20 characters into events from source `foo`.

```
[source::foo]
MAX_TIMESTAMP_LOOKAHEAD = 20
...
```

Turn off the timestamp processor

Turn off the timestamp processor entirely to significantly improve indexing performance. Turn off timestamp processing for events matching a host, source, sourcetype specified by a timestamp stanza by adding a `DATETIME_CONFIG =` key to a stanza and setting the value to `NONE`. When timestamp processing is off, Splunk won't look for timestamps to extract from event data. Splunk will instead set an event's timestamp to be its indexing time (using current system time).

Example:

This example turns off timestamp extraction for events that come from the source `foo`.

```
[source::foo]
DATETIME_CONFIG = NONE
...
```

Configure indexed field extraction

About default fields (host, source, sourcetype, and more)

About default fields (host, source, sourcetype, and more)

If you've read the "Configure event processing" chapter in this manual, you know that Splunk automatically extracts a number of default fields for each event it processes prior to indexing. These default fields include `index`, which identifies the index in which the related event is located, `linecount`, which describes the number of lines the related event contains, and `timestamp`, which describes the point in time at which an event occurred. (As discussed in the "Configure event timestamping" chapter, you have a number of options when it comes to changing the manner in which sets of events are timestamped.)

Note: For a complete list of the default fields that Splunk identifies for each event prior to indexing, see "Use default fields" in the User manual.

This chapter focuses mainly on three important default fields: `host`, `source`, and `sourcetype`. Splunk identifies `host`, `source`, and `sourcetype` values for each event it processes. This chapter explains how Splunk does this, and shows you how you can override the automatic assignment of `host` and `sourcetype` values for events when it is necessary to do so.

This chapter also shows you how to have Splunk extract additional, custom fields at index time. It's important to note that this practice is strongly discouraged, however. Adding to the list of indexed fields can negatively impact indexing and search speed. In addition, it may require you to reindex your entire dataset in order to have those fields show up for previously indexed events. It's best to extract fields at search time whenever possible. For more information, see "Index time versus search time" in this manual.

For more information about index-time field extraction, see "Configure index-time field extraction" in this manual.

Defining host, source, and sourcetype

The `host`, `source`, and `sourcetype` fields are defined as follows:

- **`host`** - An event's `host` value is typically the hostname, IP address, or fully qualified domain name of the network host from which the event originated. The host value enables you to easily locate data originating from a specific device. For an overview of the methods Splunk provides for the override of automatic host assignment, see the "Host field overview" in this topic in this manual.
- **`source`** - The `source` of an event is the name of the file, stream, or other input from which the event originates. For data monitored from files and directories, the value of source is the full path, such as `/archive/server1/var/log/messages.0` or `/var/log/`. The value of source for network-based data sources is the protocol and port, such as UDP:514.
- **`sourcetype`** - The source type of an event is the format of the data input from which it originates, such as `access_combined` or `cisco_syslog`. For an overview of how Splunk sets the source type value and the ways you can override automatic sourcetyping, see the "Override automatic source type assignment" topic in this manual.

Under what conditions should you override host and sourcetype assignment?

Much of the time, Splunk can automatically identify host and sourcetype values that are both correct *and* useful. But situations do come up that require you to intervene in this process and provide override values.

You may want to change your default `host` assignment when:

- you are bulk-loading archive data that was originally generated from a different host and you want those events to have that host value.
- your data is actually being forwarded from a different host (the forwarder will be the host unless you specify otherwise).
- you are working with a centralized log server environment, which means that all of the data received from that server will have the same host even though it originated elsewhere.

You may want to change your default `sourcetype` assignment when:

- you want to give all event data coming through a particular input or from a specific source the same source type, for tracking purposes.
- you want to apply source types to specific events coming through a particular input, such as events that originate from a discrete group of hosts, or even events that are associated with a particular IP address or userid.

There are also steps you can take to expand the range of source types that Splunk automatically recognizes, or to simply rename source types. See the "Source type field overview" section, below, for more information.

About hosts

An event's `host` field value is the name of the physical device from which the event originates. Because it is a default field, which means that Splunk assigns it to every event it indexes, you use it to search for all events that have been generated by a particular host.

The `host` value can be an IP address, device hostname, or a fully qualified domain name, depending on whether the event was received through a file input, network input, or the computer hosting the instance of Splunk.

How Splunk assigns the host value

If no other host rules are specified for a source, Splunk assigns `host` a default value that applies to all data coming from inputs on a given Splunk server. The default host value is the hostname or IP address of the network host. When Splunk is running on the server where the event occurred (which is the most common case) this is correct and no manual intervention is required.

For more information, see "Set a default host for a Splunk server" in this manual.

Set a default host for a file or directory input

If you are running Splunk on a central log archive, or you are working with files forwarded from other hosts in your environment, you may need to override the default host assignment for events coming from particular inputs.

There are two methods for assigning a host value to data received through a particular input. You can define a static host value for all data coming through a specific input, or you can have Splunk dynamically assign a host value to a portion of the path or filename of the source. The latter method can be helpful when you have a directory structure that segregates each host's log archive in a different subdirectory.

For more information, see "Set a default host for a file or directory input" in this manual.

Override default host values based on event data

You may have a situation that requires you to override host values based on event data. For example, if you work in a centralized log server environment, you may have several host servers that feed into that main log server. The central log server is called the *reporting host*. The system where the event occurred is called the *originating host* (or just the host). In these cases you need to define rules that override the automatic host assignments for events received from that centralized log host and replace them with distinct originating host values.

For more information, see "Override default host values based on event data" in this manual.

Tag host values

Tag host values to aid in the execution of robust searches. Tags enable you to cluster groups of hosts into useful, searchable categories.

For more information, see "About tags and aliases" in the Knowledge Manager manual.

About source types

Any common data input format can be a **source type**. Most source types are log formats. For example, the list of common source types that Splunk automatically recognizes includes:

- **access_combined**, for NCSA combined format HTTP Web server logs.
- **apache_error**, for standard Apache Web server error logs.
- **cisco_syslog**, for the standard syslog produced by Cisco network devices including PIX firewalls, routers, ACS, etc., usually via remote syslog to a central log host.
- **websphere_core**, which is a core file export from WebSphere.

Note: For a longer list of source types that Splunk automatically recognizes, see "List of pretrained sourcetypes" in this manual.

`sourcetype` is the name of the source type field. You can use the `sourcetype` field to find similar types of data from any source type. For example, you could search

`sourcetype=weblogic_stdout` to find all of your WebLogic server events even when WebLogic is logging from more than one domain (or "host," in Splunk terms).

Source vs source type

The source is the name of the file, stream, or other input from which a particular event originates. For data monitored from files and directories, the value of `source` is the full path, such as `/archive/server1/var/log/messages.0` or `/var/log/`. The value of source for network-based data sources is the protocol and port, such as `UDP:514`.

Events with the same source type can come from different sources. For example, say you're monitoring `source=/var/log/messages` and receiving direct syslog input from `udp:514`. If you search `sourcetype=linux_syslog`, Splunk will return events from both of those sources.

Methods Splunk uses for source type assignation and their precedence

Splunk employs a variety of methods to assign source types to event data at index time. As it processes event data, Splunk steps through these methods in a defined order of precedence. It starts with hardcoded source type configurations in `inputs.conf` and `props.conf`, moves on to rule-based source type association, and then works through methods like automatic source type recognition and automatic source type learning. This range of methods enables you to configure how Splunk applies source type values to specific kinds of events, while letting Splunk assign source type values to the remaining events automatically.

The following list discusses these methods in the order that Splunk typically uses them to assign source types to event data at index time:

1. Explicit source type specification based on the data input, as configured in `inputs.conf` stanzas:

```
[monitor://$PATH]
sourcetype=$SOURCETYPE
```

2. Explicit source type specification based on the data source, as configured in `props.conf` stanzas:

```
[source::$SOURCE]
sourcetype=$SOURCETYPE
```

3. Rule-based source type recognition:

Enables Splunk to match incoming data to source types using classification rules specified in `rule::` stanzas in `props.conf`.

```
[rule::$RULE_NAME]
sourcetype=$SOURCETYPE
MORE_THAN_[0-100] = $REGEX
LESS_THAN_[0-100] = $REGEX
```

For information about setting up or removing source type recognition rules, see "Configure rule-based source type recognition" in this manual.

4. Automatic source type matching:

Splunk uses automatic source type recognition to match similar-looking files and, through that, assign a source type. It calculates signatures for patterns in the first few thousand lines of any file or stream of network input. These signatures identify things like repeating word patterns, punctuation patterns, line length, and so on. When Splunk calculates a signature, it compares it to previously seen signatures. If the signature appears to be a radically new pattern, Splunk creates a new source type for the pattern.

Note: At this stage in the source type assignation process, Splunk just matches incoming data with source types that it has learned previously. It doesn't create new source types for unique signatures until the final stage of source typing (step 6, below).

See "List of pretrained source types" in this manual for a list of the source types that Splunk can recognize out of the box. See "Train Splunk's source type autoclassifier" for more information about expanding the list of source types that Splunk can assign through automatic source type recognition.

5. Delayed rule-based source type association:

This works like rule-based associations (see above), except you create a `delayedrule::` stanza in `props.conf`. This is a useful "catch-all" for source types, in case Splunk missed any with intelligent matching (see above).

A good use of delayed rule associations is for generic versions of very specific source types that are defined earlier with `rule::` in step 3, above. For example, you could use `rule::` to catch event data with specific syslog source types, such as "sendmail syslog" or "cisco syslog" and then have `delayedrule::` apply the generic "syslog" source type to the remaining syslog event data.

```
[delayedrule::$RULE_NAME]
sourcetype=$SOURCETYPE
MORE_THAN_[0-100] = $REGEX
LESS_THAN_[0-100] = $REGEX
```

For more information about settting up or removing delayed rules for source type recognition, see "Configure rule-based source type recognition" in this manual.

6. Automatic source type learning:

If Splunk is unable to assign a source type for the event using the preceding six methods, it creates a new source type for the event signature (see step 4, above). Splunk stores learned pattern information in sourcetypes.conf.

Set a default host for a Splunk server

Set a default host for a Splunk server

An event's `host` value is the IP address, host name, or fully qualified domain name of the physical device on the network from which the event originates. Because Splunk assigns a `host` value at index time for every event it indexes, host value searches enable you to easily find data originating from a specific device.

Default host assignment

If you have not specified other host rules for a source (using the information in this and subsequent topics in this chapter), the default host value for an event is typically the hostname, IP address, or fully qualified domain name of the network host from which the event originated. When the event originates from the server on which Splunk is running (which is the most common case) the host assignment is correct, and there's no need for you to change anything. However, if you data is being forwarded from a different host, or if you're bulk-loading archive data, you may want to change the default host value for that data.

To set the default value of the host field, you can use Splunk Manager, or edit inputs.conf.

Set the default host value using Manager

Use Manager to set the default host value for a server:

1. In Splunk Web, click on the **Manager** link in the upper right-hand corner of the screen.

2. In Manager, click **System settings** under **System configurations**.

3. On the System settings page, click **General settings**.

4. On the General settings page, scroll down to the **Index settings** section and change the **Default host name**.

5. Save your changes.

This sets the value of the host field for all events that have not received another host name.

Set the default host value using inputs.conf

This host assignment is set in inputs.conf during Splunk installation. Modify the host entry by editing `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. (We recommend using the latter directory if you want to make it easy to transfer your data customizations to other search servers).

This is the format of the host assignment in `inputs.conf`:

```
host = <string>
```

- Set `<string>` to your chosen default host value. `<string>` defaults to the IP address or domain name of the host where the data originated.
- This is a shortcut for `MetaData:Host = <string>`. It sets the host of events from this input to be the specified string. Splunk automatically prepends `host::` to the value when this shortcut is used.

Restart Splunk to enable any changes you have made to `inputs.conf`.

Override the default host value for data received from a specific input

If you are running Splunk on a central log archive, or you are working with files copied from other hosts in the environment, you may want to override the default host assignment for a particular input on a static or dynamic basis.

For more information, see "Set a default host value for an input" in this manual.

Override the default host value using event data

If you have a centralized log host sending events to Splunk, many servers may be involved. The central log server is called the *reporting host*. The system where the event occurred is called the *originating host* (or just the host). In this case you need to define rules that set the host field value based on the information in the events themselves.

For more information, see "Override default host values based on event data" in this manual.

Set a default host for a file or directory input

Set a default host for a file or directory input

In certain situations you may want to explicitly set a host value for all data coming in to Splunk through a particular file or directory input. You can set the host statically or dynamically.

- To **statically** set the host means you're setting the same host for every event that comes to Splunk through a designated file or directory input.
- If you **dynamically** set the host value, Splunk extracts the host name from a portion of the source input using a regex or segment of the source's full directory path.

You can also assign host values to events coming through a particular file or directory input based on their source or sourcetype values (as well as other kinds of information). For more information, see "Overriding default host assignments based on event data," in this manual.

Note: Splunk currently does not enable the setting of default host values for event data received through TCP, UDP, or **scripted inputs**.

Statically setting the default host value for a file or directory input

This method applies a single default host value to each event received through a specific file or directory input.

Note:A static host value assignment only impacts new data coming in through the input with which it's associated. You cannot assign a default host value to data that has already been processed, split into events, and indexed.

If you need to assign a host value to data that's already been indexed, you need to tag the host value instead.

Via Splunk Web

You can statically define a host for a file or directory input whenever you add a new input of that type through the "Data inputs" page of Splunk Web's Manager interface:

1. In Splunk Web, click on the **Manager** link in the upper right-hand corner of the screen.

2. In Manager, click **Data inputs** under **System configurations**.

3. On the Data inputs page, select **Files & Directories** to go to the list page for that input type.

4. On the Files & directories page, you can either click the name of an input that you want to update, or click **New** to create a new file or directory input.

5. Once you're on the detail page for the file or directory input, select the *Constant value* option from the **Set host** dropdown.

6. Enter the static host value for the input in the **Host field value** field.

7. Save your changes.

For more information about inputs and input types, see "What Splunk can monitor" in the Admin guide.

Via configuration files

Edit inputs.conf to specify a host value for a monitored file or directory input. Include a `host =` attribute within the appropriate stanza.

```
[monitor://<path>]
host = $YOUR_HOST
```

Edit `inputs.conf` in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For more information on configuration files in general, see "About configuration files" in the Admin manual.

For more information about inputs and input types, see "What Splunk can monitor" in the Admin manual.

Example of static host value assignment for an input

This example covers any events coming in from `/var/log/httpd`. Any events coming from this input will receive a `host` value of `webhead-1`.

```
[monitor:///var/log/httpd]
host = webhead-1
```

Dynamically setting the default host value for a file or directory input

Use this method if you want to dynamically extract the host value for a file or directory input, either from a segment of the source input path, or from a regular expression. For example, if you want to index an archived directory and the name of each file in the directory contains relevant host information, you can use Splunk to extract this information and assign it to the host field.

Via SplunkWeb

Start by following the steps for setting up a static host assignment via Splunk Web, above. However, when you get to the **Set host** dropdown list on the input details page for a file or directory input, choose one of the following two values:

- *Regex on path* - Choose this option if you want to extract the host name via a regular expression. Enter the regex for the host you want to extract in the **Regular expression** field.
- *Segment in path* - Choose this option if you want to extract the host name from a segment in your data source's path. Enter the segment number in the **Segment #** field. For example, if the path to the source is `/var/log/[host server name]` and you want the third segment (the host server name) to be the host value, enter *3* into the **Segment #** field.

Note: For a primer on regular expression syntax and usage, see Regular-Expressions.info. You can test regexes by using them in searches with the rex search command. Splunk also maintains a list of useful third-party tools for writing and testing regular expressions.

Via configuration files

You can set up dynamic host extraction rules when you are configuring `inputs.conf`. Edit `inputs.conf` in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For more information on configuration files in general, see "About configuration files" in the Admin manual.

Add `host_regex = <regular expression>` to override the host field with a value extracted using a regular expression.

```
[monitor://<path>]
host_regex = $YOUR_REGEX
```

The regular expression extracts the `host` value from the filename of each input. The first capturing group of the regex is used as the host.

Note: If the regex fails to match, the default `host =` attribute is set as the host.

Important: For a primer on regular expression syntax and usage, see Regular-Expressions.info. You can test regexes by using them in searches with the rex search command. Splunk also maintains a list of useful third-party tools for writing and testing regular expressions.

```
host_segment = <integer>
```

Define a `host_segment` instead of a `host_regex` if you want to override the host field with a value extracted using a segment of the data source path. For example, if the path to the source is

`/var/log/[host server name]` and you want the third segment (the host server name) to be the host value, your input stanza would look like:

```
[monitor://var/log/]
host_segment = 3
```

Note: If the `<integer>` value is not an integer, or is less than 1, Splunk sets the default `host =` attribute as the host.

Note: You cannot simultaneously specify a `host_regex` and `host_segment`.

Examples of dynamic host assignment for an input

This example uses regex on the file path to set the host:

```
[monitor://var/log]
host_regex = /var/log/(\w+)
```

With that regex, all events from `/var/log/foo.log` are given the a `host` value of *foo*.

This example uses the segment of the data source filepath to set the host:

```
[monitor://apache/logs/]
host_segment = 3
```

It sets the `host` value to the third segment in the path `apache/logs`.

Override default host values based on event data

Override default host values based on event data

Splunk assigns default host names to your events based on data in those events. This topic shows you how to override specific default host assignments when these default assignments are incorrect.

Configuration

To set up host value overrides based on event data, you need to edit transforms.conf and props.conf. Edit these files in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For more information about configuration files in general, see "About configuration files" in this manual.

Edits to transforms.conf

Add your custom stanza to `$SPLUNK_HOME/etc/system/local/transforms.conf`. Configure your stanza as follows:

```
[$UNIQUE_STANZA_NAME]
DEST_KEY = MetaData:Host
REGEX = $YOUR_REGEX
FORMAT = host::$1
```

Fill in the stanza name and the regex fields with the correct values for your data.

Leave `DEST_KEY = MetaData:Host` to write a value to the `host::` field. `FORMAT = host::$1` writes the `REGEX` value into the `host::` field.

Note: Name your stanza with a unique identifier (so it is not confused with an existing stanza in `$SPLUNK_HOME/etc/system/default/transforms.conf`).

Edits to props.conf

Create a stanza in `$SPLUNK_HOME/etc/system/local/props.conf` to map the `transforms.conf` regex to the source type in `props.conf`.

```
[<spec>]
TRANSFORMS-$name=$UNIQUE_STANZA_NAME
```

- `<spec>` can be:
  - <sourcetype>, the sourcetype of an event
  - host::<host>, where <host> is the host for an event
  - source::<source>, where <source> is the source for an event
- `$name` is whatever unique identifier you want to give to your transform.
- `$UNIQUE_STANZA_NAME` must match the stanza name of the transform you just created in `transforms.conf`.

Note: Optionally add any other valid attribute/value pairs from props.conf when defining your stanza. This assigns the attributes to the `<spec>` you have set. For example, if you have custom line-breaking rules to set for the same <spec>, append those attributes to your stanza.

Example

Here is a set of events from the `houseness.log` file. They contain the host in the third position.

```
41602046:53 accepted fflanda
41602050:29 accepted rhallen
41602052:17 accepted fflanda
```

Create a regex to extract the host value and add it to a new stanza in `$SPLUNK_HOME/etc/system/local/transforms.conf`:

```
[houseness]
DEST_KEY = MetaData:Host
REGEX = \s(\w*)$
FORMAT = host::$1
```

Now, link your `transforms.conf` stanza to `$SPLUNK_HOME/etc/system/local/props.conf` so your transforms are called. Optionally add any additional attribute/value pairs from props.conf as needed.

The transform above works with the following stanza in `props.conf`:

```
[source::.../houseness.log]
TRANSFORMS-rhallen=houseness
```

```
SHOULD_LINEMERGE = false
```

The above stanza has the additional attribute/value pair `SHOULD_LINEMERGE = false`. This specifies that Splunk should create new events at a newline.

Note: The additional `-rhallen` in the attribute `TRANSFORMS-rhallen` serves to differentiate this transform from other transforms.

The events now appear in SplunkWeb as the following:

Handle incorrectly-assigned host values

Handle incorrectly-assigned host values

At some point, you may discover that the host value for some of your events might be set incorrectly for some reason. For example, you might be scraping some Web proxy logs into a directory directly on your Splunk server and add that directory as an input to Splunk without remembering to override the value of the host field, causing all those events to think their original host value is the same as your Splunk host.

If something like that happens, here are your options, in order of complexity:

- Delete and reindex the entire index
- Use a search to delete the specific events that have the incorrect host value and reindex those events
- Tag the incorrect host values with a tag, and search with that
- Set up a static field lookup to look up the host, map it in the lookup file to a new field name, and use the new name in searches
- Alias the host field to a new field (such as `temp_host`), set up a static field lookup to look up the correct host name using the name `temp_host`, then have the lookup overwrite the original `host` with the new lookup value (using the `OUTPUT` option when defining the lookup)

Of these options, the last option will look the nicest if you can't delete and reindex the data, but deleting and reindexing the data will always give the best performance.

Override automatic source type assignment

Override automatic source type assignment

You can override automatic source type assignment for event data that comes from specific inputs, or which has a particular source.

Note: While source type assignment by input seems like a simple way to handle things, it isn't very granular--when you use it Splunk gives *all* event data from an input the same source type, even if they actually have different sources and hosts. If you want to bypass automatic source type assignment in a more targeted manner, arrange for Splunk to assign source types according to the event data source.

Override automated source type matching for an input

Use these instructions to override automated source type assignation and explicitly assign a single source type value to data coming from a specific input such as `/var/log/`.

Note: This only affects new data coming in *after* the override is set up. To correct the source types of events that have already been indexed, create a tag for the source type instead.

Through Splunk Web

When you define a data input in Manager, you can set a sourcetype value that Splunk applies to all incoming data from that input. Manager gives you the option of picking a sourcetype value from a list or entering a unique sourcetype value of your own.

To select a sourcetype value for an input, click the **Manager** link to go to the Splunk Manager page, select **Data inputs** and then drill down to the details page of the input for which you want to define a sourcetype.

Pick a sourcetype value from a list for an input

If the data from a particular input belongs to one of Splunk's pretrained source types, you can choose the sourcetype value that Splunk would otherwise assign automatically from a drop down list. For a description of Splunk's pretrained source types, see the reference list of pretrained sourcetypes in the "List of pretrained sourcetypes" topic, in this manual.

On the details page for the input that you're defining a source type for, select *From list* from **Set source type**. Then choose a pretrained sourcetype from the **Select source type from list** dropdown list.

Save your input settings. After that point, Splunk will assign the sourcetype that you've selected to all events it indexes for that input.

Manually enter a sourcetype value for an input

You can manually enter a sourcetype value for data that Splunk receives from a particular input.

On the details page for the input that you're defining a source type for, select *Manual* from the **Set source type** list, and then enter a source type in **Source type**.

Save your input settings. After that point, Splunk will assign the sourcetype that you've specified to all events it indexes for that input.

Through configuration files

When you configure inputs in inputs.conf, you can set a sourcetype as well. Edit `inputs.conf` in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For more information on configuration files in general, see "About configuration files" in this manual.

Note: This only impacts new data coming in *after* your configuration change. If you want to correct the source types of events that have already been indexed, create a tag for the source type instead.

Include a `sourcetype =` attribute within the appropriate stanza in `inputs.conf`:

```
[tcp://:9995]
connection_host = dns
sourcetype = log4j
source = tcp:9995
```

The above example sets any events coming from your TCP input on port 9995 as `sourcetype=log4j`.

Override automatic source type matching for a source

Use these instructions to override automated source type assignation and explicitly assign a single source type value to data coming from a specific source.

Use these instructions to assign a source type based on a source through props.conf. Edit `props.conf` in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For more information on configuration files in general, see "About configuration files".

Important: If you are forwarding data to one or more receivers, and want to set up an override of automatic source type matching for a specific source, you must set it up on the `props.conf` file for the forwarder. If you set it up on the receiver, the override will not take effect.

Note: This only impacts new data coming in *after* your configuration change. If you want to correct the source types of events that have already been indexed, create a tag for the source type instead.

Learn more about props.conf.

Through configuration files

Add a stanza for your source in `props.conf`. In the stanza, identify the source path, using regex syntax for flexibility if necessary. Then identify the source type by including a `sourcetype =` attribute:

```
[source::.../var/log/anaconda.log(.\d+)?]
sourcetype = anaconda
```

This example sets any events from sources containing the string `/var/log/anaconda.log` followed by any number of numeric characters to `sourcetype = anaconda`.

Splunk recommends that your stanza source path regexes (such as `[source::.../web/....log]`) be as *specific* as possible. It is HIGHLY recommended that you not have the regex end in "...". For example, don't do this:

```
[source::/home/fflanda/...]
sourcetype = mytype
```

This is dangerous. The above example tells Splunk to process gzip files in `/home/fflanda` as mytype files rather than gzip files.

It would be much better to write:

```
[source::/home/fflanda/....log(.\d+)?]
sourcetype = mytype
```

Note: For a primer on regular expression syntax and usage, see Regular-Expressions.info. You can test regexes by using them in searches with the rex search command. Splunk also maintains a list of useful third-party tools for writing and testing regular expressions.

Advanced source type overrides

Advanced source type overrides

This topic shows you how to configure Splunk to override sourcetypes on a per-event basis. It includes an example that demonstrates the use of `transforms.conf` in tandem with `props.conf` to override sourcetypes for events associated with a specific host, and goes on to show how you can do this for event data coming from a particular input or source.

For more information about performing basic source type overrides for event data that comes from specific inputs, or which has a particular source, see "Override automatic source type assignment" in this manual.

Configuration

To do this you'll set up two stanzas, one in `transforms.conf`, and another in `props.conf`. Edit these files in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`.

transforms.conf

The `transforms.conf` stanza should follow this format:

```
[<unique_stanza_name>]
REGEX = <your_regex>
FORMAT = sourcetype::<your_custom_sourcetype_value>
DEST_KEY = MetaData:Sourcetype
```

- `<unique_stanza_name>` should reflect that it involves a sourcetype. You'll use this name later in the `props.conf` stanza.

- <your_regex> is a regular expression that identifies the events that you want to apply a custom sourcetype to (such as events carrying a particular hostname or other field value).
- <your_custom_sourcetype_value> is the sourcetype value that you want to apply to the regex-selected events.

Note: For a primer on regular expression syntax and usage, see Regular-Expressions.info. You can test regexes by using them in searches with the rex search command. Splunk also maintains a list of useful third-party tools for writing and testing regular expressions.

props.conf

Next you create a stanza in props.conf that references the transforms.conf stanza, as follows.

```
[<spec>]
TRANSFORMS-<class> = <unique_stanza_name>
```

- <spec> can be:
  - <sourcetype>, the sourcetype value of an event.
  - host::<host>, where <host> is the host value for an event.
  - source::<source>, where <source> is the source value for an event.
- <class> is any name that you want to give to your stanza to identify it. In this case you might just use "sourcetype" to identify it as a sourcetype.
- <unique_stanza_name>is the name of your stanza from transforms.conf.

Example - Sourcetyping events originating from different hosts, indexed from a single input

Let's say that you have a shared UDP input, UDP514. Your Splunk instance indexes a wide range of data from a number of hosts through this input. You've found that you need to apply a particular sourcetype--which, for the purposes of this example we'll call "my_log"--to data originating from three specific hosts (host1, host2, and host3) that reaches Splunk through UDP514.

To start, you can use the regex that Splunk typically uses to extract the host field for syslog events. You can find it in system/default/transforms.conf:

```
[syslog-host]
REGEX = :\d\d\s+(?:\d+\s+|(?:user|daemon|local.?)\.\w+\s+)*\[?(\w[\w\.\-]{2,})\]?\s
FORMAT = host::$1
DEST_KEY = MetaData:Host
```

You can easily modify this regex to only match events from the hostnames you want (for the purposes of this example we're calling them *host1*, *host2*, and *host3*):

```
REGEX
= :\d\d\s+(?:\d+\s+|(?:user|daemon|local.?)\.\w+\s+)*\[?(host1|host2|host3)[\w\
```

Now you can use that modified regex in a transform that applies the my_log sourcetype to events that come from those three hosts:

```
[set_sourcetype_my_log_for_some_hosts]
REGEX = :\d\d\s+(?:\d+\s+|(?:user|daemon|local.?)\.\w+\s+)*\[?(host1|host2|host3)[\w\.\-]*\]?\s
FORMAT = sourcetype::my_log
```

```
DEST_KEY = MetaData:Sourcetype
```

And then you can refer that transform to `props.conf`, which in this case is used to identify the specific input that carries the events that you want to sourcetype:

```
[source::udp:514]
TRANSFORMS-changesourcetype = set_sourcetype_my_log_for_some_hosts
```

Note: The above configuration applies the sourcetype to your specified event data by host at index-time. Make yourself aware of the implications of making changes to index-time processing.

Configure rule-based source type recognition

Configure rule-based source type recognition

Configure rule-based source type recognition to expand the range of source types that Splunk recognizes. Splunk automatically assigns rule-based source types based on regular expressions you specify in `props.conf`.

You can create two kinds of rules in `props.conf`: *rules* and *delayed rules*. The only difference between the two is the point at which Splunk checks them during the source typing process. As it processes each string of event data, Splunk uses several methods to determine source types:

- After checking for explicit source type definitions based on the event data input or source, Splunk looks at the `rule::` stanzas defined in `props.conf` and tries to match source types to the event data based on the classification rules specified in those stanzas.
- If Splunk is unable to find a matching source type using the available `rule::` stanzas, it tries to use automatic source type matching, where it tries to identify patterns similar to source types it has learned in the past.
- When that method fails, Splunk then checks the `delayedrule::` stanzas in `props.conf`, and tries to match the event data to source types using the rules in *those* stanzas.

You can set up your system so that `rule::` stanzas contain classification rules for specialized source types, while `delayedrule::` stanzas contain classification rules for generic source types. This way the the generic source types are applied to broad ranges of events that haven't qualified for more specialized source types. For example, you could use `rule::` stanzas to catch event data with specific syslog source types, such as `sendmail_syslog` or `cisco_syslog` and then have a `delayedrule::` stanza apply the generic `syslog` source type to remaining syslog event data.

Configuration

To set source typing rules, edit `props.conf` in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For more information on configuration files in general, see "About configuration files" in this manual.

Create a rule by adding a `rule::` or `delayedrule::` stanza to `props.conf`. Provide a name for the rule in the stanza header, and declare the source type name in the body of the stanza. After the source type declaration, list the the source type assignation rules. These rules use one or more

MORE_THAN and LESS_THAN statements to find patterns in the event data that fit given regular expressions by specific percentages.

Note: You can specify any number of MORE_THAN and LESS_THAN statements in a source typing rule stanza. All of the statements must match a percentage of event data lines before those lines can be assigned the source type in question. For example, you could define a rule that assigns a specific source type value to event data where more than 10% match one regular expression and less than 10% match another regular expression.

Add the following to props.conf:

```
[rule::$RULE_NAME] OR [delayedrule::$RULE_NAME]
sourcetype=$SOURCETYPE
MORE_THAN_[0-100] = $REGEX
LESS_THAN_[0-100] = $REGEX
```

The MORE_THAN and LESS_THAN numerical values refer the percentage of lines that contain the string specified by the regular expression. To match, a rule can be either MORE_THAN or LESS_THAN those percentages.

Note: For a primer on regular expression syntax and usage, see Regular-Expressions.info. You can test regexes by using them in searches with the rex search command. Splunk also maintains a list of useful third-party tools for writing and testing regular expressions.

Examples

The following examples come from $SPLUNK_HOME/etc/system/default/props.conf.

Postfix syslog files

```
# postfix_syslog sourcetype rule
[rule::postfix_syslog]
sourcetype = postfix_syslog
# If 80% of lines match this regex, then it must be this type
MORE_THAN_80=^\w{3} +\d+ \d\d:\d\d:\d\d .* postfix(/\w+)?\[\d+\]:
```

Delayed rule for breakable text

```
# breaks text on ascii art and blank lines if more than 10% of lines have
# ascii art or blank lines, and less than 10% have timestamps
[delayedrule::breakable_text]
sourcetype = breakable_text
MORE_THAN_10 = (^(?:---|===|\*\*\*|___|=+=))|^\s*$
LESS_THAN_10 = [: ][012]?[0-9]:[0-5][0-9]
```

Rename source types

Rename source types

When configuring a source type in props.conf, you can rename the source type; several source types can share one name.

To rename the source type, add the following into your source type stanza:

```
rename = <string>
```

After renaming, you can search for the source type with:

```
sourcetype=<string>
```

Since one string can be used for multiple source types, to search for the original source type before renaming, use:

```
_sourcetype=<sourcetype>
```

Train Splunk's source type autoclassifier

Train Splunk's source type autoclassifier

Use these instructions to train Splunk to recognize a new source type, or give it new samples to better recognize a pre-trained source type. Autoclassification training enables Splunk to classify future event data with similar patterns as a specific source type. This can be useful when Splunk is indexing directories that contains data with a mix of source types (such as `/var/log`). Splunk ships "pre-trained," with the ability to assign `sourcetype=syslog` to most syslog files.

Note: Keep in mind that source type autoclassification training applies to future event data, not event data that has already been indexed.

You can also bypass auto-classification in favor of hardcoded configurations, and just override a sourcetype for an input, or override a sourcetype for a source. Or configure rule-based source type recognition.

You can also anonymize your file using Splunk's built in anonymizer utility.

If Splunk fails to recognize a common format, or applies an incorrect source type value, you should report the problem to Splunk support and send us a sample file.

via the CLI

Here's what you enter to train source types through the CLI:

```
# splunk train sourcetype $FILE_NAME $SOURCETYPE_NAME
```

Fill in `$FILE_NAME` with the entire path to your file. `$SOURCETYPE_NAME` is the custom source type you wish to create.

It's usually a good idea to train on a few different samples for any new source type so that Splunk learns how varied a source type can be.

List of pretrained source types

List of pretrained source types

Splunk ships pre-trained to recognize many different source types. A number of source types are automatically recognized, tagged and parsed appropriately. Splunk also contains a significant number of pre-trained source types that are not automatically recognized but can be assigned via Splunk Web or inputs.conf.

It's a good idea to use a pre-trained source type if it matches your data, as Splunk contains optimized indexing properties for pre-trained source types. However, if your data does not fit with any pre-trained source types, Splunk can index virtually any format of data without custom properties.

Learn more about source types and how they work.

Automatically recognized source types

Source type name	Origin	Exampl
access_combined	NCSA combined format http web server logs (can be generated by apache or other web servers)	`10.1.1.43 - webdev [08/Aug/2005:13:18:16 "-" "check_http/1.10 (nagios-plugins 1.4`
access_combined_wcookie	NCSA combined format http web server logs (can be generated by apache or other web servers), with cookie field added at end	`"66.249.66.102.1124471045570513" 59.92.1 -0700] "GET /themes/splunk_com/images/lo "http://www.splunk.org/index.php/docs" " en-US; rv:1.7.8) Gecko/20050524 Fedora/1 "61.3.110.148.1124404439914689"`
access_common	NCSA common format http web server logs (can be generated by apache or other web servers)	`10.1.1.140 - - [16/May/2005:15:01:52 -07 /themes/ComBeta/images/bullet.png HTTP/1`
apache_error	Standard Apache web server error log	`[Sun Aug 7 12:17:35 2005] [error] [clien /home/reba/public_html/images/bullet_ima`
asterisk_cdr	Standard Asterisk IP PBX call detail record	`"","5106435249","1234","default","""Jame Jesse""<5106435249>","SIP/5249-1ce3","", 15:19:25","2005-05-26 15:19:25","2005-05 15:19:42",17,17,"ANSWERED","DOCUMENTATIO`
asterisk_event		`Aug 24 14:08:05 asterisk[14287]: Manager`

	Standard Asterisk event log (management events)	
asterisk_messages	Standard Asterisk messages log (errors and warnings)	`Aug 24 14:48:27 WARNING[14287]: Channel extension 's' in context 'default', but`
asterisk_queue	Standard Asterisk queue log	NONE|NONE|NONE|CONFIGRELOAD|
cisco_syslog	Standard Cisco syslog produced by all Cisco network devices including PIX firewalls, routers, ACS, etc., usually via remote syslog to a central log host	`Sep 14 10:51:11 stage-test.splunk.com Au Inbound TCP connection denied from IP_ad TCP_flags on interface int_name Inbound 144.1.10.222/9876 to 10.0.253.252/6161 f`
db2_diag	Standard IBM DB2 database administrative and error log	`2005-07-01-14.08.15.304000-420 I27231H32 4760 PROC : db2fmp.exe INSTANCE: DB2 NOD Table Maintenance, db2HmonEvalStats, pro evaluation has finished on database TRAD`
exim_main	Exim MTA mainlog	`2005-08-19 09:02:43 1E69KN-0001u6-8E => R=send_to_relay T=remote_smtp H=mail.int`
exim_reject	Exim reject log	`2005-08-08 12:24:57 SMTP protocol violat sent without waiting for greeting): reje H=gate.int.splunk.com [10.2.1.254]`
linux_messages_syslog	Standard linux syslog (/var/log/messages on most platforms)	`Aug 19 10:04:28 db1 sshd(pam_unix)[15979 (uid=0)`
linux_secure	Linux securelog	`Aug 18 16:19:27 db1 sshd[29330]: Accepte from ::ffff:10.2.1.5 port 40892 ssh2`
log4j	Log4j standard output produced by any J2EE server using log4j	`2005-03-07 16:44:03,110 53223013 [PoolTh property...`
mysqld_error	Standard mysql error log	`050818 16:19:29 InnoDB: Started; log seq /usr/libexec/mysqld: ready for connectio '/var/lib/mysql/mysql.sock' port: 3306 S`
mysqld	Standard mysql query log; also matches mysql's	`53 Query SELECT xar_dd_itemid, xar_dd_pr xar_dynamic_data WHERE xar_dd_propid IN`

	binary log following conversion to text	
postfix_syslog	Standard Postfix MTA log reported via the Unix/Linux syslog facility	`Mar 1 00:01:43 avas postfix/smtpd[1822]: client=host76-117.pool80180.interbusines`
sendmail_syslog	Standard Sendmail MTA log reported via the Unix/Linux syslog facility	`Aug 6 04:03:32 nmrjl00 sendmail[5200]: q ctladdr=root (0/0), delay=00:00:01, xdel min=00026, relay=[101.0.0.1] [101.0.0.1] (v00F3HmX004301 Message accepted for del`
sugarcrm_log4php	Standard Sugarcrm activity log reported using the log4php utility	`Fri Aug 5 12:39:55 2005,244 [28666] FATA application list language file for the s default language(en_us)`
weblogic_stdout	Weblogic server log in the standard native BEA format	`####<Sep 26, 2005 7:27:24 PM MDT> <Warni <asiAdminServer> <ListenThread.Default> <HostName: 0.0.0.0, maps to multiple IP addresses:169.254.25.129,169.254.193.219`
websphere_activity	Websphere activity log, also often referred to as the service log	`---------- ComponentId: Application Server ProcessI ThreadName: Non-deferrable Alarm : 3 Sou com.ibm.ws.channel.framework.impl. WSCha MethodName: Manufacturer: IBM Product: W 6.0.1.0 o0510.18] ServerName: nd6Cell01\ 2005-07-01 13:04:55.187000000 UnitOfWork PrimaryMessage: CHFW0020I: The Transport Chain labeled SOAPAcceptorChain2 Extende ----------`
websphere_core	Corefile export from Websphere	`NULL---------- 0SECTION TITLE subcomponent dump routine 1TISIGINFO signal 0 received 1TIDATETIME 1TIFILENAME Javacore filename: /kmbcc/ja ---------- 0SECTION XHPI subcomponent dump routine 1XHTIME Tue Aug 2 10:19:24 20051XHSIGREC <unknown>. Processing terminated. 1XHFUL ca131-20031105 NULL`
websphere_trlog_syserr	Standard Websphere system error log in IBM's native tr log format	`[7/1/05 13:41:00:516 PDT] 000003ae Syste inbound.impl.HttpICLReadCallback.complet Code)) (truncated)`
websphere_trlog_sysout	Standard Websphere system out log in IBM's native trlog format;	`[7/1/05 13:44:28:172 PDT] 0000082d Syste TradeStreamerMDB: 100 Trade stock prices update Quote Price message count = 4400 messages (in seconds): min: -0.013 max:`

	similar to the log4j server log for Resin and Jboss, sampe format as the system error log but containing lower severity and informational events	`current price update is: Update Stock pr price = 21.50`
windows_snare_syslog	Standard windows event log reported through a 3rd party Intersect Alliance Snare agent to remote syslog on a Unix or Linuxserver	`0050818050818 Sep 14 10:49:46 stage-test MSWinEventLog 0 Security 3030 Day Aug 24 User Success Audit Test_Host Object Open Type: File Object Name: C:\Directory\sec Operation ID: {0,117792} Process ID: 924 Domain: FLAME Primary Logon ID: (0x0,0x8 Domain: - Client Logon ID: - Accesses SY ListDirectory) Privileges -Sep`

Pre-trained source types

This list contains both automatically recognized source types and pre-trained source types that are not automatically recognized.

Category	Source type(s)
Application servers	log4j, log4php, weblogic_stdout, websphere_activity, websphere_core, websphere_trlog
Databases	mysqld, mysqld_error, mysqld_bin
E-mail	exim_main, exim_reject, postfix_syslog, sendmail_syslog, procmail
Operating systems	linux_messages_syslog, linux_secure, linux_audit, linux_bootlog, anaconda, anaconda_syslog, osx_asl, osx_crashreporter, osx_crash_log, osx_install, osx_secure, osx_daily, osx_weekly, osx_monthly, osx_window_server, windows_snare_syslog, dmesg, ftp, ssl_error, syslog, sar, rpmpkgs
Network	novell_groupwise, tcp
Printers	cups_access, cups_error, spooler
Routers and firewalls	cisco_cdr, cisco_syslog, clavister
VoIP	asterisk_cdr, asterisk_event, asterisk_messages, asterisk_queue
Webservers	access_combined, access_combined_wcookie, access_common, apache_error, iis
Miscellaneous	snort

Finding out how a pre-trained source type is configured to work

To find out what configuration information Splunk is using to index a given source type, you can use the `btool` utility to list out the properties. For more information on using `btool`, refer to "Command line tools for use with Support's direction" in this manual.

The following example shows how to list out the configuration for the `tcp` source type:

```
$ btool props list tcp
[tcp]
BREAK_ONLY_BEFORE = (=\+)+
BREAK_ONLY_BEFORE_DATE = True
CHARSET = UTF-8
DATETIME_CONFIG = /etc/datetime.xml
KV_MODE = none
LEARN_SOURCETYPE = true
MAX_DAYS_AGO = 2000
MAX_DAYS_HENCE = 2
MAX_DIFF_SECS_AGO = 3600
MAX_DIFF_SECS_HENCE = 604800
MAX_EVENTS = 256
MAX_TIMESTAMP_LOOKAHEAD = 128
MUST_BREAK_AFTER =
MUST_NOT_BREAK_AFTER =
MUST_NOT_BREAK_BEFORE =
REPORT-tcp = tcpdump-endpoints, colon-kv
SEGMENTATION = inner
SEGMENTATION-all = full
SEGMENTATION-inner = inner
SEGMENTATION-outer = foo
SEGMENTATION-raw = none
SEGMENTATION-standard = standard
SHOULD_LINEMERGE = True
TRANSFORMS =
TRANSFORMS-baindex = banner-index
TRANSFORMS-dlindex = download-index
TRUNCATE = 10000
maxDist = 100
pulldown_type = true
```

Specify source type settings in props.conf

Specify source type settings in props.conf

There are source type specific settings in props.conf. Specify settings for a source type using the following attribute/value pairs. Add a sourcetype stanza to `props.conf` in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For more information on configuration files, see "About configuration files".

Note: The following attribute/value pairs can only be set for a stanza that begins with `[<$SOURCETYPE>]`:

```
invalid_cause = <string>
```

- Can only be set for a [<sourcetype>] stanza.
- Splunk will not index any data with invalid_cause set.
- Set <string> to "archive" to send the file to the archive processor (specified in unarchive_cmd).
- Set to any other string to throw an error in the splunkd.log if running Splunklogger in debug mode.
- Defaults to empty.

`unarchive_cmd = <string>`

- Only called if invalid_cause is set to "archive".
- <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.
- Defaults to empty.

`LEARN_MODEL = <true/false>`

- For known sourcetypes, the fileclassifier will add a model file to the learned directory.
- To disable this behavior for diverse sourcetypes (such as sourcecode, where there is no good exemplar to make a sourcetype) set LEARN_MODEL = false.
    - More specifically, set LEARN_MODEL to false if you can easily classify your source by its name or a rule and there's nothing gained from trying to analyze the content.
- Defaults to empty.

`maxDist = <integer>`

- Determines how different a sourcetype model may be from the current file.
- The larger the value, the more forgiving.
- For example, if the value is very small (e.g., 10), then files of the specified sourcetype should not vary much.
- A larger value indicates that files of the given sourcetype vary quite a bit.
- Defaults to 300.

Configure index-time field extractions

Configure index-time field extractions

We do *not* recommend that you add custom fields to the set of **default fields** that Splunk automatically extracts and indexes at **index time**, such as `timestamp`, `punct`, `host`, `source`, and `sourcetype`. Adding to this list of fields can negatively impact indexing performance and search times, because each indexed field increases the size of the searchable index. Indexed fields are also less flexible--whenever you make changes to your set of fields, you must re-index your entire dataset. For more information, see "Index time versus search time" in the Admin manual.

With those caveats, there are times when you may find a need to change or add to your indexed fields. For example, you may have situations where certain search-time field extractions are noticeably impacting search performance. This can happen, for example, if you commonly search a large event set with expressions like `foo!=bar` or `NOT foo=bar`, and the field `foo` nearly *always* takes on the value `bar`.

Conversely, you may want to add an indexed field if the value of a search-time extracted field exists outside of the field more often than not. For example, if you commonly search only for `foo=1`, but 1 occurs in many events that do *not* have `foo=1`, you may want to add `foo` to the list of fields extracted by Splunk at index time.

In general, you should try to extract your fields at search time. For more information see "Create search-time field extractions" in the Knowledge Manager manual.

Define additional indexed fields

Define additional indexed fields by editing `props.conf`, `transforms.conf` and `fields.conf`.

Edit these files in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For more information on configuration files in general, see "About configuration files" in this manual.

Splunk only accepts field names that contain alpha-numeric characters or an underscore:

- Valid characters for field names are **a-z, A-Z, 0-9,** or _ .
- Field names cannot begin with **0-9** or _ . Leading underscores are reserved for Splunk's internal variables.
- International characters are not allowed.

Add a regex stanza for the new field to transforms.conf

Follow this format when you define an index-time field transform in `transforms.conf` (Note: Some of these attributes, such as `LOOKAHEAD` and `DEST_KEY`, are only required for certain use cases):

```
[<unique_stanza_name>]
REGEX = <your_regex>
FORMAT = <your_custom_field_name>::"$1"
WRITE_META = <true | false>
DEST_KEY = <KEY>
DEFAULT_VALUE = <string>
SOURCE_KEY = <KEY>
REPEAT_MATCH = <true | false>
LOOKAHEAD = <integer>
```

- The `<unique_stanza_name>` is required for all transforms, as is the `REGEX`.

- `REGEX` is a regular expression that operates on your data to extract fields.
  - Name-capturing groups in the `REGEX` are extracted directly to fields, which means that you don't have to specify a `FORMAT` for simple field extraction cases.
  - If the `REGEX` extracts both the field name *and* its corresponding value, you can use the following special capturing groups to skip specifying the mapping in the `FORMAT` attribute:

    `_KEY_<string>, _VAL_<string>`

- For example, the following are equivalent:

  Using `FORMAT`:

  ```
  REGEX = ([a-z]+)=([a-z]+)
  FORMAT = $1::$2
  ```

  Not using `FORMAT`:

  ```
  REGEX = (?<_KEY_1>[a-z]+)=(?<_VAL_1>[a-z]+)
  ```

- `FORMAT` is optional. Use it to specify the format of the field/value pair(s) that you are extracting, including any field names or values that you want to add. You don't need to specify the `FORMAT` if you have a simple `REGEX` with name-capturing groups.
  - For index-time transforms, you use `$n` to specify the output of each `REGEX` match (for example, `$1`, `$2`, and so on).
  - If the `REGEX` does not have `n` groups, the matching fails.
  - `FORMAT` defaults to `$1`.
  - The special identifier `$0` represents what was in the `DEST_KEY` before the `REGEX` was performed (in the case of index-time field extractions the `DEST_KEY` is `_meta`). For more information, see "How Splunk builds indexed fields," below.
  - For index-time field extractions, you can set up `FORMAT` in several ways. It can be a `<field-name>::<field-value>` setup like

    `FORMAT = field1::$1 field2::$2` (where the `REGEX` extracts field values for captured groups "field1" and "field2")

    or

    `FORMAT = $1::$2 $3::$4` (where the `REGEX` extracts both the field name and the field value)

    However you can also set up index-time field extractions that create concatenated fields:

    ```
    FORMAT = ipaddress::$1.$2.$3.$4
    ```

- `WRITE_META = true` writes the extracted field name and value to `_meta`, which is where Splunk stores indexed fields. This attribute setting is required for all index-time field extractions, except for those where `DEST_KEY = meta` (see the discussion of `DEST_KEY`, below.
  - For more information about `_meta` and its role in indexed field creation, see "How Splunk builds indexed fields," below.

- `DEST_KEY` is required for index-time field extractions where `WRITE_META = false` or is not set. It specifies where Splunk sends the results of the `REGEX`.
  - For index-time searches, `DEST_KEY = _meta`, which is where Splunk stores indexed fields. For other possible `KEY` values see the `transforms.conf` page in this manual.

- For more information about `_meta` and its role in indexed field creation, see "How Splunk builds indexed fields," below.
- When you use `DEST_KEY = _meta` 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`.
- **Note:** The `$0` value is in no way derived *from* the `REGEX`.

- `DEFAULT_VALUE` is optional. The value for this attribute is written to `DEST_KEY` if the `REGEX` fails.
  - Defaults to empty.

- `SOURCE_KEY` is optional. You use it to identify a KEY whose values the `REGEX` should be applied to.
  - By default, `SOURCE_KEY = _raw`, which means it is applied to the entirety of all events.
  - Typically used in conjunction with `REPEAT_MATCH`.
  - For other possible `KEY` values see the `transforms.conf` page in this manual.

- `REPEAT_MATCH` is optional. Set it to `true` to run 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 field/value matches are expected per event.
  - Defaults to `false`.

- `LOOKAHEAD` is optional. Use it to specify how many characters to search into an event.
  - Defaults to 256. You may want to increase your `LOOKAHEAD` value if you have events with line lengths longer than 256 characters.

Note: For a primer on regular expression syntax and usage, see Regular-Expressions.info. You can test regexes by using them in searches with the rex search command. Splunk also maintains a list of useful third-party tools for writing and testing regular expressions.

Note: The capturing groups in your regex must identify field names that use ASCII characters (a-zA-Z0-9_-.). International characters will not work.

Link the new field to props.conf

To `props.conf`, add the following lines:

```
[<spec>]
TRANSFORMS-<value> = <unique_stanza_name>
```

- `<spec>` can be:
  - <sourcetype>, the sourcetype of an event.
  - host::<host>, where <host> is the host for an event.
  - source::<source>, where <source> is the source for an event.
  - **Note:** You can use regex-type syntax when setting the `<spec>`. Also, source and source type stanzas match in a case-sensitive manner while host stanzas do not. For more information, see the `props.conf` spec file.

- <value> is any value you want, to give your attribute its name-space.
- <unique_stanza_name> is the name of your stanza from transforms.conf.

Note: For index-time field extraction, props.conf uses TRANSFORMS-<class>, as opposed to EXTRACT-<value>, which is used for configuring search-time field extraction.

Add an entry to fields.conf for the new field

Add an entry to fields.conf for the new indexed field:

```
[<your_custom_field_name>]
INDEXED=true
```

- <your_custom_field_name> is the name of the custom field you set in the unique stanza that you added to transforms.conf.
- Set INDEXED=true to indicate that the field is indexed.

Note: If a field of the same name is extracted at search time, you **must** set INDEXED=false for the field. In addition, you must *also* set INDEXED_VALUE=false if events exist that have values of that field that are not pulled out at index time, but which *are* extracted at search time.

For example, say you're performing a simple <field>::1234 extraction at index time. This could work, but you would have problems if you also implement a search-time field extraction based on a regex like A(\d+)B, where the string A1234B yields a value for that field of 1234. This would turn up events for 1234 at search time that Splunk would be unable to locate at index time with the <field>::1234 extraction.

Restart Splunk for your changes to take effect

Changes to configuration files such as props.conf and transforms.conf won't take effect until you shut down and restart Splunk.

How Splunk builds indexed fields

Splunk builds indexed fields by writing to _meta. Here's how it works:

- _meta is modified by all matching transforms in transforms.conf that contain either DEST_KEY = _meta or WRITE_META = true.
- Each matching transform can overwrite _meta, so use WRITE_META = true to append _meta.
    - If you don't use WRITE_META, then start your FORMAT with $0.
- After _meta is fully built during parsing, Splunk interprets the text in the following way:
    - The text is broken into units; each unit is separated by whitespace.
    - Quotation marks (" ") group characters into larger units, regardless of whitespace.
    - Backslashes (\ ) immediately preceding quotation marks disable the grouping properties of quotation marks.
    - Backslashes preceding a backslash disable that backslash.

- Units of text that contain a double colon (::) are turned into extracted fields. The text on the left side of the double colon becomes the field name, and the right side becomes the value.

Note: Indexed fields with regex-extracted values containing quotation marks will generally not work, and backslashes may also have problems. Fields extracted at search time do not have these limitations.

Here's an example of a set of index-time extractions involving quotation marks and backslashes to disable quotation marks and backslashes.

```
WRITE_META = true
FORMAT = field1::value field2::"value 2" field3::"a field with a \" quotation mark" field4::"a
ends with a backslash\\"
```

When Splunk creates field names

When Splunk creates field names, it applies the following rules to all extracted fields, whether they are extracted at index-time or search-time, by default or through a custom configuration:

1. All characters that are not in a-z,A-Z, and 0-9 ranges are replaced with an underscore (_).

2. All leading underscores are removed. In Splunk, leading underscores are reserved for internal variables.

Index-time field extraction examples

Here are a set of examples of configuration file setups for index-time field extractions.

Define a new indexed field

This basic example creates an indexed field called `err_code`.

transforms.conf

In `transforms.conf` add:

```
[netscreen-error]
REGEX =  device_id=\[\w+\](?<err_code>[^:]+)
FORMAT = err_code::"$1"
WRITE_META = true
```

This stanza takes `device_id=` followed with a word within brackets and a text string terminating with a colon. The source type of the events is `testlog`.

Comments:

- The `FORMAT =` line contains the following values:
  - `err_code::` is the name of the field.

- ♦ $1 refers to the new field written to the index. It is the value extracted by `REGEX`.

- `WRITE_META = true` is an instruction to write the content of `FORMAT` to the index.

props.conf

Add the following lines to `props.conf`:

```
[testlog]
TRANSFORMS-netscreen = netscreen-error
```

fields.conf

Add the following lines to `fields.conf`:

```
[err_code]
INDEXED=true
```

Restart Splunk for your configuration file changes to take effect.

Define two new indexed fields with one regex

This example creates two indexed fields called `username` and `login_result`.

transforms.conf

In `transforms.conf` add:

```
[ftpd-login]
REGEX = Attempt to login by user: (.*): login (.*)\.
FORMAT = username::"$1" login_result::"$2"
WRITE_META = true
```

This stanza finds the literal text `Attempt to login by user:`, extracts a username followed by a colon, and then the result, which is followed by a period. A line might look like:

```
2008-10-30 14:15:21 mightyhost awesomeftpd INFO Attempt to login by user:
root: login FAILED.
```

props.conf

Add the following lines to `props.conf`:

```
[ftpd-log]
TRANSFORMS-login = ftpd-login
```

fields.conf

Add the following lines to `fields.conf`:

```
[username]
INDEXED=true

[login_result]
INDEXED=true
```

Restart Splunk for your configuration file changes to take effect.

Concatenate field values from event segments at index time

This example shows you how an index-time transform can be used to extract separate segments of an event and combine them to create a single field, using the `FORMAT` option.

Let's say you have the following event:

```
20100126 08:48:49 781 PACKET 078FCFD0 UDP Rcv 127.0.0.0 8226 R Q [0084 A
NOERROR] A (4)www(8)google(3)com(0)
```

Now, what you want to do is get `(4)www(8)google(3)com(0)` extracted as a value of a field named `dns_requestor`. But you don't want those garbage parentheses and numerals, you just want something that looks like `www.google.com`. How do you achieve this?

transforms.conf

You would start by setting up a transform in `transforms.conf` named `dnsRequest`:

```
[dnsRequest]
REGEX = UDP[^\(]+\(\d\)(\w+)\(\d\)(\w+)\(\d\)(\w+)
FORMAT = dns_requestor::$1.$2.$3
```

This transform defines a custom field named `dns_requestor`. It uses its `REGEX` to pull out the three segments of the `dns_requestor` value. Then it uses `FORMAT` to order those segments with periods between them, like a proper URL.

Note: This method of concatenating event segments into a complete field value is something you can *only* perform with index-time extractions; search-time extractions have practical restrictions that prevent it. If you find that you must use `FORMAT` in this manner, you will have to create a new indexed field to do it.

props.conf

Then, the next step would be to define a field extraction in `props.conf` that *references* the `dnsRequest` transform and applies it to events coming from the `server1` source type:

```
[server1]
TRANSFORM-dnsExtract = dnsRequest
```

fields.conf

Finally, you would enter the following stanza in `fields.conf`

```
[dns_requestor]
INDEXED = true
```

Restart Splunk for your configuration file changes to take effect.

Extract fields from file headers at index time

Extract fields from file headers at index time

Certain data sources and source types, such as CSV and MS Exchange log files, can have headers that contain field information. You can configure Splunk to automatically extract these fields during index-time event processing.

For example, a legacy CSV file--which is essentially a static table--could have a header row like

```
name, location, message, "start date"
```

which behaves like a series of column headers for the values listed afterwards in the file.

Note: Automatic header-based field extraction doesn't impact index size or indexing performance because it occurs during source typing (before index time).

How automatic header-based field extraction works

When you enable automatic header-based field extraction for a specific source or source type, Splunk scans it for header field information, which it then uses for field extraction. If a source has the necessary header information, Splunk extracts fields using delimiter-based key/value extraction.

Splunk does this at index time by changing the source type of the incoming data to `[original_sourcetype]-N`, where N is a number). Next, it creates a stanza for this new source type in `props.conf`, defines a delimeter-based extraction rule for the static table header in `transforms.conf`, and then ties that extraction rule to the new source type back in its new `props.conf` stanza. Finally, at search time, Splunk applies field transform to events from the source (the static table file).

You can use fields extracted by Splunk for filtering and reporting just like any other field by selecting them from the fields sidebar in the Search view (select **Pick fields** to see a complete list of available fields).

Note: Splunk will record the header line of a static table in a CSV or similar file as an event. To perform a search that gets a count of the events in the file *without* including the header event, you can run a search that identifies the file as the source while explicitly excluding the comma delimited list of header names that appears in the event. Here's an example:

```
source=/my/file.csv NOT "header_field1,header_field2,header_field3,..." |
stats count
```

Enable automatic header-based field extraction

Enable automatic header-based field extraction for any source or source type by editing `props.conf`. Edit this file in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/<app_name>/local`.

For more information on configuration files in general, see "About configuration files" in the Admin manual.

To turn on automatic header-based field extraction for a source or source type, add `CHECK_FOR_HEADER=TRUE` under that source or source type's stanza in `props.conf`.

Example `props.conf` entry for an MS Exchange source:

```
[MSExchange]
CHECK_FOR_HEADER=TRUE
...
```

OR

```
[source::C:\\Program Files\\Exchsrvr\\ServerName.log]
sourcetype=MSExchange

[MSExchange]
CHECK_FOR_HEADER=TRUE
```

Set `CHECK_FOR_HEADER=FALSE` to turn off automatic header-based field extraction for a source or source type.

Important: Changes you make to `props.conf` (such as enabling automatic header-based field extraction) won't take effect until you restart Splunk.

Note: CHECK_FOR_HEADER must be in a source or source type stanza.

Changes Splunk makes to configuration files

If you enable automatic header-based field extraction for a source or sourcetype, Splunk adds stanzas to copies of `transforms.conf` and `props.conf` in `$SPLUNK_HOME/etc/apps/learned/local/` when it extracts fields for that source or sourcetype.

Important: Don't edit these stanzas after Splunk adds them, or the related extracted fields won't work.

transforms.conf

Splunk creates a stanza in `transforms.conf` for each source type with unique header information matching a source type defined in `props.conf`. Splunk names each stanza it creates as `[AutoHeader-N]`, where `N` in an integer that increments sequentially for each source that has a unique header (`[AutoHeader-1], [AutoHeader-2],...,[AutoHeader-N]`). Splunk populates each stanza with transforms that the fields (using header information).

Here the `transforms.conf` entry that Splunk would add for the MS Exchange source that was enabled for automatic header-based field extraction in the preceding example:

```
...
[AutoHeader-1]
FIELDS="time", "client-ip", "cs-method", "sc-status"
DELIMS=" "
...
```

props.conf

Splunk then adds new sourcetype stanzas to `props.conf` for each source with a unique name, fieldset, and delimiter. Splunk names the stanzas as `[yoursource-N]`, where `yoursource` is the source type configured with automatic header-based field extraction, and `N` is an integer that increments sequentially for each transform in `transforms.conf`.

For example, say you're indexing a number of CSV files. If each of those files has the same set of header fields and with the same delimiter in `transforms.conf`, Splunk maps the events indexed from those files to a sourcetype of `csv-1` in `props.conf`. But if that batch of CSV files also includes a couple of files with unique sets of fields and delimiters, Splunk gives the events it indexes from those files sourcetypes of `csv-2` and `csv-3`, respectively. Events from files with the same source, fieldset, and delimiter in `transforms.conf` will have the same sourcetype value.

Note: If you want to enable automatic header-based field extraction for a particular source, and you have already manually specified a source type value for that source (either by defining the source type in Splunk Web or by directly adding the source type to a stanza in `inputs.conf`) be aware that setting `CHECK_FOR_HEADER=TRUE` for that source allows Splunk to override the source type value you've set for it with the sourcetypes generated by the automatic header-based field extraction process. This means that even though you may have set things up in `inputs.conf` so that all csv files get a sourcetype of `csv`, once you set `CHECK_FOR_HEADER=TRUE`, Splunk overrides that sourcetype setting with the incremental sourcetype names described above.

Here's the source type that Splunk would add to `props.conf` to tie the transform to the MS Exchange source mentoned earlier:

```
[MSExchange-1]
REPORT-AutoHeader = AutoHeader-1
...
```

Note about search and header-based field extraction

Use a wildcard to search for events associated with source types that Splunk generated during header-based field extraction.

For example, a search for `sourcetype="yoursource"` looks like this:

```
sourcetype=yoursource*
```

Examples of header-based field extraction

These examples show how header-based field extraction works with common source types.

MS Exchange source file

This example shows how Splunk extracts fields from an MS Exchange file using automatic header-based field extraction.

This sample MS Exchange log file has a header containing a list of field names, delimited by spaces:

```
# Message Tracking Log File
# Exchange System Attendant Version 6.5.7638.1
# Fields: time client-ip cs-method sc-status
14:13:11 10.1.1.9 HELO 250
14:13:13 10.1.1.9 MAIL 250
14:13:19 10.1.1.9 RCPT 250
14:13:29 10.1.1.9 DATA 250
14:13:31 10.1.1.9 QUIT 240
```

Splunk creates a header and transform in `tranforms.conf`:

```
[AutoHeader-1]
FIELDS="time", "client-ip", "cs-method", "sc-status"
DELIMS=" "
```

Note that Splunk automatically detects that the delimiter is a whitespace.

Splunk then ties the transform to the source by adding this to the source type stanza in `props.conf`:

```
# Original source type stanza you create
[MSExchange]
CHECK_FOR_HEADER=TRUE
...
# source type stanza that Splunk creates
[MSExchange-1]
REPORT-AutoHeader = AutoHeader-1
...
```

Splunk automatically extracts the following fields from each event:

```
14:13:11 10.1.1.9 HELO 250
```

- `time="14:13:11" client-ip="10.1.1.9" cs-method="HELO" sc-status="250"`

```
14:13:13 10.1.1.9 MAIL 250
```

- `time="14:13:13" client-ip="10.1.1.9" cs-method="MAIL" sc-status="250"`

```
14:13:19 10.1.1.9 RCPT 250
```

- `time="14:13:19" client-ip="10.1.1.9" cs-method="RCPT" sc-status="250"`

```
14:13:29 10.1.1.9 DATA 250
```

- `time="14:13:29" client-ip="10.1.1.9" cs-method="DATA" sc-status="250"`

```
14:13:31 10.1.1.9 QUIT 240
```

- `time="14:13:31" client-ip="10.1.1.9" cs-method="QUIT" sc-status="240"`

CSV file

This example shows how Splunk extracts fields from a CSV file using automatic header-based field extraction.

Example CSV file contents:

```
foo,bar,anotherfoo,anotherbar
100,21,this is a long file,nomore
200,22,wow,o rly?
300,12,ya rly!,no wai!
```

Splunk creates a header and transform in `tranforms.conf` (located in: `$SPLUNK_HOME/etc/apps/learned/transforms.conf`):

```
# Some previous automatic header-based field extraction
[AutoHeader-1]
...
# source type stanza that Splunk creates
[AutoHeader-2]
FIELDS="foo", "bar", "anotherfoo", "anotherbar"
DELIMS=","
```

Note that Splunk automatically detects that the delim is a comma.

Splunk then ties the transform to the source by adding this to a new source type stanza in `props.conf`:

```
...
[CSV-1]
REPORT-AutoHeader = AutoHeader-2
...
```

Splunk extracts the following fields from each event:

```
100,21,this is a long file,nomore
```

- `foo="100" bar="21" anotherfoo="this is a long file" anotherbar="nomore"`

```
200,22,wow,o rly?
```

- `foo="200" bar="22" anotherfoo="wow" anotherbar="o rly?"`

```
300,12,ya rly!,no wai!
```

- `foo="300" bar="12" anotherfoo="ya rly!" anotherbar="no wai!"`

Answers

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

Add and manage users

About users and roles

About users and roles

If you're running Splunk Enterprise, you can create users with passwords and assign them to **roles** you have created. Splunk Free does not support user authentication.

Splunk 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 upon installing Splunk.

About roles

A role contains a set of **capabilities**, like whether or not someone is allowed to add inputs or edit saved searches, etc. The various capabilities are listed in Add users and assign roles and in `$SPLUNK_HOME/etc/system/README/authorize.conf.spec`. Once a role exists, you can assign users to that role.

Additionally, whenever you create a user, you can automatically create a role for that user.

By default, Splunk 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.

Disallowed characters

Usernames stored in Splunk's local authentication cannot contain spaces, colons, or forward slashes.

Role names must use lowercase characters only. They cannot contain spaces, colons, or forward slashes.

Find existing users and roles

To locate an existing user or role in Manager, use the Search bar at the top of the Users or Roles page. Wildcards are supported. Splunk searches for the string you enter in all available fields by default. 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=".

Add users and assign roles

Add users and assign roles

This topic describes how to create new users and change the properties (like password) of existing users. This topic also describes how to assign users to **roles** in Splunk's **role-based access control system**.

Note: Role names must use lowercase characters. For example: "admin", not "Admin".

Add and edit users via Splunk Web

- In Splunk Web, click **Manager**.
- Click **Access controls**.
- Click **Users**.
- Click **New** or edit an existing user.
- Specify new or changed information for this user.
- Assign this user to an existing role or roles and click **Save**.

When you create a user, you can create a role for that user as well. You can then edit that role to specify what access that user has to Splunk.

Add and edit users using the CLI

- To add a new administrator user with password changeme2:
    - `./splunk add user admin2 -password changeme2 -role admin -auth admin:changeme`
- To change an existing user password to fflanda:
    - `./splunk edit user admin -password fflanda -role admin -auth admin:changeme`

NB: Passwords with special characters that would be interpreted by the shell (for example '$' or '!') must be either escaped or single-quoted:

```
./splunk edit user admin -password 'fflanda$' -role admin -auth
admin:changeme
```

or

```
./splunk edit user admin -password fflanda\$ -role admin -auth
admin:changeme
```

Add and edit roles using Splunk Web

- In Splunk Web, click **Manager**.
- Click **Access controls**.
- Click **Roles**.
- Click **New** or edit an existing role.
- Specify new or changed information for this role. In particular, you can:

    - restrict what data this role can search with a search filter (see "Search filter format" below for more information)
    - restrict over how large of a window of time this role can search
    - specify whether this role inherits capabilities and properties from any other roles
    - choose individual capabilities for this role
    - specify an index or indexes that this role will search by default
    - specify whether this role is restricted to a specific index or indexes.
- Click **Save**.

Note: Members of multiple roles inherit capabilities and properties from the role with the loosest permissions.

Add and edit roles using authorize.conf

Configure roles by editing authorize.conf. Roles are defined by lists of capabilities. You can also use roles to create fine-grained access controls by setting a search filter for each role.

Caution: Do not edit or delete any roles in `$SPLUNK_HOME/etc/system/default/authorize.conf`. This could break your admin capabilities. Edit this file in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For more information on configuration files in general, see About configuration files.

Add the following attribute/value pairs to `$SPLUNK_HOME/etc/system/local/authorize.conf`:

```
[role_$ROLE_NAME]
$CAPABILITY1 = enabled
$CAPABILITY2 = enabled
...
importRoles = $OTHER_ROLES
srchFilter = $SEARCH_STRING
```

You can include these attributes:

- `role_$ROLE_NAME`
    - The name you want to give your role, for example `security`, `compliance`, `ninja`. Make sure the name is lowercase.
- `$CAPABILITY1`
    - Any capability from the list. here. You can have any number of capabilities for a role.
- `importRoles = <role>;<role>;...`
    - When set, the current role will inherit all the capabilities from `<role>`.
    - Separate multiple roles, if any, with semicolons.
- `srchFilter = <search>`
    - Use this field for fine-grained access controls. Searches for this role will be filtered by this expression.
- `srchTimeWin = <string>`
    - Maximum time span (in seconds) of a search executed by this role.
- `srchDiskQuota = <int>`
    - Maximum amount of disk space (MB) that can be taken by search jobs of a user that belongs to this role.

- `srchJobsQuota = <int>`
  - Maximum number of concurrently running searches a member of this role can have.
- `rtSrchJobsQuota = <number>`
  - Maximum number of concurrently running real-time searches a member of this role can have.
- `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 '_*'.
- `srchIndexesAllowed = <string>`
  - Semicolon delimited list of indexes this role is allowed to search.
  - Follows the same wildcarding semantics as `srchIndexesDefault`.

Note: You must reload authentication or restart Splunk after making changes to `authorize.conf`. Otherwise, your new roles will not appear in the **Role** list. To reload authentication, go to the **Manager > Authentication** section of Splunk Web. This refreshes the authentication caches, but does not boot current users.

Search filter format

The `srchFilter`/Search filter field can include any of the following search terms:

- `source=`
- `host=` and host tags
- `index=` and index names
- `eventtype=` and event type tags
- `sourcetype=`
- search fields
- wildcards
- use `OR` to use multiple terms, or `AND` to make searches more restrictive

Note: Members of multiple roles inherit properties from the role with the loosest permissions. In the case of search filters, if a user is assigned to roles with different search filters, they are all combined via OR.

The search terms cannot include:

- saved searches
- time operators
- regular expressions
- any fields or modifiers Splunk Web can overwrite

Map a user to a role via Splunk Web

Once you've created a role in `authorize.conf`, map a user or users to it via Splunk Web.

- Click on the **Manager** link in the upper right-hand corner.
- Then, click the **Users** link.
- Edit an existing user or create a new one.

- Choose which role to map to from the **Role** list.
    - ♦ Any custom roles you have created via `authorize.conf` will be listed here.

Example of creating a role in authorize.conf

This example creates the role "ninja", which inherits capabilities from the default "user" role. ninja has almost the same capabilities as the default "power" role, except it cannot schedule searches. In addition:

- 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 occupy up to 500MB total space on disk for all its jobs.

```
[role_ninja]
rtsearch = enabled
importRoles = user
srchFilter = host=foo
srchIndexesAllowed = *
srchIndexesDefault = mail;main
srchJobsQuota   = 8
rtSrchJobsQuota = 8
srchDiskQuota   = 500
```

List of available capabilities

This list shows capabilities available for roles. Check authorize.conf for the most up-to-date version of this list. The admin role has all the capabilities in this list except for the "delete_by_keyword" capability.

Capablity	Meaning
`admin_all_objects`	Has access to objects in the system (user objects, search jobs, etc.).
`change_authentication`	Can change authentication settings and reload authentication.
`change_own_password`	Can change own user password.
`delete_by_keyword`	Can use the "delete" search operator.
`edit_deployment_client`	Can change deployment client settings.
`edit_deployment_server`	Can change deployment server settings.
`edit_dist_peer`	Can add and edit peers for distributed search.
`edit_forwarders`	Can change forwarder settings.
`edit_httpauths`	Can edit and end user sessions.
`edit_input_defaults`	Can change default hostnames for input data.
`edit_monitor`	Can add inputs and edit settings for monitoring files.

edit_roles	Can edit roles and change user/role mappings.
edit_scripted	Can create and edit scripted inputs.
edit_search_server	Can edit general distributed search settings like timeouts, heartbeats, and blacklists.
edit_server	Can edit general server settings like server name, log levels, etc.
edit_splunktcp	Can change settings for receiving TCP inputs from another Splunk instance.
edit_splunktcp_ssl	Can list or edit any SSL-specific settings for Splunk TCP input.
edit_tcp	Can change settings for receiving general TCP inputs.
edit_udp	Can change settings for UDP inputs.
edit_user	Can create, edit, or remove users.
edit_web_settings	Can change settings for web.conf.
get_metadata	Enables the "metadata" search processor.
get_typeahead	Enables typeahead.
indexes_edit	Can change index settings like file size and memory limits.
license_tab	Can access and change the license.
list_forwarders	Can show forwarder settings.
list_httpauths	Can list user sessions.
list_inputs	Can list various inputs, including input from files, TCP, UDP, scripts, etc.
request_remote_tok	Can get a remote authentication token.
rest_apps_management	Can edit settings in the python remote apps handler.
rest_apps_view	Can list properties in the python remote apps handler.
rest_properties_get	Can get information from the services/properties endpoint.
rest_properties_set	Can edit the services/properties endpoint.
restart_splunkd	Can restart Splunk through the server control handler.
rtsearch	Can run real-time searches.
schedule_search	Can schedule saved searches.
search	Can run searches.
use_file_operator	Can use the "file" search operator.

Set up user authentication with Splunk

Set up user authentication with Splunk

Splunk ships with support for three types of authentication systems:

- Splunk's own built-in system

- LDAP
- A scripted authentication API for use with an external authentication system, such as PAM or RADIUS.

Splunk with an Enterprise license comes with Splunk's built-in authentication enabled by default. Splunk's authentication allows you to add users, assign them to roles, and give those roles custom permissions as needed for your organization. Splunk's built-in system always takes precedence over any external systems, including LDAP.

If your enterprise uses LDAP and you'd like to use the users and groups defined there instead of using Splunk's authentication system, look here for information. To use scripted authentication to connect with a system such as PAM or RADIUS, see this topic.

Set up user authentication with LDAP

Set up user authentication with LDAP

Splunk ships with support for three types of authentication systems:

- Splunk's own built-in system, described in Add users and assign roles.
- LDAP, described in the topic you're now reading.
- A scripted authentication API for use with an external authentication system, such as PAM or RADIUS, described in Configure Splunk to use scripted authentication.

Splunk supports LDAP v2 and v3, but does not support LDAP referrals. LDAP v3 is the default protocol used. Check the Splunk Community Wiki for information about ways to authenticate against an LDAP server that returns referrals (such as Active Directory).

Overview of the process

This topic provides procedures to do the following:

- Configure Splunk to use LDAP authentication
- Map existing LDAP groups to Splunk roles

In addition, see how to import your CA if your LDAP server requires it for SSL use.

Be sure to read "Things to know about Splunk and LDAP" at the end of this topic before proceeding.

User Management

You cannot add, edit, or delete LDAP users with Splunk. Instead, you must manage users within your LDAP server. For example:

- To add an LDAP user to a Splunk role, add the user to the LDAP group on your LDAP server.
- To change a user's role membership, change the LDAP group that the user is a member of on your LDAP server.
- To remove a user from a Splunk role, remove the user from the LDAP group on your LDAP server.

Note: Beginning with 4.1, Splunk automatically checks LDAP membership information when a user attempts to log into Splunk. You no longer need to reload the authentication configuration when adding or removing users.

Configure LDAP

This topic describes how to configure LDAP through Splunk Web. If you want to configure LDAP by editing `authentication.conf`, you can see complete configuration examples in the Configuration file reference and the Splunk Community Wiki topic "Authenticate against an LDAP server that returns referrals".

If you are configuring authentication via the configuration file and wish to switch back to the default Splunk authentication, the simplest way is to move the existing `authentication.conf` file out of the way (rename to `*.disabled` is fine) and restart Splunk. This will retain your previous configuration unchanged if you expect to return to it later.

Determine your User and Group Base DN

Before you map your LDAP settings in Splunk, figure out your user and group base DN, or distinguished name. The DN is the location in the directory where authentication information is stored. If group membership information for users is kept in a separate entry, enter a separate DN identifying the subtree in the directory where the group information is stored. If your LDAP tree does not have group entries, you can set the group base DN to the same as the user base DN to treat users as their own group. This requires further configuration, described later.

If you are unable to get this information, please contact your LDAP Administrator for assistance.

Set up LDAP via Splunk Web

First, set LDAP as your authentication strategy:

1. Click **Manager** in Splunk Web.

2. Under **System configurations**, click **Access controls**.

3. Click **Authentication method**.

4. Select the **LDAP** radio button.

5. Click **Configure Splunk to work with LDAP**.

6. Click **New**.

7. Enter an **LDAP strategy name** for your configuration.

8. Enter the **Host** name of your LDAP server. Be sure that your Splunk Server can resolve the host name.

9. Enter the **Port** that Splunk should use to connect to your LDAP server.

- By default LDAP servers listen on TCP port 389.
- LDAPS (LDAP with SSL) defaults to port 636.

10. To turn on SSL, check **SSL enabled**.

- **Important:** This setting is recommended for security.
- **Note:** You must also have SSL enabled on your LDAP server.

11. Enter the **Bind DN**.

- This is the distinguished name used to bind to the LDAP server.
- This is typically the administrator or manager user. This user needs to have access to all LDAP user and group entries you want to retrieve.
- Leave blank if anonymous bind is sufficient.

12. Enter and confirm the **Bind DN password** for the binding user.

13. Specify the **User base DN**. You can specify multiple user base DN entries by separating them with semicolons.

- Splunk uses this attribute to locate user information.

- **Note:** You must set this attribute for authentication to work.

14. Enter the **User base filter** for the object class you want to filter your users on.

- **Note:** This is recommended to return only applicable users. For example, (department=IT).
- Default value is empty, meaning no user entry filtering.

15. Enter the **User name attribute** that contains the user name.

- **Note:** The username attribute cannot contain whitespace. The username must be lowercase.
- In Active Directory, this is `sAMAccountName`.
- The value `uid` should work for most other configurations.

16. Enter the **Real name attribute** (common name) of the user.

- Typical values are `displayName` or `cn` (common name).

17. Enter the **Group mapping attribute**.

- This is the user entry attribute whose value is used by group entries to declare membership.
- The default is `dn` for active directory; set this attribute only if groups are mapped using some other attribute besides user DN.
- For example, a typical attribute used to map users to groups is `uid`.

18. Enter the **Group base DN**. You can specify multiple group base DN entries by separating them with semicolons.

- Ths is the location of the user groups in LDAP.
- If your LDAP environment does not have group entries, you can treat each user as its own group:
    - Set groupBaseDN to the same value as userBaseDN. This 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.

19. Enter the **Group base filter** for the object class you want to filter your groups on.

- **Note:** This is recommended to return only applicable groups. For example, (department=IT).
- Default value is empty, meaning no group entry filtering.

20. Enter the **Group name attribute**.

- This is the group entry attribute whose value stores the group name.
- This is usually `cn`.

21. Enter the **Group member attribute**.

- This is the group attribute whose values are the group's members.
- This is typically `member` or `memberUid`.

Map existing LDAP groups to Splunk roles

Once you have configured Splunk to authenticate via your LDAP server, map your existing LDAP groups to any roles you have created. If you do not use groups, you can map users individually.

Note: You can map either users or groups, but not both. If you are using groups, all users you want to access Splunk must be members of an appropriate group. Groups inherit capabilities from the highest level role they're a member of.

All users that can login are visible in the **Users** page in Splunk Manager. Assign roles to groups in the group mapping page under **Access controls** in Splunk Manager.

Test your LDAP configuration

If you find that your Splunk install is not able to successfully connect to your LDAP server, try these troubleshooting steps:

1. Check `$SPLUNK_HOME/var/log/splunk/splunkd.log` for any authentication errors.

2. Remove any custom values you've added for **userBaseFilter** and **groupBaseFilter**.

3. Perform an `ldapsearch` to confirm that the variables you are specifying will return the expected entries:

```
ldapsearch -h "<host>" -p "<port>" -b "<userBaseDN>" -x -D "<bindDN>" -W "realNameAttribute"

ldapsearch -h "<host>" -p "<port>" -b "<groupBaseDN>" -x -D "<bindDN>" -W "groupNameAttribute"
```

Note: On Solaris you have to add filter to the search:

```
ldapsearch -h "<host>" -p "<port>" -b "<groupBaseDN>" -x -D "<bindDN>" "(groupBaseFilter)"
```

Example using authentication.conf

This example steps you through the process of obtaining LDIFs and setting up `authentication.conf`. You can also enter these settings through Splunk Web, as described above.

Note: The particulars of your LDAP server may be different. Check your LDAP server settings and adapt `authentication.conf` attributes to your environment.

You can see a complete example authentication.conf here, and another example configuration in the Splunk Community Wiki topic: "Authenticate against an LDAP server that returns referrals".

Get LDIFs

You need the user and group LDIFs to set up `authentication.conf`.

User LDIF

Note: On Windows systems you can extract LDIFs with the ldifde command from the AD server:

```
ldifde -f output.ldif
```

The ldifde command will export all entries in AD. Then, open the file in a text editor and find the appropriate entries.

On non-Windows systems, get the user LDIF by running the following command (use your own `ou` and `dc`):

```
# ldapsearch -h ldaphost -p 389 -x -b "ou=People,dc=splunk,dc=com" -D
"cn=bind_user" -W
```

On Solaris:

```
# ldapsearch -h ldaphost -p 389 -x -b "ou=People,dc=splunk,dc=com" -D
"cn=bind_user" "(objectclass=*)" -W
```

This returns:

```
# splunkadmin, People, splunk.com
dn: uid=splunkadmin,ou=People, dc=splunk,dc=com
uid: splunkadmin
givenName: Splunk
```

```
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetorgperson
sn: Admin
cn: Splunk Admin
```

Group LDIF

Get the group LDIF by running the following command (use your own `ou` and `dc`):

```
# ldapsearch -h ldaphost -p 389 -x -b "ou=groups,dc=splunk,dc=com" -D
"cn=bind_user" -W
```

This returns:

```
# SplunkAdmins, Groups, splunk.com
dn: cn=SplunkAdmins,ou=Groups, dc=splunk,dc=com
description: Splunk Admins
objectClass: top
objectClass: groupofuniquenames
cn: SplunkAdmins
uniqueMember: uid=splunkadmin,ou=People, dc=splunk,dc=com
```

Configure authentication.conf

Use the following instructions to configure `authentication.conf`. Edit the file in `$SPLUNK_HOME/etc/system/local/`. For more information on configuration files in general, see "About configuration files".

Set authentication type

By default, Splunk uses its own authentication type. Change that to LDAP in the `[authentication]` stanza:

```
[authentication]
authType = LDAP
authSettings = ldaphost
```

Note the following:

- Turn on LDAP by setting `authType = LDAP`.
- Map `authSettings` to your LDAP configuration stanza (below).

Map to LDAP server entries

Now, map your LDIFs to the attribute/values in `authentication.conf`:

```
[ldaphost]
host = ldaphost.domain.com
port = 389
SSLEnabled = 0
bindDN = cn=bind_user
bindDNpassword = bind_user_password
```

```
groupBaseDN = ou=Groups,dc=splunk,dc=com
groupBaseFilter = (objectclass=*)
groupMappingAttribute = dn
groupMemberAttribute = uniqueMember
groupNameAttribute = cn
realNameAttribute = displayName
userBaseDN = ou=People,dc=splunk,dc=com
userBaseFilter = (objectclass=*)
userNameAttribute = uid
```

Map roles

You can set up a stanza to map any custom roles you have created in authorize.conf to LDAP groups you have enabled for Splunk access in `authentication.conf`:

```
[roleMap]
admin = SplunkAdmins
itusers = ITAdmins
```

Map users directly

If you need to map users directly to a Splunk role, you can do so by setting the `groupBaseDN` to the value of `userBaseDN`. Also, set the attributes for `groupMappingAttribute`, `groupMemberAttribute`, and `groupNameAttribute` to the same attribute as `userNameAttribute`. For example:

```
[supportLDAP]
SSLEnabled = 0
bindDN = cn=Directory Manager
bindDNpassword = #########
groupBaseDN = ou=People,dc=splunksupport,dc=com
groupBaseFilter = (objectclass=*)
groupMappingAttribute = uid
groupMemberAttribute = uid
groupNameAttribute = uid
host = supportldap.splunksupport.com
port = 389
realNameAttribute = cn
userBaseDN = ou=People,dc=splunksupport,dc=com
userBaseFilter = (objectclass=*)
userNameAttribute = uid

[roleMap]
admin = Gina Lee
```

Converting from Splunk built-in authentication to LDAP

Usernames in Splunk's built-in authentication system always take precedence over the same usernames in LDAP. So, if you have converted from Splunk's built-in authentication system to LDAP, you might need to delete users from Splunk's built-in system to ensure that you're using LDAP credentials. This is only necessary if usernames are the same in both systems.

If your LDAP usernames are the same as the names you previously used in the built-in system, saved searches should work without any conversion.

If you have existing saved searches created when your system was using Splunk's built-in authentication and you'd like to transfer them to an LDAP user of a different name, edit the metadata:

1. Modify `$SPLUNK_HOME/etc/apps/<app_name>/metadata/local.meta` and swap the `owner = <username>` field under each `savedsearch` permission stanza to the corresponding LDAP username and save your changes.

2. Restart Splunk for your changes to take effect.

Things to know about Splunk and LDAP

When configuring Splunk to work with your LDAP instance, note the following:

- Entries in Splunk Web and `authentication.conf` are case sensitive.
- Splunk only works with one LDAP server at a time.
- Any user explicitly created locally using Splunk native authentication will have precedence over an LDAP user of the same name. For example, if the LDAP server has a user with a cname of 'admin' and the default Splunk user of the same name is present, the Splunk user will win. Only the local password will be accepted, and upon login the roles mapped to the local user will be in effect.
- The number of LDAP groups Splunk Web can display for mapping to roles is limited to the number your LDAP server can return in a query.
    - To prevent Splunk from listing unnecessary groups, use the `groupBaseFilter`. Example: `groupBaseFilter = (|(cn=SplunkAdmins)(cn=SplunkPowerUsers)(cn=Help Desk))`
    - If you must role map more than the maximum number of groups, you can edit `authentication.conf` directly:

```
[roleMap]
admin = SplunkAdmins
power = SplunkPowerUsers
user = Help Desk
```

Answers

Have questions? Visit Splunk Answers and see what questions and answers the Splunk community has around LDAP authentication with Splunk.

Set up user authentication with external systems

Set up user authentication with external systems

Splunk ships with support for three types of authentication systems:

- Splunk's own built-in system, described in Add users and assign roles.
- LDAP, described in Set up user authentication with LDAP.
- A scripted authentication API for use with an external authentication system, such as PAM or RADIUS, described in this topic.

How scripted authentication works

In scripted authentication, a user-generated Python script serves as the middleman between the Splunk server and an external authentication system such as PAM or RADIUS.

The API consists of a few functions that handle communications between Splunk and the authentication system. You need to create a script with handlers that implement those functions.

To use your authentication system with Splunk, make sure the authentication system is running and then do the following:

- Create the Python authentication script.
- Test the script with the command shell.
- Edit authentication.conf to specify scripted authentication and associated settings.

Examples

Splunk provides several example authentication scripts and associated configuration files, including one set for RADIUS and another for PAM. There is also a simple script called `dumbScripted.py`, which focuses on the interaction between the script and Splunk.

You can use an example script and configuration file as the starting point for creating your own script. You must modify them for your environment.

You can find these examples in `$SPLUNK_HOME/share/splunk/authScriptSamples/`. That directory also contains a README file with information on the examples, as well as additional information on setting up the connection between Splunk and external systems.

Important: Splunk does not provide support for these scripts, nor does it guarantee that they will fully meet your authentication and security needs. They are meant to serve as examples that you can modify or extend as needed.

Create the authentication script

You must create a Python script that implements these authentication functions:

- userLogin
- getUserInfo
- getUsers

The Splunk server will call these functions as necessary, either to authenticate user login or to obtain information on a user's roles.

The script can optionally also include a handler for this function:

- getSearchFilter

This table summarizes the authentication functions, their arguments, and their return values:

Function	Description	Argument string	Retu

`userLogin`	Login with user credentials.	`--username=<username>`
		
		`--password=<password>`
		
		(values passed one per line over `stdin`)
			`--status=success|fail`
			
			(safely passed over `stdout`)

`getUserInfo`	Return a user's information, including name and role(s).	`--username=<username>`	`--status=success|fail --userInfo=`
			
			Note the following:
			
			• `userInfo` must specify a sem
			• `<userId>` is deprecated; you
			• `<username>` is required.
			• `<realname>` is optional, but it
			• `<roles>` is required. To return
			For example: `admin:power`
			• This example returns just the r
			
			`--status=success --userInfo=;docs`

`getUsers`	Return information for all Splunk users.	none	`--status=success|fail --userInfo=`
			`--userInfo=<userId>;<username>;<r`
			`--userInfo=<userId>;<username>;<r`
			
			Note the following:
			
			• See `getUserInfo` for informa
			information.
			• Separate each user's informati

`getSearchFilter`	Optional. Returns the filters applied specifically to this user, along with those applied to the user's roles. The filters are OR'd together. **Note:** User-based search filters are optional and not recommended. A better approach is to assign search	`--username=<username>`	`--status=success|fail --search_fi`

	filters to roles and then assign users to the appropriate roles.		

See the example scripts for detailed information on how to implement these functions.

Test the script

Since the communication between Splunk and the script occurs via `stdin` and `stdout`, you can test the script interactively in your command shell, without needing to call it from Splunk. Be sure to send one argument per line and end each function call with an EOF (Ctrl-D).

Test each function individually, using this pattern:

```
> python [script] [function name]
[pass arguments here, one per line]
[send eof, with Ctrl-D]
[output appears here, check that it's correct]
>
```

The following example shows a debugging session that does some simple testing of a fictional script called "example.py", with two users "alice" and "bob". "alice" is a member of the "admin" and "super" roles, and "bob" is a member of the "user" role.

```
> python example.py userLogin
--username=alice
--password=correctpassword
<send an EOF>
--status=success
> python example.py userLogin
--username=bob
--password=wrongpassword
<send an EOF>
--status=fail
> python example.py getUsers
<no arguments for this function, send an EOF>
--status=success --userInfo=bob;bob;bob;user --userInfo=alice;alice;alice;admin:super
> python example.py getUserInfo
--username=bob
<send an EOF>
--status=success --userInfo=bob;bob;bob;user
> python example.py getUserInfo
--username=userdoesnotexist
<send an EOF>
--status=fail
>
```

Important: This is just an example of how to go about testing a script. It does not attempt to perform exhaustive debugging of any real script.

Edit authentication.conf

Add the following settings to `authentication.conf` in `$SPLUNK_HOME/etc/system/local/` to enable your script. You can also copy and edit a sample `authentication.conf` from `$SPLUNK_HOME/share/splunk/authScriptSamples/`.

Specify `Scripted` as your authentication type under the `[authentication]` stanza heading:

```
[authentication]
authType = Scripted
authSettings = script
```

Set script variables under the `[script]` stanza heading. For example:

```
[script]
scriptPath = $SPLUNK_HOME/bin/python $SPLUNK_HOME/bin/<scriptname.py>
```

Set cache durations

To significantly speed authentication performance when using scripted authentication, make use of Splunk's authentication caching capability. You do so by adding the optional `[cacheTiming]` stanza. Each script function (except `getSearchFilter`) has a settable `cacheTiming` attribute, which turns on caching for that function and specifies its cache duration. For example, to specify the cache timing for the `getUserInfo` function, use the `getUserInfoTTL` attribute. Caching for a function occurs only if its associated attribute is specified.

The `cacheTiming` settings specify the frequency at which Splunk calls your script to communicate with the external authentication system. You can specify time in seconds (s), minutes (m), hours (h), days (d), etc. Typically, you'll limit the cache frequency to seconds or minutes. If a unit is not specified, the value defaults to seconds. So, a value of "5" is equivalent to "5s".

This example shows typical values for the caches:

```
[cacheTiming]
userLoginTTL    = 10s
getUserInfoTTL  = 1m
getUsersTTL     = 2m
```

You'll want to set `userLoginTTL` to a low value, since this setting determines login requirements.

To refresh all caches immediately, use the CLI command `reload auth`:

```
./splunk reload auth
```

Note: This command does not boot current users off the system.

You can also refresh caches in Splunk Web:

1. Click **Manager** in the upper right-hand corner of Splunk Web.

2. Under System configurations, click **Access controls**.

3. Click **Authentication method**.

4. Click **Reload authentication configuration** to refresh the caches.

Each specified function, except `getUsers`, has a separate cache for each user. So, if you have 10 users logged on and you've specified the `getUserInfoTTL` attribute, the `getUserInfo` function will have 10 user-based caches. The `getUsers` function encompasses all users, so it has a single, global cache.

Edit pamauth for PAM authentication

If you're using PAM and you're unable to authenticate after following the steps in the example directory's README, edit `/etc/pam.d/pamauth` and add this line:

```
auth sufficient pam_unix.so
```

Use single sign-on (SSO) with Splunk

Use single sign-on (SSO) with Splunk

Use this topic to configure Splunk to work with your enterprise's **SSO** solution.

How Splunk works with SSO

Configuring Splunk to work with SSO requires that any Splunk instance to be accessed via SSO is secured behind an HTTP proxy. The HTTP proxy you configure is then responsible for handling authentication and is the sole entity capable of communicating with Splunk Web (and the underlying Splunk server process, `splunkd`). Splunk's SSO implementation currently supports IIS and Apache HTTP proxies.

How authentication is handled

Splunk's SSO implementation expects that your user authentication is handled outside of Splunk by a web proxy. The web proxy server must be configured to authenticate against your external authentication system. Once a user has been authenticated by the proxy, the proxy must insert the authenticated user's username as a `REMOTE_USER` header in all HTTP requests forwarded to Splunk Web.

Splunk accepts incoming HTTP requests which include a `REMOTE_USER` header from a trusted proxy. If the user in the `REMOTE_USER` header is not currently authenticated by Splunk, an authentication request is made to Splunk via a trusted authentication endpoint the `splunkd` process provides. If Splunk returns a valid session key, Splunk Web can begin making secure requests as the user set in the `REMOTE_USER` header. All subsequent requests from the proxy to Splunk Web must include the `REMOTE_USER` header. If `REMOTE_USER` is not provided in every request, the `REMOTE_USER` is assumed to not be authenticated and will receive a Splunk login screen.

Note: If your proxy uses some other remote user header name besides `REMOTE_USER`, you can change the name of the header through the `remoteUser` attribute in `web.conf`, as described below.

Important: Splunk's SSO implementation supports logging in to Splunk Web only, not the command line interface or directly to `splunkd` endpoints.

Configure Splunk to work with SSO

Setting up SSO on Splunk requires several steps:

- Set up a proxy server.
- Edit Splunk's `server.conf` file.
- Edit Splunk's `web.conf` file.
- Set up Splunk users.

Set up a proxy server

Splunk's SSO implementation supports most proxy servers, including Apache and IIS HTTP proxies. The proxy server must handle its own authentication and must insert the authorized username into a `REMOTE_USER` (or equivalent) header for all HTTP requests it forwards to Splunk.

For more information on configuring an Apache or IIS proxy server, refer to this material:

- For Apache, see http://httpd.apache.org/docs/2.0/mod/mod_proxy.html
- For IIS, the Splunk SSO implementation has been tested with Helicon's ISAPI_Rewrite 3.0. Get it at: http://www.helicontech.com/download-isapi_rewrite3.htm

Note: The Helicon ISAPI must include the proxy module. Do not use the 'lite' version of Helicon ISAPI, as it does not have the required proxy module.

Edit server.conf

Edit `$SPLUNK_HOME/etc/system/local/server.conf` to set the value of `trustedIP` to the IP address that will make the secure requests to the `splunkd` process. Typically, this is the address of your Splunk Web interface. If on the same machine, you can use the localhost IP (127.0.0.1). If you host Splunk Web on a different interface, use the IP for that interface.

Note: You can only configure `splunkd` to trust one Splunk Web IP.

Example:

```
[general]
serverName = myserver
trustedIP = 127.0.0.1

[sslConfig]
sslKeysfilePassword = $1&Jf96BQ0bdFG0
```

Edit web.conf

In the `[settings]` stanza of `$SPLUNK_HOME/etc/system/local/web.conf`, set one or more of the following attributes:

Attribute	Required?	Default	Value

trustedIP	yes	n/a	Set this to the IP address of the authenticating proxy or proxies. Specify a single address or a comma-separated list of addresses; IP ranges and netmask notation are not supported.
root_endpoint	no	the proxy's root	If you host Splunk Web behind a proxy that does not place Splunk Web at the proxy's root, change this setting to reflect the offset from the root. For example, if your proxy hosts Splunk Web at "fflanda.com:9000/splunk", set the value to `/splunk`.
remoteUser	no	REMOTE_USER	Sets the remote user header. Most proxies forward the authenticated username in an HTTP header called `REMOTE_USER`. However, some may use a different header, such as `REMOTE-USER` (with a hyphen instead of an underscore). If the proxy you are using does not use `REMOTE_USER`, specify the HTTP header that Splunk Web should look for.
tools.proxy.on	no	false	*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.
SSOMode	no	permissive	Specifies the SSO mode for Splunk Web. The value is either "permissive" or "strict": • Permissive mode honors incoming requests from IPs not specified in the trustedIP setting but refuses to use SSO authentication if it receives requests from these unsupported IPs. • Strict mode completely shuts down all requests unless they originate from an IP address specified in the trustedIP setting.

Example:

```
[default]

[settings]
mgmtHostPort = localhost:8089
httpport = 8000
enableSplunkWebSSL = 0
```

```
# SSO
remoteUser = REMOTE-USER
SSOMode = permissive
trustedIP = 10.1.6.199,10.1.7.161,10.1.7.38
```

Set up users in Splunk that match users in your authentication system

You must create or map users in Splunk that have the same username as the users authenticating via the proxy. For example, if you configure LDAP to use a name field that resolves as "omarlittle", Splunk assumes that the proxy will set `REMOTE_USER` to "omarlittle". Alternatively, you can create a Splunk user with a username of "omarlittle" to allow this user to login via SSO using native Splunk authentication. For information about creating users in Splunk, refer to "Add users and assign roles" in this manual.

Debug issues

Splunk provides a URI for debugging any problems with SSO. This URI is located at:

```
http://YourSplunkServer:8000/debug/sso
```

This page provides diagnostic information about headers, usernames, and more.

Delete user accounts using the CLI

Delete user accounts using the CLI

Remove all the user data (user accounts) from your Splunk installation by typing `./splunk clean` followed by the `userdata` argument. This deletes all the user accounts other than the default user accounts included with Splunk (admin, power, user).

Caution: Removing user data is irreversible; if you accidentally delete user data, you must re-add the accounts manually.

To remove all of the user accounts in the system:

```
./splunk clean userdata
```

To remove the user accounts in the system and force Splunk to skip the confirmation prompt:

```
./splunk clean userdata -f
```

User language and locale

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. When looking for a suitable translation, Splunk first tries to find an exact match for the whole locale, but will fallback to just the language specifier if the entire setting is not available. For example, translations for "fr" answer to requests for "fr_CA" and "fr_FR".

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:

```
en_GB
en_US
ja_JP
zh_CN
zh_TW
```

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 traditional European date format: `DD/MM/YYYY:HH:MM:SS`.

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

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 Manager field sets both of them. To set the timeout in the Manager:

1. Click **Manager** in the upper right-hand corner of Splunk Web.

2. Under System configurations, click **System settings**.

3. Click **General settings**.

4. In the **System 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 the Manager.

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.session.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.session.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 the Manager or in configuration files, you must restart Splunk for the change to take effect.

Manage indexes

About managing indexes

About managing indexes

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 create and specify other indexes for Splunk to use for different data inputs.

Indexes are stored in directories, which are located in `$SPLUNK_HOME/var/lib/splunk`. An index is a collection of directories. Index directories are also called **buckets** and are organized by age. For detailed information on index storage, see "How Splunk stores indexes".

In addition to the **main** index, Splunk comes preconfigured with a number of internal indexes. Internal indexes are named starting with an underscore (_). The internal indexes store audit, indexing volume, Splunk logging, and other data. You can see a full list of indexes in Splunk Web if you click on the **Manager** link in the upper right hand of Splunk Web and then click **Indexes**:

- **main**: the default Splunk index. All processed data is stored here unless otherwise specified.
- **_internal**: this index includes internal logs and metrics from Splunk's processors.
- **sampledata**: a small amount of sample data is stored here for training purposes.
- **_audit**: events from the file system change monitor, auditing, and all user search history.

Read on in this section for information about ways to manage the indexing process, including:

- Setting up multiple indexes, moving indexes, removing index data
- Managing disk usage by limiting index size or configuring segmentation

If you're interested in the indexing process

Refer to:

- The section How indexing works in this manual.
- The section "How Splunk stores indexes" in this manual.
- The section Set up and use summary indexes in the Knowledge Manager manual, for information on working with extremely large datasets.
- The topic about Search performance on the Community Wiki.

Set up multiple indexes

Set up multiple indexes

Splunk ships with an index called `main` that, by default, holds all your events. By default, Splunk also creates a number of other indexes for use by its internal systems, as well as for additional Splunk features such as summary indexing and event auditing.

Splunk with an Enterprise license lets you add an unlimited number of additional indexes. The `main`

index serves as the default index for any input or search command that doesn't specify an index, although you can change the default. You can add indexes using Splunk Web, Splunk's CLI, or indexes.conf.

Why have multiple indexes?

There are several key reasons for having multiple indexes:

- To control user access.
- To accommodate varying retention policies.
- To speed searches in certain situations.

The main reason you'd set up multiple indexes is to control user access to the data that's in them. When you assign users to roles, you can limit user searches to specific indexes based on the role they're in.

In addition, if you have different policies for retention for different sets of data, you might want to send the data to different indexes and then set a different archive or retention policy for each index.

Another reason to set up multiple indexes has to do with the way Splunk search works. If you have both a high-volume/high-noise data source and a low-volume data source feeding into the same index, and you search mostly for events from the low-volume data source, the search speed will be slower than necessary, because Splunk also has to search through all the data from the high-volume source. To mitigate this, you can create dedicated indexes for each data source and route data from each source to its dedicated index. Then, you can specify which index to search on. You'll probably notice an increase in search speed.

Specify an index or indexes to search

When Splunk searches, it targets the default index (by default, **main**) unless otherwise specified. If you have created a new index, or want to search in any index that is not default, you can specify the index in your search:

```
index=hatch userid=henry.gale
```
This searches in the `hatch` index for the `userid=henry.gale`.

You can also specify an alternate default index for a given role to search when you create or edit that role.

Create and edit indexes

You can create or edit indexes with Splunk Web, the Splunk CLI, or directly, via `indexes.conf`.

Use Splunk Web

1. In Splunk Web, navigate to **Manager > Indexes** and click **New**.

2. To create a new index, enter:

- A name for the index. Index names must consist of only numbers, letters, periods, underscores, and dashes.
- The path locations (all optional, will default to be rooted in `$SPLUNK_DB/<index_name>/`):
    - Home path; leave blank for default `$SPLUNK_DB/<index_name>/db`
    - Cold db path; leave blank for default `$SPLUNK_DB/<index_name>/colddb`
    - Thawed/resurrected db path, leave blank for default `$SPLUNK_DB/<index_name>/thaweddb`
- The maximum size (in MB; optional) of the entire index. Defaults to 500000MB.
- The maximum size (in MB; optional) of the hot (currently actively searched) portion of this index.

Note: When setting the maximum size (`maxDataSize`), you should use "auto_high_volume" for high volume indexes (such as the main index), otherwise use "auto".

3. When you've set the values you want, click **Save**. The index is created. You must restart Splunk when you create a new index or edit the properties of an existing index.

You can edit an index by clicking on the index name in the **Indexes** section of **Manager** in Splunk Web. If you edit the properties of an existing index, you must restart Splunk.

Properties that you cannot change are grayed out. To change these properties, use `indexes.conf`.

Note: Some index properties are configurable only if you create or edit indexes through the `indexes.conf` file. Check the `indexes.conf` topic for a complete list of properties.

Use the CLI

To use Splunk's CLI, navigate to the `$SPLUNK_HOME/bin/` directory and use the `./splunk` command.

Important: You must stop Splunk before you edit the properties of an existing index. You do not need to stop Splunk to create a new index.

To add or edit a new index called "fflanda" using the CLI:

```
./splunk [add|edit] index fflanda
```

You can also specify a value for any option in indexes.conf by passing it as a flag (for example, `-dir`) to the `[add|edit] index <name>` command.

You must restart Splunk when you create a new index or edit the properties of an existing index.

Edit indexes.conf

To add a new index, you add a stanza to `indexes.conf` in `$SPLUNK_HOME/etc/system/local`, identified by the name of the new index. See configuration details and examples in the indexes.conf topic.

Note: The most accurate and up-to-date 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** in this manual, or in `$SPLUNK_HOME/etc/system/README`.

Disable or delete an index

You can disable the use of an index in Splunk Web. To do this, navigate to **Manager > Indexes** and click **Disable** to the right of the index you want to disable.

To delete an index, edit `indexes.conf` and remove its stanza. You cannot delete an index with Splunk Web or the CLI.

Important: You must stop Splunk before deleting an index.

Route events to specific indexes

Just as you can route events to specific queues, you can also route events to specific indexes.

By default, Splunk sends all events to the index called **main**. However, you may want to send specific events to other indexes. For example, you might want to segment data or to send event data from a noisy source to an index that is dedicated to receiving it. You can route data locally or route data you are receiving from remote sources or Splunk instances.

Note: When you place data in an alternate index, you must specify the index in your search with the `index=` command when you want to search that index:

```
index=foo
```

Send all events from a data input to a specific index

To configure routing for all events from a particular data input to an alternate index, add the following to the appropriate stanza in inputs.conf.

```
index = myindex
```

The following example `inputs.conf` entry routes data to `index = fflanda`:

```
[monitor:///var/log]
disabled = false
index = fflanda
```

If you specify a different index on a forwarder, when the events reach the indexing instance they will be routed to the named index, which must already exist.

Route specific events to a different index

To route certain events to an alternate index, edit props.conf and transforms.conf on the local Splunk instance:

1. Identify a common attribute for the events that can be used to differentiate them.

2. In `props.conf`, create a stanza for the source, source type, or host. This stanza specifies a `transforms_name` that corresponds to a regex-containing stanza you will create in `transforms.conf`.

3. In `transforms.conf`, create an stanza named with the `transforms_name` you specified in step 2. This stanza:

- Specifies a regular expression that matches the identified attribute from step 1.
- Specifies the alternate index that events matching the attribute should be routed to.

The sections below fill out the details for steps 2 and 3.

Edit props.conf

Add the following stanza to `$SPLUNK_HOME/etc/system/local/props.conf`:

```
[<spec>]
TRANSFORMS-<class_name> = <transforms_name>
```

Note the following:

- `<spec>` is one of the following:
  - `<sourcetype>`, the sourcetype of an event
  - `host::<host>`, where `<host>` is the host for an event
  - `source::<source>`, where `<source>` is the source for an event
- `<class_name>` is any unique identifier.
- `<transforms_name>` is whatever unique identifier you want to give to your transform in `transforms.conf`.

Edit transforms.conf

Add the following stanza to `$SPLUNK_HOME/etc/system/local/transforms.conf`:

```
[<transforms_name>]
REGEX = <your_custom_regex>
DEST_KEY = _MetaData:Index
FORMAT = <alternate_index_name>
```

Note the following:

- `<transforms_name>` must match the `<transforms_name>` identifier you specified in `props.conf`.
- `<your_custom_regex>` must provide a match for the attribute you identified earlier.
- `DEST_KEY` must be set to the index attribute `_MetaData:Index`.
- `<alternate_index_name>` specifies the alternate index that the events will route to.

Example

In this example, we route events of `windows_snare_log` sourcetype to the appropriate index based on their log types. "Application" logs will go to an alternate index, while all other log types, such as

"Security", will go to the default index.

To make this determination, we use `props.conf` to direct events of `windows_snare_log` sourcetype through the `transforms.conf` stanza named "AppRedirect", where a regex then looks for the log type, "Application". Any event with a match on "Application" in the appropriate location is routed to the alternate index, "applogindex". All other events go to the default index.

Identify an attribute

The events in this example look like this:

```
web1.example.com        MSWinEventLog   1       Application     721     Wed Sep 06 17:05:31 200
4156    MSDTC   Unknown User    N/A     Information     WEB1    Printers                String
message: Session idle timeout over, tearing down the session.   179
web1.example.com        MSWinEventLog   1       Security        722     Wed Sep 06 17:59:08 200
576     Security        SYSTEM  User    Success Audit   WEB1    Privilege Use
Special privileges assigned to new logon:     User Name:      Domain:      Logon
ID: (0x0,0x4F3C5880)     Assigned: SeBackupPrivilege   SeRestorePrivilege
SeDebugPrivilege   SeChangeNotifyPrivilege   SeAssignPrimaryTokenPrivilege 525
```

Some events contain the value "Application", while others contain the value "Security" in the same location.

Edit props.conf

Add this stanza to `$SPLUNK_HOME/etc/system/local/props.conf`:

```
[windows_snare_syslog]
TRANSFORMS-index = AppRedirect
```

This directs events of `windows_snare_syslog` sourcetype to the `AppRedirect` stanza in `transforms.conf`.

Edit transforms.conf

Add this stanza to `$SPLUNK_HOME/etc/system/local/transforms.conf`:

```
[AppRedirect]
REGEX = MSWinEventLog\s+\d+\s+Application
DEST_KEY = _MetaData:Index
FORMAT = applogindex
```

This stanza processes the events directed here by `props.conf`. Events that match the regex, by containing the string "Application" in the specified location, get routed to the alternate index, "applogindex". All other events route to the default index.

Set limits on disk usage

Set limits on disk usage

There are several methods for controlling disk space used by Splunk. Most disk space will be used by Splunk's indexes, which include the compressed raw data. If you run out of disk space, Splunk will

stop indexing. You can set a minimum free space limit to control how low you will let free disk space fall before indexing stops. Indexing will resume once space exceeds the minimum.

Set minimum free disk space

You can set a minimum amount of space to keep free on the disk where indexed data is stored. If the limit is reached, the server stops indexing data until more space is available. The default minimum is 2000MB.

Note:

- Splunk will not clear any of its own disk space with this method. It will simply pause for more space to become available.
- Events can be lost if they are not written to a file during such a pause.

You can set minimium free disk space through Splunk Web, the CLI, or the `server.conf` configuration file.

In Splunk Web

- Click **Manager** in the upper right corner of Splunk Web.
- Click **System settings**.
- Under the **Index settings** section, find **Pause indexing if free disk space (in MB) falls below:**:

- Enter your desired minimum free disk space in megabytes.
- Click **Save**.

Restart Splunk for your changes to take effect.

From the command line interface (CLI)

You can set the minimum free disk space via Splunk's CLI. To use the CLI, navigate to the `$SPLUNK_HOME/bin/` directory and use the `./splunk` command. Here, you set the minimum free disk space to 20,000MB (20GB):

```
# splunk set minfreemb 20000

# splunk restart
```

In server.conf

You can also set the minimum free disk space in the server.conf file. The relevant stanza/attribute is this:

```
[diskUsage]
minFreeSpace = <num>
```

Note that `<num>` represents megabytes. The default is 2000.

Control database storage

The indexes.conf file contains index configuration settings. You can control disk storage usage by specifying maximum index size or maximum age of data. When one of these limits is reached, the oldest indexed data will be deleted (the default) or archived. You can archive the data by using a predefined archive script or creating your own.

For detailed instructions on how to use `indexes.conf` to set maximum index size or age, see "Set a retirement and archiving policy".

For information on creating archive scripts, see "Archive indexed data".

For detailed information on index storage, see "How Splunk stores indexes".

How Splunk stores indexes

How Splunk stores indexes

As Splunk indexes your data, it creates a bunch of files. These files contain two types of data:

- The raw data in compressed form
- Indexes that point to the raw data

Together, these files constitute the Splunk index. The files reside in sets of directories organized by age. Some directories contain newly indexed data; others contain previously indexed data. The number of such directories can grow quite large, depending on how much data you're indexing.

Why you might care

You might not care, actually. Splunk handles indexed data by default in a way that gracefully ages the data through several stages. After a long period of time, typically several years, Splunk removes old data from your system. You might well be fine with the default scheme it uses.

However, if you're indexing large amounts of data, have specific data retention requirements, or otherwise need to carefully plan your aging policy, you've got to read this topic. Also, to back up your data, it helps to know where to find it. So, read on....

How Splunk ages data

Each of the index directories is known as a **bucket**. To summarize so far:

- A Splunk "index" contains compressed raw data and associated indexes.
- A Splunk index resides across many age-designated index directories.
- An index directory is a bucket.

A bucket moves through several stages as it ages:

- hot
- warm
- cold
- frozen

As buckets age, they "roll" from one stage to the next. Newly indexed data goes into a hot bucket, which is a bucket that's both searchable and actively being written to. After the hot bucket reaches a certain size, it becomes a warm bucket, and a new hot bucket is created. Warm buckets are searchable, but are not actively written to. There are many warm buckets.

Once Splunk has created some maximum number of warm buckets, it begins to roll the warm buckets to cold based on their age. Always, the oldest warm bucket rolls to cold. Buckets continue to roll to cold as they age in this manner. After a set period of time, cold buckets roll to frozen, at which point they are either archived or deleted. By editing attributes in indexes.conf, you can specify the bucket aging policy, which determines when a bucket moves from one stage to the next.

Here are the stages that buckets age through:

Bucket stage	Description	Searchable?
Hot	Contains newly indexed data. Open for writing. One or more hot buckets for each index.	Yes.
Warm	Data rolled from hot. There are many warm buckets.	Yes.
Cold	Data rolled from warm. There are many cold buckets.	Yes, but only when the search specifies a time range included in these files.
Frozen	Data rolled from cold. Splunk deletes frozen data by default, but you can also archive it.	No.

The collection of buckets in a particular stage is sometimes referred to as a database or "db": the "hot db", the "warm db", the "cold db", etc.

What the index directories look like

Each bucket occupies its own subdirectory within a larger database directory. Splunk organizes the directories to distinguish between hot/warm/cold buckets. In addition, the bucket directory names are based on the age of the data.

Here's the directory structure for the default index:

Bucket type	Default location	Notes
Hot	`$SPLUNK_HOME/var/lib/splunk/defaultdb/db/*`	There can be multiple hot subdirectories. Each hot buck occupies its own subdirectory, uses this naming convention: `hot_v1_<ID>`
Warm	`$SPLUNK_HOME/var/lib/splunk/defaultdb/db/*`	There are multiple warm subd Each warm bucket occupies it subdirectory, which uses this convention: `db_<newest_time>_<oldest_ti` where `<newest_time>` and `<oldest_time>` are timestar indicating the age of the data The timestamps are expresse epoch time (in seconds). For e `db_1223658000_12236544` is a warm bucket containing d October 10, 2008, covering th period of 9am-10am.
Cold	`$SPLUNK_HOME/var/lib/splunk/defaultdb/colddb/*`	There are multiple cold subdir When warm buckets roll to col get moved into this directory, b not renamed.
Frozen	N/A: Data deleted, or archived into a directory structure of your design.	Deletion is the default; archivi accomplished through user-cr script.
Thawed	`$SPLUNK_HOME/var/lib/splunk/defaultdb/thaweddb/*`	Location for data that has bee archived and later thawed. Se "Restore archived data" for inf on restoring archived data to a "thawed" state.

The paths for hot/warm and cold directories are configurable, so you can store cold buckets in a separate location from hot/warm buckets. See "Use multiple partitions for index data".

Caution: All index locations must be writable.

You configure indexes in `indexes.conf`. You can edit a copy of `indexes.conf` in `$SPLUNK_HOME/etc/system/local/` or in your own custom application directory in `$SPLUNK_HOME/etc/apps/`. Do not edit the copy in `$SPLUNK_HOME/etc/system/default`. For information on configuration files and directory locations, see "About configuration files".

This table lists the key `indexes.conf` attributes affecting buckets and what they configure. It also provides links to other topics that show how to use these attributes. For the most detailed information on these attributes, as well as others, always refer to "indexes.conf".

Attribute	What it configures	Default
homePath	The path that contains the hot and warm buckets.	`$SPLUNK_HOME/var/lib/splunk/defaultdb/db/`
coldPath	The path that contains the cold buckets.	`$SPLUNK_HOME/var/lib/splunk/defaultdb/colddb/`
thawedPath	The path that contains any thawed buckets.	`$SPLUNK_HOME/var/lib/splunk/defaultdb/thaweddb/`
maxDataSize	**Determines rolling behavior, hot to warm.** The maximum size for a hot bucket. When a hot bucket reaches this size, it rolls to warm. This attribute also determines the approximate size for all buckets.	Depends; see indexes.conf.

maxWarmDBCount	**Determines rolling behavior, warm to cold.** The maximum number of warm buckets. When the maximum is reached, warm buckets begin rolling to cold.	300
maxTotalDataSizeMB	**Determines rolling behavior, cold to frozen.** The maximum size of an index. When this limit is reached, cold buckets begin rolling to frozen.	500000 (MB)
frozenTimePeriodInSecs	**Determines rolling behavior, cold to frozen.** Maximum age for a bucket, after which it rolls to frozen.	188697600 (in seconds; approx. 6 years)
coldtoFrozenScript	Script to run just before a cold bucket rolls to frozen.	Default behavior is to log the bucket's directory name and then delete it once it rolls.

Use multiple partitions for index data

Splunk can use multiple disks and partitions for its index data. It's possible to configure Splunk to use many disks/partitions/filesystems on the basis of multiple indexes and bucket types, so long as you mount them correctly and point to them properly from `indexes.conf`. However, we recommend that you use a single high performance file system to hold your Splunk index data for the best experience.

If you do use multiple partitions, the most common way to arrange Splunk's index data is to keep the hot/warm buckets on the local machine, and to put the cold bucket on a separate array or disks (for longer term storage). You'll want to run your hot/warm buckets on a machine with with fast read/write partitions, since most searching will happen there. Cold buckets should be located on a reliable array of disks.

Configure multiple partitions

1. Set up partitions just as you'd normally set them up in any operating system.

2. Mount the disks/partitions.

3. Edit indexes.conf to point to the correct paths for the partitions. You set paths on a per-index basis, so you can also set separate partitions for different indexes. Each index has its own `[<index>]` stanza, where `<index>` is the name of the index. These are the settable path attributes:

- `homePath = <path on server>`
  - This is the path that contains the hot and warm databases for the index.
  - **Caution:** The path must be writable.

- `coldPath = <path on server>`
  - This is the path that contains the cold databases for the index.
  - **Caution:** The path must be writable.

- `thawedPath = <path on server>`
  - This is the path that contains any thawed databases for the index.

Buckets and Splunk administration

When you're administering Splunk, it helps to understand how Splunk stores indexes across buckets. In particular, several admin activities require a good understanding of buckets:

For information on setting a retirement and archiving policy, see "Set a retirement and archiving policy". You can base the retirement policy on either size or age of data.

For information on how to archive your indexed data, see "Archive indexed data". For information on archive signing, see "Configure archive signing". To learn how to restore data from archive, read "Restore archived data".

To learn how to backup your data, read "Back up indexed data". This topic also discusses how to manually roll hot buckets to warm (so that you can then back them up). Also, see "Best practices for backing up" on the Community Wiki.

For information on setting limits on disk usage, see "Set limits on disk usage".

Troubleshoot your buckets

This section tells you how to deal with an assortment of bucket problems. We're starting small, but we'll add new issues as they arise.

Recover invalid hot buckets

A hot bucket becomes an invalid hot (`invalid_hot_<ID>`) bucket when Splunk detects that the metadata files (`Sources.data, Hosts.data, SourceTypes.data`) are corrupt or incorrect. Incorrect data usually signifies incorrect time ranges; it can also mean that event counts are incorrect.

Splunk ignores invalid hot buckets. Data does not get added to such buckets, and they cannot be searched. Invalid buckets also do not count when determining bucket limit values such as `maxTotalDataSizeMB`. This means that invalid buckets do not negatively affect the flow of data through the system, but it also means that they can result in disk storage that exceeds the configured maximum value.

To recover an invalid hot bucket, use the `recover-metadata` command:

1. Make backup copies of the metadata files, `Sources.data, Hosts.data, SourceTypes.data`.

2. Rebuild the metadata from the raw data information:

```
./splunk cmd recover-metadata path_to_your_hot_buckets/invalid_hot_<ID>
```

3. If successful, rename the bucket as it would normally be named.

For more information

For more information on buckets, see "indexes.conf" in this manual and "Understanding buckets" on the Community Wiki.

Configure segmentation to manage disk usage

Configure segmentation to manage disk usage

Segmentation is how Splunk breaks events up during indexing into usable chunks, called tokens. A **token** is a piece of information within an event, such as an error code or a user ID. The level of segmentation you choose can increase or decrease the size of the chunks.

Segmentation can affect indexing and searching speed, as well as disk space usage. You can change the level of segmentation to improve indexing or searching speed, although this is not typically necessary.

You can adjust segmentation rules to provide better index compression or improve the usability for a particular data source. If you want to change Splunk's default segmentation behavior, edit `segmenters.conf`. Once you have set up rules in `segmenters.conf`, tie them to a specific

source, host or sourcetype via `props.conf`. Segmentation modes other than inner and full are not recommended.

Edit all configuration files in `$SPLUNK_HOME/etc/system/local`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`.

Note: You can enable any number of segmentation rules applied to different hosts, sources, and/or sourcetypes in this manner.

There are many different ways you can configure `segementers.conf`, and you should figure out what works best for your data. Specify which segmentation rules to use for specific hosts, sources, or sourcetypes by using props.conf and segmentation. Here are the main types of index-time segmentation:

Full segmentation

Splunk is set to use full segmentation by default. Full segmentation is the combination of inner and outer segmentation.

Inner segmentation

Inner segmentation is the most efficient segmentation setting for both search and indexing, while still retaining the most search functionality. It does, however, make typeahead less comprehensive. Switching to inner segmentation does not change search behavior at all.

To enable inner segmentation, set `SEGMENTATION = inner` for your source, sourcetype, or host in `props.conf`. Under these settings, Splunk indexes smaller chunks of data. For example, `user.id=foo` is indexed as `user id foo`.

Outer segmentation

Outer segmentation is the opposite of inner segmentation. Instead of indexing only the small tokens individually, outer segmentation indexes entire terms, yielding fewer, larger tokens. For example, "10.1.2.5" is indexed as "10.1.2.5," meaning you cannot search on individual pieces of the phrase. You can still use wildcards, however, to search for pieces of a phrase. For example, you can search for "10.1*" and you will get any events that have IP addresses that start with "10.1". Also, outer segmentation disables the ability to click on different segments of search results, such as the `48.15` segment of the IP address `48.15.16.23`. Outer segmentation tends to be marginally more efficient than full segmentation, while inner segmentation tends to be much more efficient.

To enable outer segmentation, set `SEGMENTATION = outer` for your source, sourcetype, or host in `props.conf`. Also for search to behave properly, add the following stanza to `$SPLUNK_HOME/etc/system/local/segmenters.conf`, so that the search system knows to search for larger tokens:

```
[search]
MAJOR = [ ] < > ( ) { } | ! ; , ' " * \n \r \s \t & ? + %21 %26 %2526 %3B %7C %20 %2B %3D -- %2
MINOR =
```

No segmentation

The most space-efficient segmentation setting is to disable segmentation completely. This has significant implications for search, however. By setting Splunk to index with no segmentation, you restrict searches to time, source, host, and sourcetype. You must pipe your searches through the search command to further restrict results. Use this setting only if you do not need any advanced search capabilities.

To disable segmentation, set `SEGMENTATION = none` for your source, sourcetype, or host in `props.conf`. Searches for keywords in this source, sourcetype, or host will return no results. You can still search for indexed fields.

Splunk Web segmentation for search results

Splunk Web has settings for segmentation in search results. These have nothing to do with index-time segmentation. Splunk Web segmentation affects browser interaction and can speed up search results. To set search-result segmentation:

1. Perform a search. Look at the results.

2. Click **Options...** above the returned set of events.

3. In the **Event Segmentation** dropdown box, choose from the available segmentation types: full, inner, outer, or raw. The default is "full".

Configure custom segmentation for a host, source, or source type

Configure custom segmentation for a host, source, or source type

By default, Splunk fully segments events to allow for the most flexible searching. To learn more about segmentation in general, refer to this page about segmentation.

If you know how you want to search for or process events from a specific host, source, or sourcetype, you can configure custom segmentation for that specific type of event. Configuring custom segmentation for a given host, source, or sourcetype improves indexing and search performance and can reduce index storage size.

Configure custom segmentation in props.conf

Configure custom segmentation for events of a host, source, or sourcetype by adding the `SEGMENTATION` and `SEGMENTATION-<segment selection>` attributes to the appropriate stanza in `props.conf`. Assign values to the attributes using rules for index-time and search-time (Spunk Web) segmentation defined in `segmenters.conf`.

Add your stanza to `$SPLUNK_HOME/etc/system/local/props.conf`. Specify the following attribute/value pairs:

```
[<spec>]
SEGMENTATION = <segmenter>
```

SEGMENTATION-<segment selection> = <segmenter>

[<spec>] can be:

- <sourcetype>: A sourcetype in your event data.
- host::<host>: A host value in your event data.
- source::<source>: A source of your event data.

SEGMENTATION = <segmenter>

- This specifies the segmentation rule ("segmenter") from segmenters.conf to use at index time.

SEGMENTATION-<segment selection> = <segmenter>

- This setting affects how search results appear in Splunk Web; it does not change the index-time segmentation.
- This specifies that Splunk Web should use the specified segmenter (from segmenters.conf) for the given <segment selection> choice. The <segment selection> choices appear as segmentation types that the user can select when viewing search results in Splunk Web. Look here for more information.
- Default <segment selection> choices are: all, inner, outer, and raw.
- 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 first need to edit the Splunk Web UI, which you probably will not want to attempt to do. You can, however, change the segmenter that a given <segment selection> calls.

<segmenter>

- This is a segmentation rule defined in segmenters.conf.
- Pre-defined default rules include: inner, outer, none, and full.
- You can create your own custom rule by editing $SPLUNK_HOME/etc/system/local/segmenters.conf.
- For more information on configuring segmenters.conf, see this page.

Example

The following example can increase search performance and reduce the size of syslog events in your index.

Add the following to the [syslog] source type stanza in props.conf:

```
[syslog]
SEGMENTATION = inner
SEGMENTATION-all = inner
```

This changes the segmentation of all events that have a sourcetype of syslog to inner segmentation, both at index time (through the SEGMENTATION attribute) and at search time in Splunk Web (through the SEGMENTATION-<segment selection> attribute).

Note: You must restart Splunk to apply changes to Splunk Web search-time segmentation, and you must re-index your data to apply changes to index-time segmentation.

Move an index

Move an index

You can move an entire index from one location to another. Here's how to move an index:

For *nix Users:

1. Make sure the target file system has enough space - at least 1.2 times the size of the total amount of raw data you plan to index.

2. Make sure the target directory has the correct permissions, so that the `splunkd` process can write to files there:

```
# mkdir /foo/bar
# chown splunk /foo/bar/
# chmod 755 /foo/bar/
```

3. When the new index home is ready, stop the server (if it is running) from Splunk's CLI. Navigate to the `$SPLUNK_HOME/bin/` directory and use the command:

```
# ./splunk stop
```

4. Copy the existing index filesystem to its new home:

```
# cp -r $SPLUNK_DB/* /foo/bar/
```

5. Edit `./etc/splunk-launch.conf` to reflect the new index directory. Change the `SPLUNK_DB` attribute in that file to point to your new index directory:

```
SPLUNK_DB=/foo/bar
```

Note: Ensure that the path `$SPLUNK_HOME/var/lib/splunk/searches` exists. Splunk saves a small amount of index data here and without it your index may appear to vanish.

6. Start the server:

```
# ./splunk start
```

The Splunk Server picks up where it left off, reading from, and writing to, the new copy of the index.

For Windows Users:

1. Make sure the target drive or directory has enough space available.

Caution: Using mapped network drives for index stores is strongly discouraged, and not supported.

2. From a command prompt, go to your target drive and make sure the target directory has the correct permissions, so that the `splunkd` process can write to files there:

```
C:\Program Files\Splunk> D:
D:\> mkdir \new\path\for\index
D:\> cacls D:\new\path\for\index /T /E /G <the user Splunk runs as>:F
```

Note 1: For more information about determining the user Splunk runs as, review this topic on installing Splunk on Windows.

Note 2: Windows Vista, 7, Server 2003 and Server 2008 users may also use `icacls` to ensure directory permissions are correct; this Microsoft TechNet article gives information on specific command-line arguments.

3. Stop the Splunk server (if it is running) from Splunk's CLI. Navigate to the `%SPLUNK_HOME%\bin` directory and use the command:

```
> .\splunk stop
```

Note: You can also use the Services control panel to stop the Splunkd and SplunkWeb services.

4. Copy the existing index filesystem to its new home:

```
> xcopy C:\Program Files\Splunk\var\lib\splunk\*.* D:\new\path\for\index /s /e /v /o /k
```

5. Use Notepad or your favorite text editor to change `%SPLUNK_HOME%\etc\splunk-launch.conf` to reflect the new index directory. Change the `SPLUNK_DB` attribute in that file to point to your new index directory:

```
SPLUNK_DB=D:\new\path\for\index
```

Note 1: If the line in the configuration file that contains the SPLUNK_DB attribute has a pound sign (#) as its first character, the line is commented out, and the # needs to be removed.

Note 2: Ensure that the path `%SPLUNK_HOME%\var\lib\splunk\searches` exists. Splunk saves a small amount of index data here and without it your index may appear to vanish.

6. Start the server:

```
> .\splunk start
```

The Splunk Server picks up where it left off, reading from, and writing to, the new copy of the index.

Caution: Do not try to break up and move parts of an index manually. If you must subdivide an existing index, contact Splunk Support for assistance.

Remove indexed data from Splunk

Remove indexed data from Splunk

You can remove data from indexes in two ways:

- Delete events from future searches with the `delete` operator.
- Remove all data from one or more indexes with the CLI `clean` command.

Caution: Removing data is irreversible. Use caution when choosing what events to remove from searches, or what data to remove from your Splunk indexes. If you want to get your data back, you must re-index the applicable data source(s).

Delete data from future searches with the "delete" operator

Splunk provides the special operator `delete` to delete data from future searches. Before using the `delete` operator, read this section carefully.

Who can delete?

The `delete` operator can only be accessed by a user with the "delete_by_keyword" **capability**. By default, Splunk ships with a special **role**, "can_delete" that has this capability (and no others). The admin role does not have this capability by default. Splunk recommends you create a special user that you log into when you intend to delete index data.

For more information, refer to "Add users and assign roles" in this manual.

How to delete

To use the `delete` operator, run a search that returns the events you want deleted. Make sure that this search returns **ONLY** events you want to delete, and no other events.

For example, if you want to remove the events you've indexed from a source called `/fflanda/incoming/cheese.log` so that they no longer appear in searches, do the following:

1. Disable or remove that source so that it no longer gets indexed.

2. Search for events from that source in your index:

```
source="/fflanda/incoming/cheese.log"
```

3. Look at the results to confirm that this is the data you want to delete.

4. Once you've confirmed that this is the data you want to delete, pipe the search to `delete`:

```
source="/fflanda/incoming/cheese.log" | delete
```

See the page about the delete operator in the Search Reference Manual for more examples.

Piping a search to the `delete` operator marks all the events returned by that search so that they are never returned by any future search. No user (even with admin permissions) will be able to see this data when searching with Splunk.

Note: Piping to `delete` does not reclaim disk space.

The `delete` operator also does not update the metadata of the events, so any metadata searches will still include the events although they are not searchable. The main **All indexed data** dashboard will still show event counts for the deleted sources, hosts, or sourcetypes.

Remove data from indexes with the CLI "clean" command

To delete index data permanently from your disk, use the CLI `clean` command. This command completely deletes the data in one or all indexes, depending on whether you provide an `<index_name>` argument. Typically, you run `clean` before re-indexing all your data.

How to use the "clean" command

Here are the main ways to use the `clean` command:

- To access the help page for `clean`, type:

  ```
  ./splunk help clean
  ```

- To permanently remove event data from **all indexes**, type:

  ```
  ./splunk clean eventdata
  ```

- To permanently remove event data from **a single index**, type:

  ```
  ./splunk clean eventdata <index_name>

  where <index_name> is the name of the targeted index.
  ```

- Add the `-f` parameter to force `clean` to skip its confirmation prompts.

Examples

Note: You must stop Splunk before you run the `clean` command:

```
./splunk stop
```

This example removes event data from all indexes:

```
./splunk clean eventdata
```

This example removes event data from the `_internal` index and forces Splunk to skip the confirmation prompt:

```
./splunk clean eventdata _internal -f
```

Optimize indexes

Optimize indexes

While Splunk is indexing data, one or more instances of the `splunk-optimize` process will run intermittently, merging index files together to optimize performance when searching the data. The

`splunk-optimize` process can use a significant amount of cpu, but should not consume it indefinitely, only for a short amounts of time. You can alter the number of concurrent instances of `splunk-optimize` by changing the value set for `maxConcurrentOptimizes` in `indexes.conf`, but this is not typically necessary.

`splunk-optimize` should only run on hot **buckets**. You can run it on warm buckets manually, if you find one with a larger number of `.tsidx` files (more than 25):

```
./splunk-optimize <directory>
```

If `splunk-optimize` does not run often enough, search efficiency will be affected.

For more information on buckets, see "How Splunk stores indexes".

Define alerts

How alerting works

How alerting works

Alerts are searches you've configured to run on a schedule and send you their results. Use alerts to notify you of changes in your data, network infrastructure, file system or other devices you're monitoring. Alerts can be sent via email or RSS, or trigger a shell script. You can turn any saved search into an alert.

An alert is comprised of:

- a schedule for performing the search
- conditions for triggering an alert
- actions to perform when the triggering conditions are met

Enable alerts

Set up an alert at the time you create a saved search, or enable an alert on any existing saved search. Configure both basic and advanced conditional alerts for searches by:

- Scheduling and defining alerts for saved searches through Splunk Web (if you have permission to edit them).
- Entering or updating saved search configurations in `savedsearches.conf`. For more information, see "Set up alerts in savedsearches.conf", in this chapter.

Specify default email alert action settings

To specify the mail host, email format, subject, sender, and whether or not the results of the alert should be included inline:

- In Splunk Web, click **Manager > Email alert settings** and specify your choices.
- Click **Save**.

All alerts that use the email alert action now use these settings.

You can also set default settings for alert actions (including scripted alerts) by making changes directly to alert_actions.conf.

PDF report settings

On the Email alert settings page, select **Use PDF report server** to open the PDF report settings section. This is where you enable the ability to have .pdf printouts of report results sent as attachments with alert emails.

Note: You must have the PDF Printer app set up on a central Linux host before you can enable the PDF printing functionality here. For more information see "Configure PDF printing for Splunk Web" in the Installation manual.

Scripted alerts

Alerts can also trigger shell scripts. When you configure an alert, specify a script you've written. You can use this feature to send alerts to other applications. Learn more about configuring scripted alerts.

You can use scripted alerts to send syslog events, or SNMP traps.

Considerations

When configuring alerts, keep the following in mind:

- Too many alerts/saved searches running at once may slow down your system -- depending on the hardware, 20-30 alerts running at once should be OK. If the searches your alerts are based on are complex, you should make the interval longer and spread the searches out more.
- Set a time frame for alerts that makes sense -- if the search takes longer than 4-5 minutes to run, don't set it to run every five minutes.
- You must have a mail server running on the LAN that the Splunk server can connect to. Splunk does not authenticate against the mail server.
- Read more about best practices for alert configuration on the Splunk Community Wiki, here.

Set up alerts in savedsearches.conf

Set up alerts in savedsearches.conf

Configure alerts with savedsearches.conf. Use the `$SPLUNK_HOME/etc/system/README/savedsearches.conf.example` as an example, or create your own `savedsearches.conf`. Edit this file in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For more information on configuration files in general, see how configuration files work.

Follow these steps:

1. Create a saved search.

2. Schedule the search.

3. Define alert conditions.

4. Configure alert actions.

You can set up an alert at the time you create a saved search, or add the alert configurations to your saved search stanza later.

Note: You must have email enabled on your Splunk server for alerts to be sent out. Alternately, your Splunk server must be able to contact your email server. Configure email settings in Manager.

Create a saved search

First, set up a saved search, either via Splunk Web or savedsearches.conf.

Schedule the search

Next, schedule your search. This means your search runs on a schedule that you specify. For example, you can arrange to have Splunk run your search every hour, or every day at midnight.

To schedule a search via `savesearches.conf`, add the following attribute/value pairs to your saved search stanza:

```
userid = <integer>
```

- UserId of the user who created this saved search.
  - Splunk needs this information to log who ran the search, and create editing capabilities in Splunk Web.
- Possible values: Any Splunk user ID.
- User IDs are found in `$SPLUNK_HOME/etc/passwd`.
  - Look for the first number on each line, right before the username.
  - For example `2:penelope....`

```
enableSched = < 0 | 1 >
```

- Set this to 1 to enable schedule for search
- Defaults to 0.

```
cron_sched = <cron string>
```

- The cron schedule used to execute the search.
- For example, `*/5 * * * *` causes the search to execute every five minutes.

Note Cron scheduling 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're staggered over time. This reduces system load. Running all of them (*/20) every 20 minutes means they would all launch at hh:00 (20, 40) and might slow your system every 20 min.

```
max_concurrent = <integer>
```

- The maximum number of concurrent instances of this search the scheduler is allowed to run.
- Defaults to 1.

Set up alert conditions

Next, define alert conditions for the scheduled search. When Splunk runs a scheduled search, these are the the conditions that trigger an alert action (such as an email) when they are met.

Alerts fit into two categories: basic conditional alerts and advanced conditional alerts.

- **Basic conditional alerts** trigger alert actions when set thresholds in the number of events, sources, or hosts in your results are exceeded.
- **Advanced conditional alerts** are based on the results of a conditional search that is evaluated against the results of the scheduled search. If the conditional search returns one or more events, the event is triggered.

Define a basic conditional alert

Define a threshold number of events, sources, or hosts. If the alert conditions are met when the search is run, Splunk notifies you via email or triggers a shell script. You can also set `counttype = always` if you want the alert action (such as an email) to be triggered each time the scheduled search runs.

`counttype = <string>`

- Set the type of count for alerting.
- Possible values: `number of events`, `number of hosts`, `number of sources`, and `always`.
- Used in conjunction with the `relation` and `quantity` attributes, except when set to `always`.
- Use `counttype = always` to trigger the alert action each time the scheduled search is run.

`relation = <string>`

- How to compare against counttype.
- Possible values: greater than, less than, equal to, drops by, rises by.

`quantity = <integer>`

- Number to compare against the given counttype.

So if you have the following:

```
counttype = number of events
relation = rises by
quantity = 25
```

Splunk alerts you if your search results have risen by 25 since the last time the search ran.

For more information about configuring alert actions, see the "Configure alert actions" subtopic, below.

Define an advanced conditional alert

If you'd rather define an advanced conditional alert, you use the `alert_condition` attribute in place of `counttype`, `relation`, and `quantity`.

```
alert_condition = <string>
```

- In `<string>`, enter a search. Splunk will evaluate this secondary search on the artifacts of the saved search to determine whether to trigger an alert action.
- Alert actions are triggered if this secondary search yields a non-empty search result list.

For an in-depth discussion of a use case for advanced conditional alerting over basic conditional alerting, see "Set alert conditions for scheduled searches" in the User Manual. This topic discusses alert setup using Manager, but the underlying principles are the same.

For more information about configuring alert actions, see the following subtopic, "Configure alert actions."

Configure alert actions

You can configure three different kinds of alert actions--actions that happen when alert conditions are met--for your scheduled searches. These alert actions are notification by email, notification by RSS, and the triggering of a shell script.

To enable or disable an alert action for a particular scheduled, alerted search, add the following to the search definition:

```
action.<action_name> = 0 | 1
```

- Indicates whether the alert action is enabled or disabled for a particular saved search. Set to `0` (disabled) by default.
- `action_name` can be `email`, `script`, or `rss`.

Global defaults for all alert actions are configured in `alert_actions.conf` (or via Splunk Manager). You can override these defaults at the individual search level in `savedsearches.conf`. If you don't need to override the alert action defaults, all you need to do is indicate which alert actions are enabled for a given scheduled search (see above).

To set a parameter for an alert action, the syntax is as follows:

```
action.<action_name>.<parameter> = <value>
```

The parameter options for each <code><action_name> are defined in the following sections.

Notification by email

Use the `email` action to have Splunk contact stakeholders when the scheduled search triggers an alert:

```
action.email = 1
```

The `email` action has a number of parameters. Defaults can be set for all of these parameters in `alert_actions.conf`, *with the exception* of the `action.email.to` parameter, which should be set for each scheduled search that uses the email alert action.

`action.email.to = <email list>`

- The email addresses to which Splunk will send the email, arranged in a comma-delimited list.
- This parameter is *not* set at the `alert_actions.conf` level. You must define it for every email alert action that you configure.

`action.email.from = <email address>`

- The email address that is used as the sender's address.
- Default is `splunk@$LOCALHOST` (or whatever is set for `from` in `alert_actions.conf`).

`action.email.subject = <string>`

- The subject of the alert email.
- Default is `SplunkAlert-<savedsearchname>` (or whatever is set for `subject` in `alert_actions.conf`).

`action.email.sendresults = <bool>`

- Specify whether to include the search results in the email. The results can be attached or included in the body of the email (see the `action.email.inline` parameter, below).
- Default is `false` (or whatever is set for `sendresults` in `alert_actions.conf`).
- **Note**: When you are using an advanced conditional alert, be aware that only the results of the *original* search are included with the email. The results of the triggering conditional search are discarded

`action.email.inline = <true | false>`

- Specify whether the search results are included in the body of the alert mail.
- Default is `false` (or whatever is set for `inline` in `alert_actions.conf`).

`action.email.mailserver = <string>`

- The address of the MTA server that sends the alert emails.
- Default is `$LOCALHOST` (or whatever is set for `mailserver` in `alert_actions.conf`).

`action.email.preprocess_results = <search-string>`

- An optional search string to preprocess results before emailing them. Usually one would set this up to filter out unwanted internal fields.
- Default is an empty string (or whatever is set for `preprocess_results` in `alert_actions.conf`).

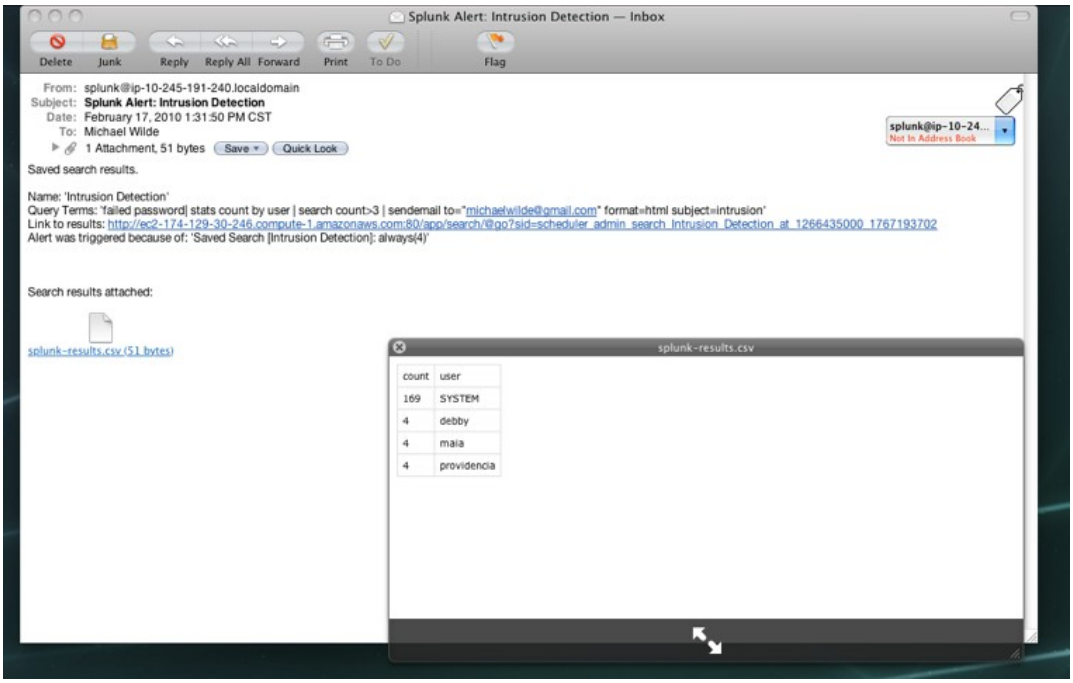
Note: You can also arrange to have .pdf printouts of dashboards delivered by email on a set schedule. For more information, see "Schedule delivery of dashboard PDF printouts via email" in this

manual.

There are settings for this feature in `alert_actions.conf`. For example, you can identify the URL of the PDF report server, and the report paper size and orientation.

Important: Use of the .pdf printout feature requires the setup of the PDF Printer app on a central Linux host. If you don't have this set up, contact a system administrator. For more information see "Configure PDF printing for Splunk Web" in the Installation manual.

The following is an example of what an email alert looks like:



Create an RSS feed

Use the `rss` action to have Splunk alert you via RSS when the scheduled search triggers an alert:

```
action.rss = 1
```

Whenever the alert conditions are met for a scheduled search that has **Create an RSS feed** selected, Splunk sends a notification out to its RSS feed. The feed is located at `http://[splunkhost]:[port]/rss/[saved_search_name]`. So, let's say you're running a search titled "errors_last15" and have a Splunk instance that is located on `localhost` and uses port 8000, the correct link for the RSS feed would be `http://localhost:8000/rss/errors_last15`.

You can also access the RSS feed for a scheduled search through the Searches and reports page in Manager. If a scheduled search has been set up to provide an RSS feed for alerting searches, when you look it up on the Searches and reports page, you will see a RSS symbol in the **RSS feed** column:

You can click on this symbol to go to the RSS feed.

Note: The RSS feed for a scheduled search will not display any searches until the search has run on its schedule *and* the alerting conditions that have been defined for it have been met. If you set the search up to alert each time it's run (by setting **Perform actions** to *always*), you'll see searches in the RSS feed after first time the search runs on its schedule.

Warning: The RSS feed is exposed to any user with access to the webserver that displays it. Unauthorized users can't follow the RSS link back to the Splunk application to view the results of a particular search, but they can see the summarization displayed in the RSS feed, which includes the name of the search that was run and the number of results returned by the search.

Here's an example of the XML that generates the feed:

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
    <channel>
        <title>Alert: errors last15</title>
        <link>http://localhost:8000/app/search/@go?sid=scheduler_Z2d1cHRh</link>
        <description>Saved Searches Feed for saved search errors last15</description>
        <item>
            <title>errors last15</title>
            <link>http://localhost:8000/app/search/@go?sid=scheduler_Z2d1cHRh</link>
            <description>Alert trigger: errors last15, results.count=123 </description>
            <pubDate>Mon, 01 Feb 2010 12:55:09 -0800</pubDate>
        </item>
    </channel>
</rss>
```

Trigger a shell script

Use the `script` action to have Splunk run a shell script when the scheduled search triggers an alert:

```
action.script = 1
```

The `script` action has a `filename` parameter which is usually defined at the individual search level, although a default filename can also be set in `alert_actions.conf`:

```
action.script.filename = <script filename>
```

- The filename of the shell script that you want Splunk to run. The script should live in `$SPLUNK_HOME/bin/scripts/`.

Example - Basic conditional alert configuration

This example is for a saved search titled "sudoalert." It runs a search for events containing the term "sudo" on a 12 minute interval. If a scheduled "sudoalert" run results in greater than 10 events, alert actions are triggered that send the results via email and post them to an RSS feed.

```
[sudoalert]
search = sudo
counttype = number of events
enableSched = 1
```

```
schedule = */12 * * * *
quantity = 10
relation = greater than
action.email = 1
action.email.to = me@work.org
action.email.from = splunk@work.org
action.email.subject = Sudo Alert!
action.email.mailserver = mail@work.org
action.rss = 1
```

Enable summary indexing

Summary indexing is an additional kind of alert action that you can configure for any scheduled search. You use summary indexing when you need to perform analysis/reports on large amounts of data over long timespans, which typically can be quite time consuming, and a drain on performance if several users are running similar searches on a regular basis.

With summary indexing, you define a scheduled search that computes sufficient statistics (a summary) for events covering a time slice. Each time Splunk runs the search it saves the results into a summary index that you've designated. You can then search and report on this smaller (and thus faster) summary index instead of working with the much larger dataset that the summary index is based on.

Note: Do not attempt to set up a summary index until you have read and understood "Use summary indexing for increased reporting efficiency" in the Knowledge Manager manual.

For more information about configuring summary index searches in `savedsearches.conf`, see "Configure summary indexes" in the Knowledge Manager Manual.

Configure scripted alerts

Configure scripted alerts

Configure scripted alerts with savedsearches.conf. Use the `$SPLUNK_HOME/etc/system/README/savedsearches.conf.example` as an example, or create your own `savedsearches.conf`. Edit this file in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For more information on configuration files in general, see "About configuration files".

Script options

Your alert can trigger a shell script, which must be located in $SPLUNK_HOME/bin/scripts. Use the following attribute/value pairs:

```
action_script = <string>
```

- Your search can trigger a shell script.
- Specify the name of the shell script to run.
- Place the script in $SPLUNK_HOME/bin/scripts.

- Command line arguments passed to the script are:
- $0 = script name.
- $1 = number of events returned.
- $2 = search terms.
- $3 = fully qualified query string.
- $4 = name of saved Splunk.
- $5 = trigger reason (i.e. "The number of events was greater than 1").
- $6 = Browser URL to view the saved search.
- $7 = This option has been deprecated and is no longer used
- $8 = file where the results for this search are stored (contains raw results).

If you want to run a script written in a different language (e.g. PERL, Python, VBScript) you must specify the interpreter you want Splunk to use in the first line of your script, following the #!. For example:

to run a PERL script:

```
---- myscript.pl ----
#!/path/to/perl
......
......
```

to use Python to interpret the script file:

```
---- myscript.py -----
#!/path/to/python
.....
.....
```

For an example on how scripts can be configured to work with alerts, see send SNMP traps.

Example

You can configure Splunk to send alerts to syslog. This is useful if you already have syslog set up to send alerts to other applications, and you want Splunk's alerts to be included.

Check the Splunk Wiki for information about the best practices for using UDP when configuring Syslog input.

Write a script that calls `logger` (or any other program that writes to syslog). Your script can call any number of the variables your alert returns.

Create the following script and make it executable:

```
logger $5
```

Put your script in `$SPLUNK_HOME/bin/scripts`.

Now write an alert that calls your script. See Set Up Alerts for information on alert configuration. Configure the alert to call your script by specifying the path in the **Trigger shell script** field of the alert.

Edit your saved search to call the script. If your script is in `$SPLUNK_HOME/bin/scripts` you don't have to specify the full path.

This logs the trigger reason to syslog:

```
Aug 15 15:01:40 localhost logger: Saved Search [j_myadmin]: The number of events(65) was greate
```

Troubleshoot

Check out this excellent topic on troubleshooting alert scripts on the Splunk Community Wiki.

Send SNMP traps to other systems

Send SNMP traps to other systems

You can use Splunk as a monitoring tool to send SNMP alerts to other systems such as a Network Systems Management console.

If you're interested in sending SNMP traps on Windows, check this Community Wiki topic.

Configuration

Requirements

- Perl is required to run the script below.
- Net-SNMP package is required in order to use the `/usr/bin/snmptrap` command - if you have another way of sending an SNMP trap from a shell script then modify as needed.
- Admin access to the `$SPLUNK_HOME/bin/scripts` directory of your Splunk install.
- For security reasons, scripts must reside in $SPLUNK_HOME/bin/scripts.

Create shell script

- Create `traphosts.pl` script in your `$SPLUNK_HOME/bin/scripts` directory.
    - For security reasons, scripts must reside in this directory. Create the directory if it doesn't already exist.
    - Copy the code below into `sendsnmptrap.pl`.
- `chmod +x sendsnmptrap.pl` to make it executable.
- Change the `Host:Port` of the SNMP trap handler, paths to external commands `splunk` and `snmptrap`, and the user/password if necessary.
- The perl script will work on MS Windows systems with Perl. However, on some Windows systems, perl may not be installed, or perl scripts may not be configured to be directly executable via Splunk. In these cases, you may find it easier to send SNMP traps using a Windows CMD script.

```
#!/usr/bin/perl
#
# sendsnmptrap.pl: A script to for Splunk alerts to send an SNMP trap.
#
```

```
# Modify the following as necessary for your local environment
#
$hostPortSNMP = "qa-tm1:162"; # Host:Port of snmpd or other SNMP trap handler
$snmpTrapCmd = "/usr/bin/snmptrap"; # Path to snmptrap, from http://www.net-snmp.org
$TRAPOID = "1.3.6.1.4.1.27389.1.2"; # Object IDentifier for traps/notifications
$OID = "1.3.6.1.4.1.27389.1.1"; # Object IDentifier for objects, Splunk Enterprise OID is 27389
# Parameters passed in from the alert.
# $1-$9 is the positional parameter list. $ARGV[0] starts at $1 in Perl.
$searchCount = $ARGV[0]; # $1 - Number of events returned
$searchTerms = $ARGV[1]; # $2 - Search terms
$searchQuery = $ARGV[2]; # $3 - Fully qualified query string
$searchName = $ARGV[3]; # $4 - Name of saved search
$searchReason = $ARGV[4]; # $5 - Reason saved search triggered
$searchURL = $ARGV[5]; # $6 - URL/Permalink of saved search
$searchTags = $ARGV[6]; # $7 - Always empty as of 4.1
$searchPath = $ARGV[7]; # $8 - Path to raw saved results in Splunk instance (advanced)

# Send trap, with the the parameter list above mapping down into the OID.
$cmd = qq/$snmpTrapCmd -v 2c -c public $hostPortSNMP '' $TRAPOID
$OID.1 i $searchCount $OID.2 s "$searchTerms" $OID.3 s "$searchQuery" $OID.4 s
"$searchName" $OID.5 s "$searchReason" $OID.6 s "$searchURL" $OID.7 s
"$searchTags" $OID.8 s "$searchPath"/;
 system($cmd);
```

Configure your alert to call a shell script

- Create a saved search. Read about setting up saved searches for more information.
- Turn your saved search into an alert. Read about setting alert conditions from scheduled searches for more information.
- Set up your alert so that it calls your shell script by specifying the name of your script which resides in $SPLUNK_HOME/bin/scripts:

Here is an example of the script running, including what it returns:

```
[root@qa-tm1 ~]# snmptrapd -f -Lo
2007-08-13 16:13:07 NET-SNMP version 5.2.1.2 Started.
2007-08-13 16:14:03 qa-el4.splunk.com [172.16.0.121] (via UDP: [172.16.0.121]:32883) TRAP, SNMP
        SNMPv2-SMI::enterprises.27389.1 Warm Start Trap (0) Uptime: 96 days, 20:45:08.35
        SNMPv2-SMI::enterprises.27389.1.1 = INTEGER: 7 SNMPv2-
SMI::enterprises.27389.1.2 = STRING: "sourcetype::syslog" SNMPv2-
SMI::enterprises.27389.1.3 = STRING: "search sourcetype::syslog starttime:12/31
/1969:16:00:00 endtime::08/13/2007:16:14:01" SNMPv2-SMI::enterprises.27389.1.4
= STRING: "SyslogEventsLast24" SNMPv2-SMI::enterprises.27389.1.5 = STRING:
"Saved Search [SyslogEventsLast24]: The number of hosts(7) was greater than 1"
SNMPv2-SMI::enterprises.27389.1.6 = STRING: "http://qa-el4:18000/?q=sourcetype
%3a%3asyslog%20starttimeu%3a%3a0%20endtimeu%3a%3a1187046841" SNMPv2-
SMI::enterprises.27389.1.7 = STRING: "/home/tet/inst/splunk/var/run/splunk
/SyslogEventsLast24"
2007-08-13 16:14:15 NET-SNMP version 5.2.1.2 Stopped.
```

Set up backups and retention policies

What you can back up

What you can back up

Splunk data falls into two major categories:

- Indexed event data, including both the compressed raw data and the indexes that access it
- Configuration data, including user data

How much space you will need

How much space you will need

This topic describes how to estimate the size of your Splunk index and associated data so that you can plan your storage capacity requirements.

When Splunk indexes your data, the resulting data falls into two categories: the compressed, persisted raw data and the indexes that point to this data. With a little experimentation, you can estimate how much disk space you will need.

Typically, the compressed, persisted data amounts to approximately 10% of the raw data that comes into Splunk. The associated indexes range in size anywhere from 10% to 110% of the data they access. This value is affected strongly by the number of unique terms in the data. Depending on the data's characteristics, you might want to tune your segmentation settings. For an introduction to how segmentation works and how it affects index size, you can also watch this video on segmentation by one of Splunk's lead developers.

The best way to get an idea of your space needs is to experiment by installing a copy of Splunk and indexing a representative sample of your data, and then checking the sizes of the resulting directories in `defaultdb`.

To do this, first index your sample. Then:

1. Go to `$SPLUNK_HOME/var/lib/splunk/defaultdb/db`.

2. Run `du -shc hot_v*/rawdata` to determine the size of the compressed, persisted raw data. Typically, this amounts to about 10% of the size of the original sample data set.

3. Run `du -ch hot_v*` and look at the last `total` line to see the size of the index.

4. Add the two values together.

This is the total size of the index and associated data for the sample you indexed. You can now use this to extrapolate the size requirements for your Splunk index and `rawdata` directories over time.

Back up indexed data

Back up indexed data

This topic discusses backing up Splunk indexed data. It first gives an overview of how your indexed data moves through Splunk, then describes a basic backup strategy based on common or default Splunk index configurations. Finally, it provides options for setting or changing the retirement policy for your Splunk index data.

The default values and policies described in this topic are set in indexes.conf. If you have a more complex index configuration, or have unusual data volumes, you can refer there for detailed information and options. Before modifying any configuration file, read "About configuration files".

For more information on backing up indexed data, see "Best practices for backing up" on the Community Wiki.

For information on setting a data retirement and archiving policy, see "Set a retirement and archiving policy".

How data ages

When Splunk is indexing, the data moves through a series of stages based on policies that you define. At a high level, the default behavior is as follows:

When data is first indexed, it is put into a "hot" database, or **bucket**.

The data remains in the hot bucket until the policy conditions are met for it to be reclassified as "warm" data. This is called "rolling" the data into the warm bucket. By default, this happens when a hot bucket reaches a specified size or age. When a hot bucket is rolled, its directory is renamed, and it becomes a warm bucket. It is safe to back up the warm buckets.

Next, when you reach a specified number of warm buckets, the oldest bucket becomes a cold bucket, thus maintaining a constant number of warm buckets. (If your `colddb` directory is located on another fileshare, the buckets are moved there and deleted from the warm db directory.) The default number of warm buckets is 300.

Finally, at a time based on your defined policy requirements, the bucket will roll from cold to "frozen". By default, Splunk deletes frozen buckets. If you need to archive or otherwise preserve the data, you can provide a script that performs actions on the bucket prior to deletion.

Summary:

- **hot bucket** - Currently written to; non-incrementally changing; do not back this up.
- **warm bucket** - Rolled from hot; added to incrementally; can be safely backed up; consists of multiple warm buckets.
- **cold bucket** - Rolled from warm; buckets are moved to another location.
- **frozen bucket** - Default policy is to delete.

For detailed information on how buckets work and where they are stored, see "How Splunk stores indexes".

Choose your backup strategy

The general recommendation is to schedule backups of your warm buckets regularly, using the incremental backup utility of your choice.

Hot buckets can only be backed up by taking a snapshot of the files, using a tool like VSS (on Windows/NTFS), ZFS snapshots (on ZFS), or a snapshot facility provided by the storage subsystem. If you do not have such a facility available, the data within the hot bucket can only be backed up after it has rolled to a warm bucket.

Splunk rolls a hot bucket to a warm bucket based on the policy defined in `indexes.conf`. By default, the main index rolls a hot bucket when it reaches a certain size. (While it is possible to force a roll of a hot bucket to a warm bucket, this is not recommended, as each forced roll permanently decreases search performance over the data. In cases where hot data needs to be backed up, a snapshot backup is the preferred method.)

You can set retirement and archiving policy by controlling the size of indexes or buckets or the age of the data.

The sizes, locations, and ages of index files are set in `indexes.conf`. See "How Splunk stores indexes" for detailed information on buckets and `indexes.conf`.

Caution: All index locations must be writable.

Recommendations for recovery

If you experience a non-catastrophic disk failure (for example you still have some of your data, but Splunk won't run), Splunk recommends that you move the index directory aside and restore from a backup rather than restoring on top of a partially corrupted datastore. Splunk will automatically create hot directories on startup as necessary and resume indexing. Monitored files and directories will pick up where they were at the time of the backup.

Rolling buckets manually from hot to warm

To roll the buckets of an index manually from hot to warm, use the following command, replacing `<index_name>` with the name of the index you want to roll:

From the CLI

```
./splunk _internal call /data/indexes/<index_name>/roll-hot-buckets ?auth
<admin_username>:<admin_password>
```

From the search bar

This has been deprecated and cannot be used from the search bar any longer

Back up configuration information

Back up configuration information

All Splunk's configuration information is contained in **configuration files**. To back up the set of cofiguration 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, including the structure of the underlying directories, read "About configuration files".

Set a retirement and archiving policy

Set a retirement and archiving policy

Configure data retirement and **archiving** policy by controlling the size of indexes or the age of data in indexes.

Splunk stores indexed data in **buckets**. For a discussion of buckets and how Splunk uses them, see "How Splunk stores indexes".

Splunk index buckets go through four stages of retirement. When indexed data reaches a frozen state, Splunk deletes it. (Splunk deletes **all** frozen data by default. You must specify an archiving script to avoid losing frozen data.)

Retirement stage	Description	Searchable?
Hot	Open for writing. One or more hot buckets for each index.	Yes.
Warm	Data rolled from hot. There are many warm buckets.	Yes.
Cold	Data rolled from warm. There are many cold buckets.	Yes.
Frozen	Data rolled from cold. Eligible for deletion.	N/A: Splunk deletes frozen data by default.

Splunk defines the sizes, locations, and ages of indexes and their buckets in indexes.conf.

Caution: When you change your data retirement and archiving policy settings, Splunk deletes old data without prompting you.

Edit a copy of `indexes.conf` in `$SPLUNK_HOME/etc/system/local/`, or in your own custom application directory in `$SPLUNK_HOME/etc/apps/`. Do not edit the copy in `$SPLUNK_HOME/etc/system/default`. For information on configuration files and directory

locations, see "About configuration files".

Note: To configure data, all index locations must be writable.

Remove files beyond a certain size

If an index grows bigger than a specified maximum size, the oldest data is rolled to **frozen**, which means it gets immediately deleted unless you have created a script to archive the data, as described in "Archive indexed data". The default maximum size for an index is 500000 MB. To change the maximum size, edit this line in `indexes.conf`:

```
maxTotalDataSizeMB = <non-negative number>
```

For example:

```
[main]
maxTotalDataSizeMB = 2500000
```

Note: Make sure that the data size you specify for `maxTotalDataSizeMB` is expressed in megabytes.

Restart Splunk for the new setting to take effect. Depending on how much data there is to process, it can take some time for Splunk to begin to move buckets out of the index to conform to the new policy. You might see high CPU usage during this time.

Remove data beyond a certain age

Splunk ages out data by buckets. Specifically, when the most recent data in a particular bucket reaches the configured age, the entire bucket is rolled.

Splunk also rolls buckets when the reach a maximum size. If you are indexing a large volume of events, bucket size is less a concern for retirement policy because the buckets will fill quickly. You can reduce bucket size by setting a smaller `maxDataSize` in `indexes.conf` so they roll faster. But note that it takes longer to search more small buckets than fewer large buckets. To get the results you are after, you will have to experiment a bit to determine the right size. Due to the structure of the index, there isn't a direct relationship between time and data size.

To remove data beyond a specified age, set `frozenTimePeriodinSecs` in `indexes.conf` to the number of seconds to elapse before the data gets erased. The default value is 188697600 seconds, or approximately 6 years. This example configures Splunk to cull old events from its index when they become more than 180 days (15552000 seconds) old:

```
[main]
frozenTimePeriodInSecs = 15552000
```

Note: Make sure that the time you specify for `frozenTimePeriodInSecs` is expressed in seconds.

Restart Splunk for the new setting to take effect. Depending on how much data there is to process, it can take some time for Splunk to begin to move buckets out of the index to conform to the new policy. You might see high CPU usage during this time.

I changed the archive policy and restarted but it's not working

If you changed your archive policy to be more restrictive because you've run out of disk space, you may notice that events haven't started being archived according to your new policy. This is most likely because you must free up some space so the process has room to run. Stop Splunk, clear out ~5GB of disk space, and then start Splunk again (refer to Start Splunk in this manual for details on stopping and starting Splunk). After a while (exactly how long depends on how much data there is to process) you should see INFO entries about `BucketMover` in `splunkd.log` showing that buckets are being archived.

Archive data

If you want to archive your frozen data instead of deleting it, you must create an archiving script, as described in "Archive indexed data". You can later restore the archived data, as described in "Restore archived data".

Archive indexed data

Archive indexed data

Set up Splunk to archive your data automatically as it ages. To do this, configure indexes.conf to call archiving scripts located in `$SPLUNK_HOME/bin`. Edit a copy of `indexes.conf` in `$SPLUNK_HOME/etc/system/local/`, or in your own custom application directory in `$SPLUNK_HOME/etc/apps/`. Do not edit the copy in `$SPLUNK_HOME/etc/system/default`. For information on configuration files and directory locations, see "About configuration files".

Caution: By default, Splunk deletes all frozen data. To avoid losing your data, you **must** specify a valid `coldToFrozenScript` in `indexes.conf`.

For detailed information on data storage in Splunk, see "How Splunk stores indexes".

Sign your archives

Splunk supports archive signing; configuring this allows you to verify integrity when you restore an archive.

Use Splunk's index aging policy to archive

Splunk rotates old data out of the index based on your data retirement policy. Data moves through several stages, which correspond to file directory locations. Data starts out in the **hot** database, located as subdirectories under `$SPLUNK_HOME/var/lib/splunk/defaultdb/db/`. Then, data moves through the **warm** database, also located as subdirectories under `$SPLUNK_HOME/var/lib/splunk/defaultdb/db`. Eventually, data is aged into the **cold** database `$SPLUNK_HOME/var/lib/splunk/defaultdb/colddb`.

Finally, data reaches the **frozen** state. Splunk erases frozen index data once it is older than `frozenTimePeriodinSecs` in `indexes.conf`. The `coldToFrozenScript` (also specified in `indexes.conf`) runs just before the frozen data is erased. The default script simply writes the name of the directory being erased to the log file `$SPLUNK_HOME/var/log/splunk/splunkd_stdout.log`. If you want to archive frozen data

rather than delete it, you'll need to substitute your own archiving script.

To substitute your own script, add the following stanza to `$SPLUNK_HOME/etc/system/local/indexes.conf`:

```
[<index>]
coldToFrozenScript = <script>
```

Note the following:

- `<index>` specifies which index to archive.
- `<script>` specifies the archiving script.
    - Define the `<$script>` path relative to `$SPLUNK_HOME/bin`. The script needs to be located in that directory or a subdirectory.

Splunk ships with two archiving scripts in the `$SPLUNK_HOME/bin` directory. You can modify these (or you can create your own):

- `compressedExport.sh`: Export with tsidx files compressed as gz.
- `flatfileExport.sh`: Export as a flat text file (not recommended for current performance and resource issues -- it can take a long time, and use a lot of ram, 2-3GB, while running).

Note: If using one of these scripts, modify it to specify the archive location for your installation. By default, the location is set to `opt/tmp/myarchive`. Also, rename the script or move it to another location to avoid having changes overwritten when you upgrade Splunk. These are ***example scripts*** and should not be applied to a production instance without editing to suit your environment and testing extensively.

Windows Users

Windows users use this notation:

```
coldToFrozenScript = <script> "$DIR"
```

Note: Enclose with a double quotes " if it contains a space.

For `<script>`, you can use one of these example scripts:

- `WindowsCompressedExport.bat`. Download the example script here.
- `WindowsFlatfileExport.bat` (not recommended for current performance and resource issues -- it can take a long time, and use a lot of ram, 2-3GB, while running). Download the example script here.

Note: Rename the script or move it to another location to avoid having changes overwritten when you upgrade Splunk. These are ***example scripts*** and should not be applied to a production instance without editing to suit your environment and testing extensively.

Examples

The following configuration will archive main index frozen buckets in d:\myarchive

Change the parameter for "dest_base" in WindowsFlatfileExport.bat or WindowsCompressedExport.bat.

set dest_base=d:\myarchive

In c:\Program Files\Splunk\etc\system\local\inputs.conf specify:

[main]

coldToFrozenScript = "C:\Program Files\Splunk\share\splunk\WindowsCompressedExport.bat" "$DIR"

Restart Splunk.

Restore archived indexed data

Restore archived indexed data

Restore archived data by moving the archive into the thawed directory, `$SPLUNK_HOME/var/lib/splunk/defaultdb/thaweddb`. You can restore an archive to a Splunk server regardless of operating system with some restrictions -- data generated on 64bit systems is not likely to work well on 32 bit systems, and data cannot be moved from PowerPC or Sparc systems to x86 or x8-64 systems or vice versa. Data in `thaweddb` is not subject to the server's index aging scheme (hot > warm> cold > frozen). You can put old archived data in thawed for as long as you need. When the data is no longer needed, simply delete it or move it out of thawed.

The details of how to restore archived data depends on how it was archived. You can restore archived data to any index or instance of Splunk, with the caveat that you do not introduce **bucket** ID conflicts to your index. Archived data does not need to be restored to its pre-archival location.

Here is an example of safely moving a previously saved archive bucket to thawed.

First:

```
# cp -r db_1181756465_1162600547_0  $SPLUNK_HOME/var/lib/splunk/defaultdb/thaweddb/temporary-db
```

Note: If the archived db was compressed, be sure to uncompress the contents in the temporary directory.

Next:

```
# cd $SPLUNK_HOME/var/lib/splunk/defaultdb/thaweddb/
# mv temporary-db_1181756465_1162600547_0  db_1181756465_1162600547_0
```

Note: This examples assumes that the bucket id '0' does not conflict with any other bucket in the main (defaultdb) index. If it does, you will have to move the bucket to a non-conflicting ID.

Now, refresh the manifests:

```
# cd $SPLUNK_HOME/bin
# ./splunk login
# ./splunk _internal call /data/indexes/main/rebuild-metadata-and-manifests
```

Wait a few moments, and the contents of your newly thawed buckets should be searchable again.

See "Archive indexed data" for information on how to archive data in the first place.

Configure data security

What you can secure with Splunk

What you can secure with Splunk

Splunk includes several options for securing your data. Authentication options allow you to secure your Splunk Server. Audit configurations enable data security, including cryptographic signing and event hashing.

Authentication

Authentication includes SSL and HTTPS, user-based access controls (known as **roles**) and LDAP.

SSL/HTTPS

You can configure SSL for both Splunk's back-end (**splunkd** talking to the browser) and the front-end (HTTPS when logging into Splunk Web). To set up SSL for Splunk's back-end, see instructions here. To enable HTTPS for Splunk Web, follow these instructions.

Configure roles

You no longer have to use Splunk's default roles of **Admin**, **Power** or **User**. While these roles remain built into Splunk, you can now define your own roles out of a list of **capabilities**. Create flexible roles for Splunk users either in **Manager** or by editing authorize.conf.

Learn more about configuring roles (in the "Add and manage users" section of this manual).

LDAP

Splunk supports authentication via its internal authentication services or your existing LDAP server.

Learn more about configuring LDAP (in the "Add and manage users" section of this manual).

Scripted authentication

Use scripted authentication to tie Splunk's authentication into an external authentication system, such as RADIUS or PAM.

Learn more about scripted authentication (in the "Add and manage users" section of this manual).

Audit

Splunk includes audit features to allow you to track the reliability of your data. Watch files and directories with the **file system change monitor**, monitor activities within Splunk (such as searches or configuration changes) with **audit events**, cryptographically sign audit events events with **audit event signing**, and block sign any data entering your Splunk index with **IT data signing**.

File system change monitor

You can use the file system change monitor in Splunk Preview to watch any directory or file. Splunk indexes an event any time the file system undergoes any sort of change or someone edits the watched files. The file system change monitor's behavior is completely configurable through `inputs.conf`.

Learn more about how to configure the file system change monitor.

Audit events

Watch your Splunk instance by monitoring audit events. Audit events are generated whenever anyone accesses any of your Splunk instances -- including any searches, configuration changes or administrative activities. Each audit event contains information that shows you what changed where and when and who implemented the change. Audit events are especially useful in distributed Splunk configurations for detecting configuration and access control changes across many Splunk Servers.

Learn more about how audit events work.

Audit event signing

If you are using Splunk with an Enterprise license, you can configure audit events to be cryptographically signed. Audit event signing adds a sequential number (for detecting gaps in data to reveal tampering), and appends an encrypted hash signature to each audit event.

Configure auditing by setting stanzas in audit.conf, and inputs.conf.

Learn more about audit event signing.

IT data signing

If you are using Splunk with an Enterprise license, you can configure Splunk to verify the integrity of IT data as it is indexed. If IT data signing is enabled, Splunk creates a signature for blocks of data as it is indexed. Signatures allow you to detect gaps in data or tampered data.

Learn more about IT data signing.

Secure access to Splunk with HTTPS

Secure access to Splunk with HTTPS

You can enable HTTPS via Splunk Web or web.conf. You can also enable SSL through separate configurations. Splunk can listen on HTTPS or HTTP, but not both.

Important: If you are using Firefox 3, enabling SSL for a Splunk deployment may result in an "invalid security exception" being displayed in the browser. Refer to this workaround documentation for more information.

Enable HTTPS using Splunk Web

To enable HTTPS in Splunk Web, navigate to **Manager > System settings > General Settings** and select the **Yes** radio button underneath the **Enable SSL (HTTPS) in Splunk Web** setting.

Note: You must restart Splunk to enable the new settings. Also, you must now append "https://" to the URL you use to access Splunk Web.

Enable HTTPS by editing web.conf

In order to enable HTTPS, modify `web.conf`. Edit this file in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For more information on configuration files in general, see how configuration files work.

```
[settings]
httpport = <port number>
enableSplunkWebSSL = true
```

- `httpport`
  - Set the port number to your HTTPS port.
- `enableSplunkWebSSL`
  - Set this key to true to enable SSL for Splunk Web.

Once you have made the changes to web.conf, you must restart Splunk for the changes to take effect.

Change HTTPS certificates by editing web.conf

The certificates used for SSL between Splunk Web and the client browser are located in `$SPLUNK_HOME/share/splunk/certs/`.

Important: Splunk STRONGLY recommends that you DO NOT use the default Splunk Web certificate. Use of the default Splunk Web certificate will not result in confidential data transmission.

The certificates to use for Splunk Web HTTPS are specified in web.conf under the `[settings]` stanza.

```
[settings]
...
privKeyPath = /certs/privkey.pem
caCertPath = /certs/cert.pem
```

Restart Splunk Web from the CLI for your changes to take effect. To use Splunk's CLI, navigate to the `$SPLUNK_HOME/bin/` directory and use the `./splunk` command.

```
./splunk restart splunkweb
```

Note: After changing the Splunk Web certificate, users may receive certificate warnings or be prompted by their browser to accept the new certificate.

Examples

Use a third-party certificate for Splunk Web

- Follow the instructions to generate a new certificate signing request (CSR) and receive a new certificate from your organization's root certificate authority here.
    - Use the following openssl command on *nix:
        - `openssl req -new -key $SPLUNK_HOME/share/splunk/certs/privkey.pem -out $SPLUNK_HOME/share/splunk/certs/newcert.csr`
    - Use the following openssl command on Windows:
        - `openssl.exe req -new -key $SPLUNK_HOME\share\splunk\certs\privkey.pem -out $SPLUNK_HOME\share\splunk\certs\newcert.csr`
    - Optionally, you can generate or supply an alternative private key
- Copy the certificate received from your CA into `$SPLUNK_HOME/share/splunk/certs/newcert.pem`.
- Create or modify the following entry under the `[settings]` stanza in `$SPLUNK_HOME/etc/system/local/web.conf`
    - `caCertPath=/certs/newcert.pem`
- Restart Splunk Web
    - `$SPLUNK_HOME/bin/splunk restart splunkweb`

Use genWebCert.py to generate a new private key and create a new self-signed Splunk Web certificate

- Follow the instructions for creating a new root certificate here
- Issue the following commands:

```
$ cd $SPLUNK_HOME
$ mv ./share/splunk/certs/privkey.pem ./share/splunk/certs/privkey.pem.old
$ mv ./share/splunk/certs/cert.pem ./share/splunk/certs/cert.pem.old
$ ./bin/splunk cmd python ./bin/genWebCert.py

Generating a 1024 bit RSA private key
.....++++++
..............++++++
writing new private key to 'privkeySecure.pem'
-----
Signature ok
subject=/CN=example.splunk.com/O=SplunkUser
Getting CA Private Key
writing RSA key

$ ./bin/splunk restart splunkweb
```

- You can also use the above procedure to generate a new self-signed certificate if your self-signed certificate has expired.

Secure access to your Splunk server with SSL

Secure access to your Splunk server with SSL

Overview

The Splunk management port (default 8089) supports both SSL and plain text connections. SSL is turned on by default for communications among Splunk servers. Distributed search will often perform better with SSL enabled because of its built-in data compression.

To make changes to SSL settings, edit server.conf.

Important: If you are using Firefox 3, enabling SSL for a Splunk deployment may result in an "invalid security exception" being displayed in the browser. Refer to this workaround documentation for more information.

Note: This only enables SSL for Splunk's back-end communication. To turn on SSL for the browser, see "Secure access to Splunk with HTTPS".

Working with SSL settings

When the Splunk server is turned on for the first time, the server generates a certificate for that instance. This certificate is stored in the `$SPLUNK_HOME/etc/auth/` directory by default.

Change SSL settings by editing `$SPLUNK_HOME/etc/system/local/server.conf`. Edit this file in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For more information on configuration files in general, see how configuration files work.

```
[sslConfig]
enableSplunkdSSL = true
sslKeysfile = server.pem
sslKeysfilePassword = password
caCertFile = cacert.pem
caPath = $SPLUNK_HOME/etc/auth
certCreateScript = $SPLUNK_HOME/bin/genSignedServerCert.py
```

- `enableSplunkdSSL` = Setting this boolean key to `true` enables SSL in Splunk.
- `keyfile` = Certificate for this Splunk instance (created on Splunk start-up by default - if the `certCreateScript` tag is present).

Note: The path to the keyfile is relative to the `caPath` setting. If your keyfile is kept outside `$SPLUNK_HOME`, you must specify a full (absolute) path outside of `$SPLUNK_HOME` to reach it.

- `keyfilePassword` = Password for the pem file store, is set to `password` by default.
- `caCertFile` = This is the name of the certificate authority file.
- `caPath` = Path where the Splunk certificates are stored. Default is `$SPLUNK_HOME/etc/auth`.
- `certCreateScript` = Script for creating & signing server certificates.

With the default script enabled, on startup, Splunk will generate a certificate in the `caPath` directory.

Deactivate SSL

To deactivate SSL, simply set `enableSplunkdSSL` to FALSE. This will disable SSL.

Note: Running splunkd without SSL is not generally recommended. Distributed search will often perform better with SSL enabled.

Disable SSLv2

To disable SSLv2 and tell the HTTP server to only accept connections from SSLv3 clients, include the `supportSSLV3Only` attribute and set it to TRUE. By default, this setting is FALSE.

Generate a new root certificate

By default, all Splunk servers use the same root certificate. This allows Splunk instances to connect to each other out of the box.

Important: The default Splunk root certificate (which can be found in `$SPLUNK_HOME/etc/auth/ca.pem`) uses a private key that every other user of Splunk in the world has access to. Possession of a certificate authority?s private key will allow attackers to generate certificates that are signed by the trusted authority, which would defeat attempts to control authentication via PKI. This is **only** important if you wish to use SSL authentication functionality.

The script `$SPLUNK_HOME/bin/genRootCA.sh` (`%SPLUNK_HOME%\bin\genRootCA.bat` on Windows) allows you to create a root certificate to be used in creating subsequent server and web certificates. Run this script when you want to regenerate the certificates Splunk uses. It generates `cacerts.pem` (public key) and `ca.pem` (public/private password protected PEM). When you run it, it checks to see if certs are already in place, and if they are, prompts you to overwrite them. It then wraps these files into an X509-formatted cert. Distribute `cacerts.pem` to clients as desired and keep `ca.pem` in a secure location.

genRootCA.sh Example for the *nix platforms

The following example generates a new root certificate and private key pair at `$SPLUNK_HOME/etc/auth/ca.pem`.

Note: if Splunk is installed anywhere but `/opt/splunk`, you will need to set the environment variable OPENSSL_CONF to the path to your Splunk installation's openssl.cnf.

```
$ export OPENSSL_CONF=$SPLUNK_HOME/openssl/openssl.cnf
$ cd $SPLUNK_HOME
$ ./bin/genRootCA.sh  -d ./splunk/etc/auth/
There is ca.pem in this directory. If you choose to replace the CA then splunk servers will rec
new certs signed by this CA before they can interact with it.
Do you wish to replace the CA ? [y/N]
y
rm: cacert.pem: No such file or directory
This script will create a root CA
It will output two files. ca.pem cacert.pem
Distribute the cacert.pem to all clients you wish to connect to you.
Keep ca.pem for safe keeping for signing other clients certs
Remember your password for the ca.pem you will need to later to sign other client certs
Your root CA will expire in 10 years
```

```
Generating a 1024 bit RSA private key
..++++++
...........++++++
writing new private key to 'cakey.pem'
-----
Signature ok
subject=/C=US/ST=CA/L=SanFrancisco/O=SplunkInc/CN=SplunkCA/O=SplunkUser
Getting Private key
subject= /C=US/ST=CA/L=SanFrancisco/O=SplunkInc/CN=SplunkCA/O=SplunkUser
issuer= /C=US/ST=CA/L=SanFrancisco/O=SplunkInc/CN=SplunkCA/O=SplunkUser
notBefore=Apr 22 16:40:09 2010 GMT
notAfter=Apr 19 16:40:09 2020 GMT
```

genRootCA.bat Example for the Windows platform

The following example generates a new root certificate and private key pair at `%SPLUNK_HOME%\etc\auth`. Make sure that the `OPENSSL_CONF` environment variable points to the Splunk installation's `openssl.cnf`. Also note that path followed by the `-d` option, which specifies the destination directory for the generated key pair, is a DOS-style path and does not contain spaces.

```
>cd "c:\Program Files\Splunk\bin"
>set OPENSSL_CONF=c:\Program Files\Splunk\openssl.cnf
>splunk.exe cmd cmd.exe /c genRootCA.bat -d c:\progra~1\Splunk\etc\auth
C:\Program Files\Splunk\bin>splunk.exe cmd cmd.exe /c genRootCA.bat -d c:\progra~1\Splunk\etc\a
There is ca.pem in this directory. If you choose to replace the CA then splunk
servers will require new certs signed by this CA before they can interact with it.
Do you wish to replace the CA ? [y/N]
y
Deleting certs cacert.pem and ca.pem
del /f /q cacert.pem
del /f /q ca.pem
This script will create a root CA.
It will output two files: ca.pem cacert.pem.
Distribute the cacert.pem to all clients you wish to connect to you.
Keep ca.pem for safe keeping for signing other clients certs.
Remember your password for the ca.pem you will need to later to sign other client certs.
Your root CA will expire in 10 years.
"C:\Program Files\Splunk\bin\openssl.exe" req -newkey rsa:1024 -passout pass:password -subj /co
localityName=SanFrancisco/organizationName=SplunkInc/commonName=SplunkCA/organizationName=Splun

Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
.....++++++
.......++++++
writing new private key to 'cakey.pem'
-----
"C:\Program Files\Splunk\bin\openssl.exe" x509 -req -in careq.pem -passin pass:password -sha1 -
-out cacert.pem -days 3650
Loading 'screen' into random state - done
Signature ok
subject=/C=US/ST=CA/L=SanFrancisco/O=SplunkInc/CN=SplunkCA/O=SplunkUser
Getting Private key
Create root cert ca.pem from cacert.pem and cakey.pem

cacert.pem

cakey.pem
```

```
"C:\Program Files\Splunk\bin\openssl.exe" x509 -subject -issuer -dates -noout -in ca.pem
subject= /C=US/ST=CA/L=SanFrancisco/O=SplunkInc/CN=SplunkCA/O=SplunkUser
issuer= /C=US/ST=CA/L=SanFrancisco/O=SplunkInc/CN=SplunkCA/O=SplunkUser
notBefore=Jul 15 21:54:14 2010 GMT
notAfter=Jul 12 21:54:14 2020 GMT
```

Generate a new signed certificate and private key pair

By default, all Splunk servers use a certificate signed by the common root certificate discussed above. This allows Splunk instances to connect to each other out of the box.

Important: Splunk STRONGLY recommends that you DO NOT use the default self-signed certificate. Use of these default certificate will not result in confidential transmission of data.

`$SPLUNK_HOME/bin/genSignedServerCert.sh` allows you to create a new private key and server certificate using the current Splunk root certificate.

`genSignedServerCert.sh`	This shell script is a wrapper for the Python script that Splunk runs to generate certificates when you start it for the first time. This script creates a CSR (certificate signing request), self-signs it, and outputs a signed private key and certificate pair.

genSignedServer.sh Example

The following example will generate a new private key and new server certificate for the server example.splunk.com which is signed against the local Splunk root certificate.

```
$ cd $SPLUNK_HOME
$ /bin/genSignedServerCert.sh -d ./etc/auth/ -n server2 -c example.splunk.com -p

* Create certificate server2.pem signed by the root CA
* Store the server2.pem key file locally with your client/server application
* Enter a secret pass phrase when requested
* The pass phrase is used to access server2.pem in your application
* Enter the application's host name as the Common Name when requested
* Enter the root CA pass phrase (Getting CA Private Key) to sign the key file
* The key file will expire after one year or sooner when the root CA expires
Generating a 1024 bit RSA private key
.........................++++++
.....................++++++
writing new private key to 'server2.pemkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:CA
Locality Name (eg, city) []:SanFrancisco
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Splunk Inc.
Organizational Unit Name (eg, section) []:Security
Common Name (eg, YOUR name) []:example.splunk.com
```

```
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Signature ok
subject=/C=US/ST=CA/L=SanFrancisco/O=Splunk Inc./OU=Security/CN=example.splunk.com
Getting CA Private Key
subject= /C=US/ST=CA/L=SanFrancisco/O=Splunk Inc./OU=Security/CN=example.splunk.com
issuer= /C=US/ST=CA/L=SanFrancisco/O=SplunkInc/CN=SplunkCA/O=SplunkUser
notBefore=Apr 22 17:20:31 2010 GMT
notAfter=Apr 21 17:20:31 2013 GMT
```

Generate a new signed certificate and private key pair on Windows

On Windows run genSignedServercert.py.

genSignedServer.py Example

```
C:\Program Files\Splunk\bin>splunk cmd python "c:\Program Files\splunk\bin\gensignedservercert.
* Create certificate server2.pem signed by the root CA
* Store the server2.pem key file locally with your client/server application
* Enter a secret pass phrase when requested
* The pass phrase is used to access server2.pem in your application
* Enter the application's host name as the Common Name when requested
* Enter the root CA pass phrase (Getting CA Private Key) to sign the key file
* The key file will expire after one year or sooner when the root CA expires
Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
.................++++++
.....................................................++++++
writing new private key to 'server2key.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
Verify failure
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:CA
Locality Name (eg, city) []:San Francisco
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Splunk, Inc.
Organizational Unit Name (eg, section) []:Splunk Customer Support
Common Name (eg, YOUR name) []:Splunk Support
Email Address []:support@splunk.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:<password>
An optional company name []:
```

```
Loading 'screen' into random state - done
Signature ok
subject=/C=US/ST=CA/L=San Francisco/O=Splunk, Inc./OU=support/CN=splunksupport/emailAd
dress=support@splunk.com
Getting CA Private Key
subject= /C=US/ST=CA/L=San Francisco/O=Splunk, Inc./OU=support/CN=splunksupport/emailA
ddress=support@splunk.com
issuer= /C=US/ST=CA/L=San Francisco/O=Splunk/CN=SplunkCommonCA/emailAddress=supp
ort@splunk.com
notBefore=Jun 14 19:28:27 2010 GMT
notAfter=Jun 13 19:28:27 2013 GMT
```

Generate a CSR (Certificate Signing Request)

If your organization requires that your Splunk deployment use a certificate signed by an external CA or you otherwise wish to use certificates signed by a root certificate other than the default Splunk authority, you can use the following procedure to generate the CSR to send to the CA:

```
openssl req -new -key [certificate name].pem -out [certificate name].csr
```

You are prompted for the following X.509 attributes of the certificate:

- Country Name: Use the two-letter code without punctuation for country, for example: US or GB.
- State or Province: Spell out the state completely; do not abbreviate the state or province name, for example: California
- Locality or City: The Locality is the city or town name, for example: Oakland. Do not abbreviate. For example: Los Angeles, not LA, Saint Louis, not St. Louis.
- Company: If your company or department contains an &, @, or any other non-alphanumeric symbol that requires you to use the shift key, you must spell out the symbol or omit it. For example, Fflanda & Rhallen Corporation would be Fflanda Rhallen Corporation or Fflanda and Rhallen Corporation.
- Organizational Unit: This field is optional; but you can specify it to help identify certificates registered to an organization. The Organizational Unit (OU) field is the name of the department or organization unit making the request. To skip the OU field, press **Enter**.
- Common Name: The Common Name is the Host + Domain Name, for example www.company.com or company.com. This must match the host name of the server where you intend to deploy the certificate exactly.

This creates a private key ([certificate name].key), which is stored locally on your server, and a CSR ([certificate name].csr), which contains the public key associated with the private key. You can then use this information to request a signed certificate from an external CA.

To copy and paste the information into your CA's enrollment form, open the `.csr` file in a text editor and save it as a .txt file.

Note: Do not use Microsoft Word; it can insert extra hidden characters that alter the contents of the CSR.

Generate a CSR (Certificate Signing Request) on Windows

This is very similar to the method described above, but it requires an extra step to set the ENV variable `OPENSSL_CONF` -

- Open up a Command Prompt window and navigate to `$SPLUNK_HOME\bin`
- Set the `OPENSSL_CONF` ENV variable - `C:\Program Files\Splunk\bin>set OPENSSL_CONF=C:\Program Files\Splunk\openssl.cnf`
- Verify the variable has been set correctly - `>echo %OPENSSL_CONF%`
- Run the command to generate the CSR - `>openssl.exe req -new -key "C:\Program Files\Splunk\etc\auth\server.pem" -out server.csr -passin pass:password`
- As above, you are then prompted for the following X.509 attributes of the certificate.

Distribute certificates to your search peers

Distribute certificates to your search peers

When you enable distributed search on a Splunk instance (and restarting), keys are generated in $SPLUNK_HOME/etc/auth/distServerKeys/

Distribute the files $SPLUNK_HOME/etc/auth/distServerKeys/trusted.pem and private.pem from one host to the others which will participate in distributed search.

Support for different keys from multiple Splunk instances

Any number of Splunk instances can have their own unique certificates stored on other instances for authentication. The instances can store keys in `$SPLUNK_HOME/etc/auth/distSearchKeys/<peer_name>/<trusted|private>.pem`

For example: if you have Splunk instances A and B and they both have different keys and want to search Splunk instance C, do the following:

- On peer C, create `$SPLUNK_HOME/etc/auth/distSearchKeys/A/` and `etc/auth/distSearchKeys/B/`.
- Then, copy A's keys to `$SPLUNK_HOME/etc/auth/distSearchKeys/A/` and B's keys to `$SPLUNK_HOME/etc/auth/distSearchKeys/B/<code/>`.
- Finally, restart C.

Configure archive signing

Configure archive signing

Use archive signing to sign your Splunk data as it is **archived** (moved from `colddb` to `frozen`). This lets you verify integrity when you restore an archive. Configure the size of the slice by setting your archiving policies.

How archive signing works

Data is archived from the `colddb` to `frozen` when either:

- the size of your index reaches a maximum that you specify.
- data in your index reaches a certain age.

Specify archiving policies to define how your data is archived.

Splunk ships with two standard scripts, but you may use your own. Data is archived from the `colddb` to `frozen` with a `coldToFrozen` script that you specify. The `coldToFrozen` script tells Splunk how to format your data (gz, raw, etc..), and where to archive it. Archive signing happens after the `coldToFrozen` script formats your data into its archive format, and then the data is moved to the archive location that you specified according to your archive policy.

An archive signature is a hash signature of all the data in the data slice.

To invoke archive signing, use the standalone signtool utility. Add `signtool -s <path_of_archive>` to the `coldToFrozen` script anywhere after the data formatting lines, but before the lines that copy your data to your archive. See the section below on configuring coldToFrozen scripts.

Verify archived data signatures

Splunk verifies archived data signatures automatically upon restoring. You can verify signatures manually by using `signtool -v <path_to_archive>`.

Configure coldToFrozen scripts

Configure any `coldToFrozen` script by adding a line for the **signtool** utility.

Note: If you use a standard Splunk archiving script, either rename the script or move it to another location (and specify that location in `indexes.conf`) to avoid having changes overwritten when you upgrade Splunk.

Standard Splunk archiving scripts

The two standard **archiving** scripts that are shipped with Splunk are shown below with archive signing.

Splunk's two archiving scripts are:

compressedExport.sh

This script exports files with the tsidx files compressed as gz.

```
#!/bin/sh
gzip $1/*.tsidx
signtool -s <path_to_archive> # replace this with the path to the archive you want signed
cp -r $1 /opt/tmp/myarchive  #replace this with your archive directory
```

flatfileExport.sh

This script exports each splunk 'source' event stream as a flat text file.

```
#!/bin/sh
exporttool $1 ${1}/index.export
rm -rf ${1}/*.data
rm -rf ${1}/rawdata
rm -rf ${1}/*.tsidx
signtool -s <path_to_archive> # replace this with the path to the archive you want signed
cp -r $1 /opt/tmp/myarchive  #replace this with your archive directory
```

Note: flatfileExport.sh is currently not recommended for performance and resource issues we hope to address in the future. It can take a long time (tens of minutes to hours), and use a lot of ram, 2-3GB, while running.

Your own custom scripts

You can also use your own scripts to move data from cold to frozen.

Sign or verify your data slices

Use `signtool`, located in `$SPLUNK_HOME/bin`, to sign data slices as they are archived or verify the integrity of an archive.

Syntax

To sign:

signtool [- s | -- sign] *archive_path*

To verify:

signtool [-v | --verify] *archive_path*

Configure IT data block signing

Configure IT data block signing

IT data signing helps you certify the integrity of your IT data. If you enable IT data signing and index some data, Splunk tells you if that data is ever subsequently tampered with at the source. For example, if you have enabled IT data signing and index a log file in Splunk, Splunk will show you if anyone removes or edits some entries from that log file on the original host. You can thus use Splunk to confirm that your data has been tampered with.

Note: Signing IT data is different than signing Splunk audit events. IT data signing refers to signing external IT data while it is indexed by Splunk; audit events are events that Splunk's auditing feature generates and stores in the audit index.

How IT data signatures work

Splunk takes external IT data (typically in the form of log files), and applies digital signatures and signature verification to show whether indexed or archived data has been modified since the index was initially created.

A signature for a block of IT data involves three things:

- A hash is generated for each individual event.
- The events are grouped into blocks of a size you specify.
- A digital signature is generated and applied to each block of events.

Note: Splunk can encrypt the hash to create a digital signature if you have configured the public and private keys in `audit.conf`. See "Configure audit event signing" for details.

This digital signature is stored in a database you specify and can be validated as needed. Splunk can demonstrate data tampering or gaps in the data by validating the digital signature at a later date. If the signature does not match the data, an unexpected change has been made.

Configure IT data signing

This section explains how to enable and configure IT data signing. You enable and configure IT data signing for each index individually, and then specify one central database for all the signing data.

You configure IT data signing in indexes.conf. Edit this file in `$SPLUNK_HOME/etc/system/local/` or in your custom application directory, in `$SPLUNK_HOME/etc/apps/`. Do not edit the copy in `default`. For more information on configuration files in general, see "About configuration files".

You can:

- Enable IT data signing and specify the number of events contained in your IT data signatures.
- Disable IT data signing.
- Specify the database to store signing data in.

Note: You must configure audit event signing by editing `audit.conf` to have Splunk encrypt the hash signature of the entire data block.

Enable IT data signing and specify the number of events in an IT data signature

By default, IT data signing is disabled for all indexes.

To enable IT data signing, set the `blockSignSize` attribute to an integer value greater than 0. This attribute specifies the number of events that make up a block of data to apply a signature to. You must set this attribute for each index using IT data signing.

This example enables IT data signing for for the `main` index and sets the number of events per signature block to 100:

```
[main]
```

```
blockSignSize=100
...
```

Note: the maximum number of events for the `blockSignSize` attribute is 2000.

You now must reindex your data for this change to take effect:

```
./splunk stop
./splunk clean all
.splunk start
```

Disable IT data signing

To disable IT data signing, set the `blockSignSize` attribute to 0 (the default). This example disables IT data signing off for the `main` index:

```
[main]
blockSignSize=0
...
```

Specify the signature database

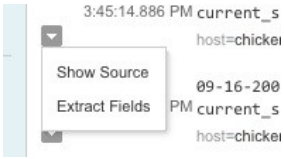
The IT data signature information for each index with IT data signing enabled is stored in the signature database. Set the value of the `blockSignatureDatabase` attribute to the name of the database where Splunk should store IT signature data. This is a global setting that applies to all indexes:

```
blockSignatureDatabase=<database_name>
```

The default database name is `_blocksignature`.

View the integrity of IT data

To view the integrity of indexed data at search time, open the **Show source** window for results of a search. To bring up the **Show source** window, click the drop-down arrow at the left of any search result. Select **Show source** and a window will open displaying the raw data for each search result.



The **Show source** window displays information as to whether the block of IT data has gaps, has been tampered with, or is valid (no gaps or tampering).

The status shown for types of events are:

- Valid
- Tampered with
- Has gaps in data

Performance implications

Because of the additional processing overhead, indexing with IT data signing enabled can negatively affect indexing performance. Smaller blocks mean more blocks to sign and larger blocks require more work on display. Experiment with block size to determine optimal performance, as small events can effectively use slightly larger blocks. The block size setting is a maximum, you may have smaller blocks if you are not indexing enough events to fill a block in a few seconds. This allows incoming events to be signed even when the indexing rate is very slow.

- Turning IT data signing ON slows indexing.
- Setting the `blockSignSize` attribute to high integer values (such as 1000) slows indexing performance.
- For best performance, set `blockSignSize` to a value near 100.

Cryptographically sign audit events

Cryptographically sign audit events

Splunk creates audit trail information (by creating and signing **audit events**) when you have auditing enabled. Audit event signing is only available if you are running Splunk with an Enterprise license.

How audit event signing works

The audit processor signs audit events by applying a sequence number ID to the event, and by creating a hash signature from the sequence ID and the event's timestamp. Once you've enabled audit signing, you can search for gaps in the sequence of these numbers and find out if your data has been tampered with.

Hash encryption

For each processed audit event, Splunk's auditing processor computes an SHA256 hash on all of the data. The processor then encrypts the hash value and applies Base64 encoding to it. Splunk then compares this value to whatever key (your private key, or the default keys) you specify in audit.conf.

Configure audit event signing

Configure the following settings of Splunk's auditing feature through audit.conf:

- Turn on and off audit event signing.
- Set default public and private keys.

Configure audit.conf

Create your own `audit.conf`. Edit this file in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For more information on configuration files in general, see how configuration files work.

Generate your own keys using genAuditKeys.py in `$SPLUNK_HOME/bin/`:

```
# ./splunk cmd python genAuditKeys.py
```

This creates your private and public keys, `$SPLUNK_HOME/etc/auth/audit/private.pem` and `$SPLUNK_HOME/etc/auth/audit/public.pem`. To use these keys, set `privateKey` and `publicKey` to the path to your keys in your `$SPLUNK_HOME/etc/system/local/audit.conf`:

```
[auditTrail]
privateKey = $PATH_TO_PRIVATE_KEY
publicKey = $PATH_TO_PUBLIC_KEY
```

Note: If the `[auditTrail]` stanza is missing, audit events are still generated, but not signed. If the `publicKey` or `privateKey` values are missing, audit events will be generated but not signed.

Search to detect gaps in your data

Note: The functionality described in this section is not yet available as of version 4.0.4 , and will be delivered in an upcoming maintenance release.

Once you've configured audit event signing, the sequence number ID that the audit processor assigns to each event lets you detect gaps in data which can identify tampering with the system. You can search the audit events to determine if gaps are detected:

```
index=_audit | audit
```

The field that contains the status of the event is called "validity". Values can be:

- VALIDATED - no gap before this event and event signature matches
- TAMPERED - event signature does not match
- NO SIGNATURE - the signature was not found
- NO PUBLIC KEY - cannot validate

The field that contains the gap status is called "gap". Values can be:

- TRUE - a gap was found
- FALSE - no gap was found
- N/A - no id was found

Audit Splunk activity

Audit Splunk activity

With auditing enabled, Splunk logs distinct events to the audit index (`index=_audit`). Every interaction with Splunk -- search, configuration changes, etc -- generates an audit event. Directories monitored by file change monitoring create audit events as well. This topic outlines the composition and generation of audit events.

Note: The punct field is not available for events in the _audit index.

What's in an audit event?

- Timestamp:
  - ♦ date and time of the event.
- User information:

    - the user who generated the event.
    - If the event contains no user information, Splunk sets the user to whoever is currently logged in.
- Additional information:
    - available event details -- what file, success/denial, etc.
- ID (only if audit event signing is turned on):
    - a sequential number assigned to the event for detecting gaps in data.
- Hash signature:
    - PKI encrypted SHA256 hash signature, including the timestamp and ID.
- Additional attribute/value pairs specific to the type of event.

Example

The following is a sample signed audit log entry:

```
11-01-2007 09:23:59.581 INFO  AuditLogger - Audit:[timestamp=Thu Nov  1 09:23:59 2007, id=1,
user=admin, action=splunkStarting, info=n/a][NSsJkuZZNn1dKaH3tjgxN/RbGeKaQ/dXArIdK2M97E0
Ckv6xqMurYbUVqC6YoICLjW/H113u6FDTPMBGdk29J95X1SecazMf+H1tRqfc+vcJPZH1RcQaiVCcJwRTJuXD4Z5
JidyvjVIECIdrhPSAGj7CSEhTdYx4tOEfl5yMckU=]
```

The information within the first set of brackets ([ ]) is the hashed and signed data. The string in the second set of brackets is the hash signature.

What activity generates an audit event?

Audit events are generated from monitoring:

- all files in Splunk's configuration directory `$SPLUNK_HOME/etc/*`
    - files are monitored for add/change/delete using the file system change monitor.
- system start and stop.
- users logging in and out.
- adding / removing a new user.
- changing a user's information (password, role, etc).
- execution of any capability in the system.
    - capabilities are listed in authorize.conf

Audit event storage

Splunk stores audit events locally in the audit index (`index=_audit`). Audit events are logged in the log file: `$SPLUNK_HOME/var/log/splunk/audit.log`.

If you have configured Splunk as a forwarder in a distributed setting, audit events are forwarded like any other event. Signing can happen on the forwarder, or on the receiving Splunk instance.

Audit event processing

The file audit.conf tells the audit processor whether or not to encrypt audit events. As audit events are generated, Splunk's auditing processor assigns a sequence number to the event and stores the event information in a SQLite database. If there is no user information specified when the event is generated, Splunk uses the currently signed in user information. Finally, if audit event signing is set, Splunk hashes and encrypts the event.

Search for audit events

Search audit events in Splunk Web or in Splunk's CLI. To do this, pipe your searches to the audit command. The audit search command is most useful if audit event signing *has* been configured. However, if you want to search for all audit events where audit event signing *has not* been configured (or to skip integrity validation) you may search the whole audit index.

- To search for all audit events, specify the `_audit` index:

```
index=_audit
```

This search returns all audit events.

- Pipe your search to the audit command:

```
index=_audit | audit
```

This search returns the entire audit index, and processes the audit events it finds through the `audit` command.

Narrow your search before piping to the audit command. However, you can only narrow the time range, or constrain by a single host. This is because each host has its own ID number sequence. Since sequential IDs exist to enable detection of gaps in audit events, narrowing a search across multiple hosts causes false gap detection.

The field that contains the status of the event is called "validity". Values can be:

- VALIDATED - no gap before this event and event signature matches
- TAMPERED - event signature does not match
- NO SIGNATURE - the signature was not found

The field that contains the gap status is called "gap". Values can be:

- TRUE - a gap was found
- FALSE - no gap was found
- N/A - no id was found

Configure event hashing

Configure event hashing

Event hashing provides a lightweight way to detect if events have been tampered with between index time and search time.

Event hashes aren't cryptographically secure. Someone could tamper with an event if they have physical access to a machine's file system.

You should use event hashing only if you don't have the capability to run Splunk's IT data block signing feature; individual event hashing is more resource intensive than data block signing.

How event hashing works

When event hashing is enabled, Splunk hashes events with a SHA256 hash just before index time. When each event is displayed at search time, a hash is calculated and compared to that event's index time hash. If the hashes match, the event is decorated in the search results as "valid". If the hashes don't match, the event is decorated as "tampered" (For the CLI: the value of the decoration is stored in the **internal field**: `_decoration`).

Configure event hashing by editing `$SPLUNK_HOME/etc/system/local/audit.conf`. Set up event hashing filters that **whitelist** or **blacklist** events based on their host, source, or sourcetype.

- A whitelist is a set of criteria that events must match to be hashed. If events don't match, they aren't hashed.
- A blacklist is a set of criteria that events must match to NOT be hashed. If events don't match, then they are hashed.

See more on configuring event hashing below.

Configure event hashing

Turn on event hashing by adding an `[eventHashing]` stanza to `audit.conf`. If you want to add filters to event hashing, list each filter for which you have a `filterSpec` stanza in a comma-separated list in the `filters =` key.

Configure filtering

Set up filters for event hashing in `audit.conf`. Create a stanza after the `[eventHashing]` stanza to define a filter. Specify the details of each filter using comma-separated lists of hosts, sources, and sourcetypes.

```
[filterSpec:FilterType:NameOfFilter]
host=<comma separated list of hosts>
source=<comma separated list of sources>
sourcetype=<comma separated list of sourcetypes>
```

Next, turn on specific filters by adding a `filter=` key under the `[eventHashing]` stanza with a list of the names of the filters you want enabled.

```
[eventHashng]
filters=filter1,filter2,...
```

Note: The filter list is an OR list that is evaluated left to right. Currently, there is no support for an AND list of filters.

Event hashing filter precedence

- Filters are evaluated from left to right.
- Whitelist filters are evaluated before blacklist filters.
- If an event doesn't match a filter and no more filters exist, then it will be hashed.

Configure a whitelist filter

Create a **whitelist** filter by changing the filter type in the `filterSpec` stanza to `event_whitelist`.

```
[filterSpec:event_whitelist:<specname>]
host=<comma separated list of hosts>
source=<comma separated list of sources>
sourcetype=<comma separated list of sourcetypes>
```

Configure a blacklist filter

Create a **blacklist** filter by changing the filter type in the `filterSpec` stanza to `event_blacklist`.

```
[filterSpec:event_blacklist:<specname>]
host=<comma separated list of hosts>
source=<comma separated list of sources>
sourcetype=<comma separated list of sourcetypes>
```

Example filter configurations

Turn on hashing for all events:

```
[eventHashing]
```

(Yes, just one line.)

Simple blacklisting

Doesn't hash events from any of the listed hosts. Events that do not come from the listed hosts will be hashed.

```
[filterSpec:event_blacklist:myblacklist]
host=foo.bigcompany.com, 45.46.1.2, 45.46.1.3
[eventHashing]
filters=myblacklist
```

Multiple type blacklisting

Doesn' t hash any of the listed hosts, sources, or sourcetypes. Events from all other hosts, sources, or sourcetypes will be hashed.

```
[filterSpec:event_blacklist:myblacklist]
host=somehost.splunk.com, 46.45.32.1
source=/some/source
sourcetype=syslog, apache.error
[eventHashing]
filters=myblacklist
```

Simple whitelisting

Hashes only events that contain the specified sourcetype. Events from any other sourcetype won't be hashed. (Note the use of the "all" tag in the blacklist specification.)

```
[filterSpec:event_whitelist:allow_syslog]
sourcetype=syslog
```

```
[filterSpec:event_blacklist:denyall]
#"all" is a special tag that matches all events
all=True
[eventHashing]
filters=allow_syslog, denyall
```

Event hashing in search results

Splunk provides different visual indicators for your search results depending on the interface you use.

In Splunk Web

Search results are decorated in Splunk Web with decorations showing whether an event is valid or has been tampered with.

If an event is valid, you'll see this above the raw data:

If an event has been tampered with, you'll see this above the raw event data:

In the CLI

Search results in the CLI return the value for the event hash result in the `_decoration` field.

Manipulate and run reports on the `_decoration` field the same way as you'd do for any other field.

Example:

```
./splunk search " * | top _decoration"
```

The resulting output:

```
_decoration                count  percent
------------               -----  ---------
decoration_audit_valid     50     50.000000
decoration_audit_tampered  50     50.000000
```

Hardening standards

Hardening standards

Splunk recommends using the following standards to harden your Splunk instances. Following these standards will reduce Splunk's attack surface and mitigate the risk and impact of most vulnerabilities.

Service accounts

- Practice the principle of least privilege by running Splunk as an unprivileged user rather than a privileged account such as root or Administrator.

- On unix or linux, use the "splunk" user that is created via the PKG or RPM packages, or create your own user that only has privilege and ownership over $SPLUNK_HOME
- On Windows, the local system context is often the best choice. However, if you require communication to occur via a windows communication channel (e.g. WMI), use an account with restricted access.

Splunk components

- Disable all unnecessary Splunk components
  - For single-server Splunk deployments:
    - Splunk forwarders should not run Splunk Web and should not be configured to receive data on TCP or UDP ports or from other Splunk instances
  - For multi-server Splunk deployments:
    - Splunk search heads should not receive data on TCP or UDP ports or from other Splunk instances
    - Splunk indexers in a distributed search environment should not run Splunk Web if users are not logging in to them to search
    - Splunk forwarders should not run Splunk Web and should not be configured to receive data on TCP or UDP ports or from other Splunk instances

Network access

- Do not place Splunk on a network segment that is Internet-facing (i.e. Splunk should not be accessible directly via the Internet)
  - Remote users will still be able to access Splunk via a Virtual Private Network
- Use a host-based firewall to restrict access to Splunk's web, management, and data ports
  - End users and administrators will need to access Splunk Web (TCP port 8000 by default)
  - Search heads will need to access their search peers on the Splunk management port (TCP port 8089 by default)
  - Deployment clients will need to access the deployment server on the Splunk management port (TCP port 8089 by default)
  - Forwarders will need to access the Splunk index server's data port (TCP port 9997 by default)
  - Remote CLI calls use the Splunk management port. Consider restricting this port to local calls only via a host firewall.
  - In most cases, it is not recommended to allow access to Splunk forwarders on any port
- Install Splunk on an isolated network segment that only trustworthy machines can access
- Do not permit Splunk access to the Internet unless access to Splunkbase or inline documentation is a requirement

Operating System

- Splunk strongly recommends hardening the operating system of all Splunk servers
  - If your organization does not have internal hardening standards, Splunk recommends the CIS hardening benchmarks
  - At the very least, limit shell/command line access to your Splunk servers

Availability and reliability

- Configure redundant instances of Splunk, both indexing a copy of the same data
- Back up Splunk data and configurations on a regular basis
- Execute a periodic recovery test by attempting to restore Splunk from backup

Physical security

- Secure physical access to all Splunk servers
- Ensure that end users of Splunk practice sound physical and endpoint security
    - ♦ Set a short time-out for user sessions in Splunk Web via Manager

Confidentiality and integrity

- Use SSL encryption on Splunk's web, management, and data ports

Authentication

- Upon installing Splunk, change the default "admin" password the first time that you log in
- Do not use the default root certificate or server certificates that ship with Splunk
    - ♦ Either generate a unique root certificate and self-signed server certificates, use your enterprise root certificate authority, or use a third-party certificate authority
    - ♦ Make sure that you back up the private keys in a secure location
- Use SSL authentication between forwarders and indexers
- Use LDAP or other third-party systems to control authentication to Splunk
    - ♦ When using LDAP, make sure that your LDAP implementation enforces:
        - ◊ Strong password requirements for length and complexity
        - ◊ A low incorrect attempt threshold for password lockout

Authorization

- Protect access to Splunk features and data by using Splunk's role-based access control
    - ♦ Practice the principle of least privilege by only granting access to features and data based on business justification
    - ♦ Use an approval process to validate access requests to Splunk features and data

Auditing

- Perform a periodic review of Splunk's access and audit logs
- Perform a periodic review of the Splunk server's audit and security logs

Configuration management

- Use a configuration management tool such as subversion to provide version control for Splunk configurations
- Integrate Splunk configuration changes into your existing change management framework
- Configure Splunk to monitor its own configuration files and alert on changes

Client browser

- Use a current version of a supported browser such as Firefox or Internet Explorer
- Use a client-side JavaScript blocker such as noscript on Firefox or Internet Explorer 8 Filters to help protect against XSS, XSRF, and similar exploits
- Ensure that users have the latest version of Flash installed

Deploy to other Splunk instances

About deployment server

About deployment server

Splunk deployments range in size from departmental, single server installations, where one Splunk instance handles all tasks ranging from data input to indexing to search and reporting, to enterprise distributed deployments with data flowing between thousands of instances of Splunk forwarders, indexers, and search heads. Splunk provides valuable tools to handle distributed deployments of any size. Chief among them is the Splunk **deployment server** and its features.

The deployment server is Splunk's technology for pushing out configurations and content to distributed Splunk instances. A key use case for the deployment server is to manage configuration for groups of forwarders. For example, if you have several sets of **forwarders**, each set residing on a different machine type, you can use the deployment server to push out different content according to machine type. Similarly, in a **distributed search** environment, you can use a deployment server to push out content to sets of indexers. If you want an overview of the different ways you can design a Splunk deployment in your organization, check out the deployment information in the Community Wiki.

The first several topics in this section explain how to configure a deployment server and its clients. Topics then follow that show how to employ this technology for specific use cases.

The big picture (in words and diagram)

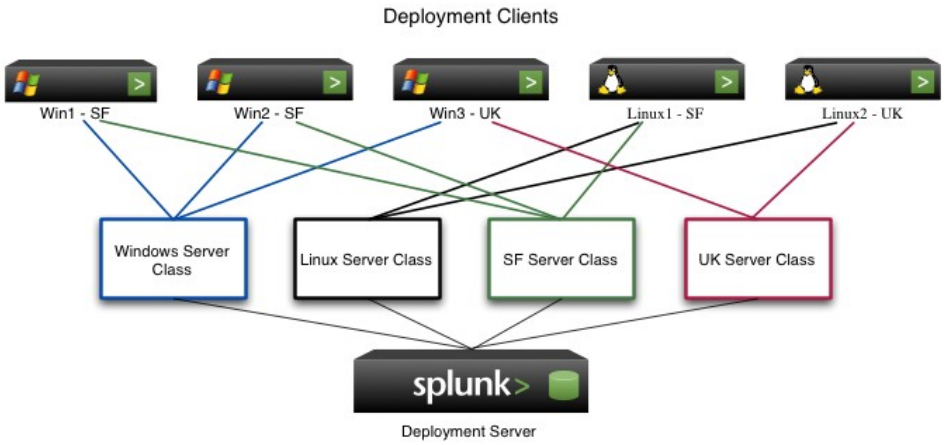
In a Splunk deployment, you use a deployment server to push out content and configurations (collectively called *deployment apps*) to **deployment clients**, grouped into **server classes**.

A *deployment server* is a Splunk instance that acts as a centralized configuration manager, collectively managing any number of Splunk instances, called "deployment clients". Any Splunk instance -- even one indexing data locally -- can act as a deployment server.

A *deployment client* is a Splunk instance remotely configured by a deployment server. A Splunk instance can be both a deployment server and client at the same time. Each Splunk deployment client belongs to one or more server classes.

A *server class* is a set of deployment clients, grouped by some set of configuration characteristics, so that they can be managed as a unit. You can group clients by application, OS, type of data, or any other feature of your Splunk deployment. To update the configuration for a set of clients, the deployment server pushes out configuration files to all or some members of a server class. Besides configuration settings, you can use the deployment server to push out any sort of content. Server classes are configured on the deployment server.

This diagram provides a conceptual overview of the relationship between a deployment server and its set of deployment clients and server classes:



In this example, each deployment client is a Splunk forwarder that belongs to two server classes, one for its OS and the other for its geographical location. The deployment server maintains the list of server classes and uses those server classes to determine what content to forward to each client. For an example of how to implement this type of arrangement to govern the flow of content to clients, see "Deploy several standard forwarders".

A *deployment app* is a set of deployment content (including configuration files) deployed as a unit to clients of a server class. A deployment app might consist of just a single configuration file, or it can consist of many files. Depending on filtering criteria, an app might get deployed to all clients in a server class or to a subset of clients. Over time, an app can be updated with new content and then redeployed to its designated clients. The deployment app can be an existing Splunk app, or one developed solely to group some content for deployment purposes.

Note: The term "app" has a somewhat different meaning in the context of the deployment server from its meaning in the general Splunk context. For more information on Splunk apps, see "What are apps and add-ons?".

For more information on deployment servers, server classes, and deployment apps, see "Define server classes". For more information on deployment clients, see "Configure deployment clients".

A **multi-tenant environment** means that you have more than one deployment server running on the same Splunk instance, and each deployment server is serving content to its own set of deployment clients. For information about multi-tenant environments, see "Deploy in multi-tenant environments".

Key terms

Here's a recap of the key definitions:

Term	Meaning
deployment server	A Splunk instance that acts as a centralized configuration manager. It pushes configuration updates to other Splunk instances.
deployment client	A remotely configured Splunk instance. It receives updates from the deployment server.
server class	A deployment configuration shared by a group of deployment clients. A deployment client can belong to multiple server classes.

deployment app	A unit of content deployed to one or more members of a server class.
multi-tenant environment	A deployment environment involving multiple deployment servers.

Communication between deployment server and clients

The deployment client periodically polls the deployment server, identifying itself. The deployment server then reviews the information in its configuration to find out if there is something new or updated to push out to that particular client. If there is new content to deploy to a given deployment client, the deployment server tells the client exactly what it should retrieve. The deployment client then retrieves the new content and treats it according to the instructions specified for the server class it belongs to--maybe it should restart, run a script, or just wait until someone tells it to do something else.

Plan a deployment

Plan a deployment

If you've got Splunk instances serving a variety of different populations within your organization, chances are their configurations vary depending on who uses them and for what purpose. You might have some number of Splunk instances serving the helpdesk team, configured with a specific app to accelerate troubleshooting of Windows desktop issues. You might have another group of Splunk instances in use by your operations staff, set up with a few different apps designed to emphasize tracking of network issues, security incidents, and email traffic management. A third group of Splunk instances might serve the Web hosting group within the operations team.

Rather than having to manage and maintain these divergent Splunk instances one at a time, you can put them into groups based on their use, identify the configurations and apps needed by each group, and then use the **deployment server** to update their various apps and configurations as needed.

In addition to grouping Splunk instances by use, there are other useful types of groupings you can specify. For example, you might group Splunk instances by OS or hardware type, by version, or by geographical location or timezone.

Configuration overview

For the great majority of deployment server configurations, perform these steps:

1. Designate one of your Splunk servers as the deployment server. A deployment server can also be a **deployment client**, either of itself or of a different deployment server.

2. Group the deployment clients into **server classes**. A server class defines the clients that belong to it and what content gets pushed out to them. Each deployment client can belong to multiple server classes.

3. Create a `serverclass.conf` file on the deployment server. It specifies the server classes and the location of the deployment apps. Refer to "Define server classes" in this manual for details.

4. Create the directories for your deployment apps, and put the content to be deployed into those directories. Refer to "Deploy apps and configurations" in this manual for details.

5. Create a `deploymentclient.conf` for each deployment client. It specifies what deployment server the client should communicate with, the specific location on that server from which it should pick up content, and where it should put it locally. Refer to "Configure deployment clients" in this manual for details.

6. For more complex deployments with multiple deployment servers, create a `tenants.conf` file on one of the deployment servers. This allows you to define multiple deployment servers on a single Splunk instance and redirect incoming client requests to a specific server according to rules you specify. Refer to "Deploy in multi-tenant environments" in this manual for more information about configuring `tenants.conf`. Most deployment server topologies don't require that you touch `tenants.conf`, however.

For an example of an end-to-end configuration, see "Deploy several standard forwarders".

Note: The deployment server and its deployment clients must agree in the SSL setting for their `splunkd` management ports. They must all have SSL enabled, or they must all have SSL disabled. To configure SSL on a Splunk instance, set the `enableSplunkdSSL` attribute in server.conf to "true" or "false".

Restart or reload?

The first time you configure the deployment server and its clients, you'll need to restart all instances of Splunk. When you restart the deployment server, it automatically deploys any new content to its clients. Later on, to deploy new or updated content without restarting, you can use the CLI `reload` command, as described in "Deploy apps and configurations" in this manual.

Define server classes

Define server classes

A **server class** defines a deployment configuration shared by a group of **deployment clients**. It defines both the criteria for being a member of the class and the set of content to deploy to members of the class. This content (encapsulated as "deployment apps") can consist of Splunk apps, system configurations, and other related content, such as scripts, images, and supporting material. You can define different server classes to reflect the different requirements, OSes, machine types, or functions of your deployment clients.

You define server classes in serverclass.conf on the **deployment server**. Create one in `$SPLUNK_HOME/etc/system/local`. For information about configuration files, including an explanation of their basic structure, see "About configuration files" in this manual.

If you have multiple server classes, you might want to define a "global" server class that applies to all deployment clients by default. You can then override various aspects of it as needed by defining more specific server classes. For example, if you have a mix of Windows and Linux forwarders sending data to the same indexer, you might want to specify that all forwarders get a common `outputs.conf` file, but that Windows forwarders get one `inputs.conf` file and Linux forwarders a different one. In that case, you could specify the `outputs.conf` in the global server class and then create separate Windows and Linux server classes for the different `inputs.conf` files.

In addition to defining attributes and content for specific server classes, you can also define attributes that pertain just to a single app within a server class.

Important: All configuration information is evaluated numerically and then alphabetically (0-9, then a-z), so nomenclature matters.

A deployment client has its own configuration, defined in deploymentclient.conf. The information in `deploymentclient.conf` tells the deployment client where to go to get the content that the server class it belongs to says it should have.

The next section provides a reference for the server class configuration settings. You might want to read it while referring to the set of simple example configurations presented later in this topic. In addition, there are several longer and more complete examples presented later in this manual, including "Deploy several standard forwarders".

What you can define for a server class

You can specify settings for a global server class, as well as for individual server classes or apps within server classes. There are three levels of **stanzas** to enable this:

Stanza	Meaning	Scope
`[global]`	The global server class.	Attributes defined here pertain to all server classes.
`[serverClass:<serverClassName>]`	Individual server class. A `serverClass` is a collection of apps.	Attributes defined here pertain to just the server class `<serverClassName>`.
`[serverClass:<serverClassName>:app:<appName>]`	App within server class.	Attributes defined here pertain to just the specified deployment app `<appName>` within the specified `<serverClassName>`. To indicate all apps within `<serverClassName>`, `<appName>` can be the wildcard character: *, in which case it will cause all content in the repositoryLocation to be added to this serverClass.

Attributes in more specific stanzas override less specific stanzas. Therefore, an attribute defined in a `[serverClass:<serverClassName>]` stanza will override the same attribute defined in `[global]`.

The attributes are definable for each stanza level, unless otherwise indicated. Here are the most common ones:

Attribute	What it's for	
repositoryLocation	The location on the deployment server where the content to be deployed for this server class is stored.	`$SPLUNK_HOME/etc/deployment-apps`
targetRepositoryLocation	The location on the deployment client where the content to be deployed for this server class should be installed. You can override this in `deploymentclient.conf` on the deployment client.	`$SPLUNK_HOME/etc/apps`
continueMatching	If set to false, the deployment server will look through the list of server classes in this configuration file and stop when it matches the first one to a client. If set to true, the deployment server will continue to look and match. This option is available because you can define multiple, layered sets of server classes. A `serverClass` can override this property and stop the matching.	true
endpoint	The HTTP location from which content can be downloaded by a deployment client. The deployment server fills in the variable substitutions itself, based on information received from the client. You can provide any URI here, as long as it uses the same variables. In most cases, this attribute does not need to be specified.	`$deploymentServerUri$/services/s`
filterType	Set to **"whitelist"** or **"blacklist"**. This determines the order of execution of filters. If `filterType` is `whitelist`, all whitelist filters are applied first, followed by blacklist filters. If `filterType` is `blacklist`, all blacklist filters are applied first, followed by whitelist filters.	whitelist

	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 You can override this value at the `serverClass` and `serverClass:app` levels. If you specify `whitelist` at the global level, and then specify `blacklist` for an individual server class, the setting becomes `blacklist` for

	that server class, and you have to provide another filter in that server class definition to replace the one you overrode.	

whitelist.<n>
 	
blacklist.<n>
	<n> is a number starting at 0, and incrementing by 1.
	
	Set the attribute to ipAddress, hostname, or clientName:
	
	• ipAddress is the IP address of the deployment client. Can use wildcards, such as 10.1.1.*
	• hostname is the host name of deployment client. Can use wildcards, such as *.splunk.com.
	• clientName is a logical or tag name that can be assigned to a deployment client in `deploymentclient.conf`. clientName takes precedence over ipAddress or hostname when matching a client to a filter.
	
	Here are some examples: When `filterType` is whitelist:
	
	`whitelist.0=*.fflanda.com`
	`blacklist.0=printer.fflanda.com`
	`blacklist.1=scanner.fflanda.com`
	
	This will cause all hosts in fflanda.com, except 'printer' and 'scanner' to match this server class.
	
	When `filterType` is blacklist:
	
	`blacklist.0=*`
	`whitelist.0=*.web.fflanda.com`
	`whitelist.1=*.linux.fflanda.com`
	
	This will cause only the 'web' and 'linux' hosts to match the server class. No other hosts will match.
		n/a

	You can override this value at the serverClass and serverClass:app levels. **Important:** Overriding one type of filter (whitelist/blacklist) causes the other to the overridden too. If, for example, you override the whitelist, the blacklist will not be inherited from the parent; you must provide one in the stanza.	
stateOnClient	Set to "enabled", "disabled", or "noop". This setting specifies whether the deployment client receiving an app should enable or disable the app once it is installed. The "noop" value is for apps that do not require enablement; for example, apps containing only Splunk knowledge, such as event or source types.	enabled

machineTypes	Matches any of the machine types in a comma-separated list.
	
	This setting lets you use the hardware type of the deployment client as a filter. This filter will be used only if a client could not be matched using the whitelist/blacklist filters. The easiest way to ensure that Splunk uses `machineTypes` as the filter is to add this setting to the top of your `serverclass.conf` file:
	
	`[global]`
	`blacklist.0=*`
	
	Note: `machineTypes` will have no effect if used in conjunction with `whitelist.0=*`. This is because `whitelist.0=*` causes a match on all clients, and `machineTypes` only gets used if no clients are matched through whitelist or blacklist filters.
		n/a

	The value for `machineTypes` is a comma-separated list of of machine types; for example, linux-i686, linux-x86_64, etc. Each machine type is a specific string designated by the hardware platform itself. The method for finding this string on the client varies by platform, but if the deployment client is already connected to the deployment server, you can determine the string's value by using this Splunk CLI command on the deployment server: `./splunk list deploy-clients` This will return a value for `utsname` that you can use to specify `machineTypes`. This setting will match any of the machine types in a comma-delimited list. Commonly-used machine types are linux-x86_64, windows-intel, linux-i686, freebsd-i386, darwin-i386, sunos-sun4u, linux-x86_64, sunos-i86pc, freebsd-amd64. **Note:** Be sure to include the 's' at the end of "machineTypes"	
restartSplunkWeb	Set to "true" or "false". Determines whether the client's Splunk Web restarts after the installation of a server class or app.	false
restartSplunkd	Set to "true" or "false". Determines whether the client's `splunkd` restarts after the installation of a server class or app.	false

Note: The most accurate and up-to-date 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 for `serverclass.conf` in serverclass.conf in the **Configuration file reference** in this manual, or in `$SPLUNK_HOME/etc/system/README`.

Examples

Here are several examples of defining server classes in the `serverclass.conf` file:

```
# 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 -
# in this case, $SPLUNK_HOME/etc/apps


# Example 2
# Assign server classes based on hostnames.

[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]
# blacklist.0=* at the global level ensures that the machineTypes filter
# invoked later will apply.
blacklist.0=*

[serverClass:AppsByMachineType]
# Include all machineTypes used by apps in this server class.
# 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 also succesfully matched.
machineTypes=windows-intel, linux-i686, linux-x86_64

[serverClass:AppsByMachineType:app:SplunkDesktop]
# Deploy this app only to Windows boxes.
machineTypes=windows-intel

[serverClass:AppsByMachineType:app:unix]
# Deploy this app only to unix boxes - 32/64 bit.
machineTypes=linux-i686, linux-x86_64
```

Configure deployment clients

Configure deployment clients

This topic describes the options for setting up **deployment clients** when using the Splunk deployment server functionality. A deployment client belongs to one or more **server classes**. Two configuration files share primary responsibility for determining how deployment functions:

- The **deployment server** has a serverclass.conf file, which specifies the server classes it deploys to. A server class defines what content a given deployment client should download. See "Define server classes" for details on how to configure this file.
- Each deployment client has a deploymentclient.conf file, which specifies what deployment server it should contact to get content, and where to put the content once it is downloaded. The current topic describes how to configure this file.

For information about configuration files, including an explanation of their basic structure, see About configuration files.

The next section provides a reference for the deployment client configuration settings. You might want to read it while referring to the set of simple example configurations presented later in this topic. In addition, there are several longer and more complete examples presented later in this manual, including "Deploy several standard forwarders".

What you can define for a deployment client

To enable a Splunk instance as a deployment client, create a `deploymentclient.conf` in `$SPLUNK_HOME/etc/system/local`.

You can also enable a deployment client through the CLI. See "Enable and disable deployment clients using the CLI" later in this topic.

The `deploymentclient.conf` file provides two **stanzas**:

Stanza	Meaning
`[deployment-client]`	Includes a number of configuration attributes. Most importantly, this stanza specifies where to place downloaded content.
`[target-broker:deploymentServer]`	Specifies the location of this client's deployment server. "deploymentServer" is the default name for a deployment server.

These are the main attributes available in the `[deployment-client]` stanza:

Attribute	What it's for	
disabled	Set to "true" or "false". If "true", it disables the deployment client.	false
clientName		deploymentClient

	A name, or "tag," that the deployment server can use to filter on. It takes precedence over hostnames.	
workingDir	A temporary folder used by the deployment client to download server classes and applications.	$SPLUNK_HOME/var/run/depl
repositoryLocation	The repository location where apps are installed after being downloaded from a deployment server. Note that, for each app that is downloaded, deployment server may also specify a repository location to install it. The deployment client will use the `serverRepositoryLocationPolicy` attribute to determine which location to use.	$SPLUNK_HOME/etc/apps
serverRepositoryLocationPolicy	Set to "acceptSplunkHome", "acceptAlways", or "rejectAlways": • acceptSplunkHome - Accept the repository location supplied by the deployment server, if and only if it is rooted by $SPLUNK_HOME. • acceptAlways - Always accept the repository location supplied by the deployment server. • rejectAlways - Always reject the server-supplied repository location. Instead, use the `repositoryLocation` specified in this configuration file.	acceptSplunkHome
endpoint	The HTTP endpoint from which content should be downloaded. **Note:** The deployment server can specify a different endpoint from which to download each set of content (individual apps, etc). The deployment client uses the `serverEndpointPolicy` atrtribute to determine which value to use. `$deploymentServerUri$` resolves to `targetUri`, defined in the	$deploymentServerUri$/service

	[target-broker] stanza. $serviceClassName$ and $appName$ mean what they say.	
serverEndpointPolicy	Set to "acceptAlways" or "rejectAlways": • acceptAlways - Always accept the endpoint supplied by the deployment server. • rejectAlways - Always reject the endpoint supplied by the deployment server. Instead, use the endpoint defined by the endpoint attribute.	acceptAlways
phoneHomeIntervalInSecs	A number that determines how frequently the deployment client should check for new content.	60

This is the attribute available in the [target-broker:deploymentServer] stanza:

Attribute	What it's for	Default
targetUri	Set to <Deployment_server_URI>:<Mgmt_port>. Specifies the deployment server connection information. The management port is typically 8089.	n/a

Note: The most accurate and up-to-date 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** in this manual, or in $SPLUNK_HOME/etc/system/README.

Examples

Here are several examples of defining deployment clients through the deploymentclient.conf file:

```
# Example 1
# Deployment client receives apps, placing them into the same repositoryLocation locally,
# relative to $SPLUNK_HOME, that 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/$serviceClassName$/$appName$.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>/$serviceClassName$/$appName$.tar

[target-broker:deploymentServer]
targetUri= deploymentserver.splunk.mycompany.com:8089
```

Enable and disable deployment clients using the CLI

To enable a deployment client, run the following command from the Splunk CLI:

```
./splunk set deploy-poll <IP address or hostname>:<port>
```

Include the IP address/hostname and management port of the deployment server. The management port is typically 8089.

Now restart the deployment client to make the change take effect.

To disable a deployment client, run the following command:

```
./splunk disable deploy-client
```

Deploy in multi-tenant environments

Note: It is recommended that you work with Splunk Professional Services when designing a multi-tenant deployment. It is best not to edit `tenants.conf` on your own.

A **multi-tenant deployment server** topology means that you have more than one **deployment server** running on the same Splunk instance, and each deployment server is serving content to its own set of **deployment clients**. (You can also achieve the same effect by using two Splunk instances, each with its own configuration.)

Use tenants.conf to redirect incoming requests from deployment clients to another deployment server or servers. The typical reason for doing this is to offload your **splunkd's** HTTP server -- having many deployment clients hitting the splunkd HTTP server at once to download apps and configurations can overload the deployment server. Over 400 connections at one time has been shown to bog down splunkd's HTTP server, but this does not take into account hardware or the size of the package the client is downloading -- this will be constrained by bandwidth size.

To set up multiple deployment servers on a single Splunk instance, you:

- Create a `tenants.conf` containing a whitelist or blacklist that tells deployment clients which deployment server instance to use.
- Create a separate instance of `serverclass.conf` for each deployment server, named for that deployment server, like so: `<tenantName>-serverclass.conf`.
- For each deployment client, configure `deploymentclient.conf` the way you would if there were just one deployment server.

For information about configuration files, including an explanation of their basic structure, see About configuration files.

What you can define in tenants.conf

You identify the different deployment servers as "tenants" in `tenants.conf` on the Splunk instance that will host these deployment servers. There isn't a `tenants.conf` file by default, so you must create one in `$SPLUNK_HOME/etc/system/local` and define the tenants in it.

For each tenant, create a stanza with the heading `[tenant:<tenantName>]` with these attributes:

Attribute	What it's for	Default
filterType	Set to "whitelist" or "blacklist". Determines the type of filter to use. Deployment clients use the filter to determine which deployment server to access.	whitelist
whitelist.<n> blacklist.<n>	<n> is a number starting at 0, and incrementing by 1. The client stops looking at the filter when <n> breaks. Set the attribute to ipAddress, hostname, or clientName: • ipAddress is the IP address of the deployment client. Can use wildcards, such as 10.1.1.*	n/a

	• hostname is the host name of deployment client. Can use wildcards, such as *.splunk.com. • clientName is a logical or tag name that can be assigned to a deployment client in `deploymentclient.conf`. clientName takes precedence over ipAddress or hostname when matching a client to a filter.	

Example

Here is an example of defining two tenants in the `tenants.conf` file:

```
# Define two tenants - dept1 and dept2.
# Deployment server configuration for dept1 will be in a matching dept1-serverclass.conf
# Deployment server configuration for dept2 will be in a matching dept2-serverclass.conf

[tenant:dept1]
whitelist.0=*.dept1.splunk.com

[tenant:dept2]
whilelist.0=*.dept2.splunk.com
```

Deploy apps and configurations

Deploy apps and configurations

After configuring the **deployment server** and **clients**, two steps remain:

1. Put the new or updated deployment content into deployment directories.

2. Inform the clients that it's time to download new content.

Create directories for deployment apps and place content in them

The location of deployment directories is configurable by means of the `repositoryLocation` attribute in `serverclass.conf`, as described in "Define server classes". The default location for deployment directories is `$SPLUNK_HOME/etc/deployment-apps`. Each app must have its own subdirectory, with the same name as the app itself, as specified in serverclass.conf.

This example creates a deployment directory in the default repository location, for an app named "fwd_to_splunk1":

```
mkdir ?p $SPLUNK_HOME/etc/deployment-apps/fwd_to_splunk1/default
```

Place the content for each app into the app's subdirectory. To update the app by with new or changed content, just add or overwrite the files in the directory.

Inform clients of new content

When you first configure the deployment server, and whenever you update its configuration by editing `serverclass.conf`, you'll need to restart or reload it for the changes to take effect. The clients will then pick up any new or changed content in configuration and in apps. You may use the CLI `reload`

command instead of restarting the server.

To use Splunk's CLI, change to the `$SPLUNK_HOME/bin/` directory and use the `./splunk` command. (On Windows, you do not need to include the ./ beforehand.)

This command checks all server classes for change and notifies the relevant clients:

```
./splunk reload deploy-server
```

This command notifies and updates only the server class you specify:

```
./splunk reload deploy-server -class <server class>
```

For example:

```
./splunk reload deploy-server -class www
```

In this example, the command notifies and updates only the clients that are members of the www server class.

Once a client receives new configurations, it restarts `splunkd` and enables the relevant apps, if configured to do so in `serverclass.conf`.

Confirm the deployment update

To confirm that all clients received the configuration correctly, run this command from the deployment server:

```
./splunk list deploy-clients
```

This lists all the deployment clients and specifies the last time they were successfully synced.

Example: deploy a light forwarder

Example: deploy a light forwarder

This example walks through the configuration needed to deploy an app, in this case, the Splunk **light forwarder**.

On the deployment server

1. Copy the SplunkLightForwarder app from `$SPLUNK_HOME/etc/apps` to the deployment directory, `$SPLUNK_HOME/etc/deployment-apps` on the **deployment server**.

2. Edit `serverclass.conf` in `/system/local` on the deployment server. Add a server class named "lightforwarders" that includes the light forwarder app:

```
[global]
whitelist.0=*

[serverClass:lightforwarders]
```

```
whitelist.0=*

[serverClass:lightforwarders:app:SplunkLightForwarder]
stateOnClient=enabled
restartSplunkd=true
```

Note the following:

- The `[global]` stanza is required. It contains any settings that should be globally applied.
  - In the `[global]` stanza, `whitelist.0=*` signifies that all of the deployment server's clients match all server classes defined in this configuration file. In this example, there is just a single server class.
- The server class name is "lightforwarders". You can call your server classes anything you want.
  - In the `[serverClass:lightforwarders]` stanza, `whitelist.0=*` signifies that all clients match the lightforwarders server class.
- The `[serverClass:lightforwarders:app:SplunkLightForwarder]` stanza contains settings specific to the SplunkLightForwarder app on the lightforwarders server class.
  - `stateOnClient` specifies that this app should be enabled on the client when it is deployed.
  - `restartSplunkd` specifies that when this app is deployed, splunkd should be restarted.

See "Define server classes" for details on how to configure this file.

On the deployment client

Edit `deploymentclient.conf` in `/system/local` on the **deployment client** to tell the client how to contact the deployment server:

```
[deployment-client]
[target-broker:deploymentServer]
targetURI=<IP:port>
```

Note the following:

- `deploymentServer` is the default name for a deployment server.
- `<IP:port>` is the IP address and port number for this client's deployment server.

The file points the client to the deployment server located at `IP:port`. There, it will pick up the Splunk light forwarder app, enable it, and restart. See "Configure deployment clients" for details on how to configure this file.

Extended example: deploy several standard forwarders

Extended example: deploy several standard forwarders

What we're aiming for

A common use for **deployment servers** is to manage **forwarder** configuration files. In some distributed environments, forwarders can number into the hundreds, and the deployment server

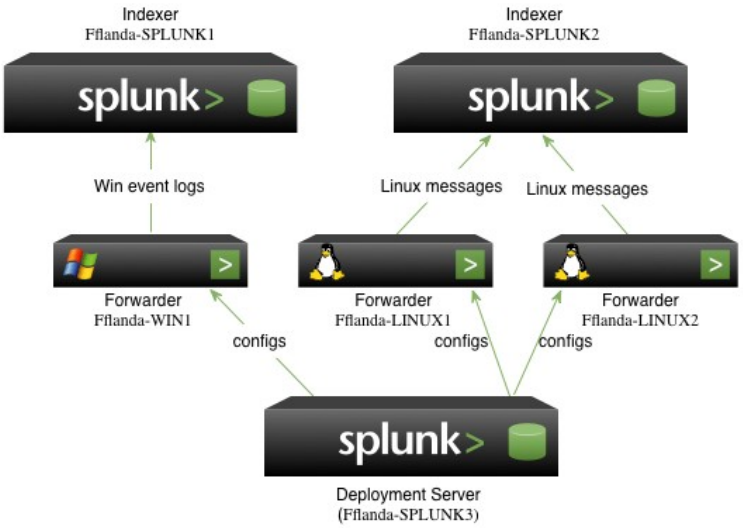
greatly eases the work of configuring and updating them. This example shows how to use the deployment server to initially configure a set of dissimilar forwarders. A follow-up example, in "Example: add an input to forwarders", shows how to use the deployment server to update the forwarders' configurations with new inputs.

The example sets up the following distributed environment, in which a deployment server deploys configurations for three forwarders sending data to two **indexers**:

- The deployment server Fflanda-SPLUNK3 (10.1.2.4) manages deployment for these forwarders:
  - Fflanda-WIN1
  - Fflanda-LINUX1
  - Fflanda-LINUX2
- Fflanda-WIN1 forwards Windows event logs to the receiving indexer Fflanda-SPLUNK1 (10.1.2.2).
- Fflanda-LINUX1 and Fflanda-LINUX2 forward Linux messages to the receiving indexer Fflanda-SPLUNK2 (10.1.2.3).

The forwarders are **deployment clients**, receiving their configuration files from the deployment server.

Here's the basic set up:



For information on forwarders, see "About forwarding and receiving" in this manual.

For information on monitoring Windows event log data, see "Monitor Windows event log data" in this manual.

For information on monitoring files, such as message logs, see "Monitor files and directories" in this manual.

Overview of the set up

Here's an overview of the set up process (the detailed steps follow in the next section):

On the deployment server:

1. Create the set of **server classes** and apps for the deployment clients (forwarders) with `serverclass.conf`. You'll create two server classes to represent the two OS types (Windows, Linux). For each server class, you'll also create two deployment apps, for a total of four apps. The apps encapsulate:

- The type of input -- the data that the forwarder will monitor (Windows event logs or Linux messages).
- The type of output -- the indexer the forwarder will send data to (SPLUNK1 or SPLUNK2).

This configuration results in each forwarder belonging to a server class and receiving two apps: one for its inputs and one for its outputs.

2. Create directories to hold the deployment apps.

3. Create configuration files (`outputs.conf` and `inputs.conf`) to deploy to the forwarders. These files constitute the deployment apps and reside in the app directories.

4. Restart the deployment server.

On each Splunk indexer that will be receiving data from forwarders:

1. Enable receiving through the Splunk CLI.

2. Restart the receiver.

On each forwarder/deployment client:

1. Create a `deploymentclient.conf` file that points to the deployment server.

2. Restart the forwarder.

The rest is Splunk magic. After a short delay (while the forwarders receive and act upon their deployed content), Windows event logs begin flowing from Fflanda-WIN1 to Fflanda-SPLUNK1, and /var/log/messages begin flowing from Fflanda-LINUX1 and Fflanda-LINUX2 to Fflanda-SPLUNK2.

Detailed configuration steps

On the deployment server, Fflanda-SPLUNK3:

1. Install Splunk, if you haven't already done so.

2. Set up your server classes. Create `$SPLUNK_HOME/etc/system/local/serverclass.conf` with the following settings:

```
# Global server class
[global]
# Filter (whitelist) all clients
whitelist.0=*

# Server class for Windows
[serverClass:Fflanda-WIN]
# Filter (whitelist) all Windows clients
whitelist.0=Fflanda-WIN*

# App for inputting Windows event logs
# This app is only for clients in the server class Fflanda-WIN
[serverClass:Fflanda-WIN:app:winevt]
#Enable the app and restart Splunk, after the client receives the app
stateOnClient=enabled
restartSplunkd=true

# App for forwarding to SPLUNK1
# This app is only for clients in the server class Fflanda-WIN
[serverClass:Fflanda-WIN:app:fwd_to_splunk1]
stateOnClient=enabled
restartSplunkd=true

# Server class for Linux
[serverClass:Fflanda-LINUX]
# Filter (whitelist) all Linux clients
whitelist.0=Fflanda-LINUX*

# App for inputting Linux messages
# This app is only for clients in the server class Fflanda-LINUX
[serverClass:Fflanda-LINUX:app:linmess]
stateOnClient=enabled
restartSplunkd=true

# App for forwarding to SPLUNK2
# This app is only for clients in the server class Fflanda-LINUX
[serverClass:Fflanda-LINUX:app:fwd_to_splunk2]
stateOnClient=enabled
restartSplunkd=true
```

See "Define server classes" for details on how to configure this file.

3. Create the app deployment directories with the following commands:

```
mkdir ?p $SPLUNK_HOME/etc/deployment-apps/fwd_to_splunk1/default
mkdir ?p $SPLUNK_HOME/etc/deployment-apps/fwd_to_splunk2/default
mkdir ?p $SPLUNK_HOME/etc/deployment-apps/winevt/default
mkdir ?p $SPLUNK_HOME/etc/deployment-apps/linmess/default
```

Each app gets its own directory, so that they can be deployed individually. In addition, the directory name determines the name of the app.

4. Create `$SPLUNK_HOME/etc/deployment-apps/fwd_to_splunk1/default/outputs.conf` with the following settings:

```
[tcpout]
```

```
defaultGroup=splunk1

[tcpout:splunk1]
# Specifies the server that receives data from the forwarder.
server=10.1.2.2:9997
```

For information on `outputs.conf`, see "Configure forwarders with outputs.conf" in this manual.

5. Create `$SPLUNK_HOME/etc/deployment-apps/fwd_to_splunk2/default/outputs.conf` with the following settings:

```
[tcpout]
defaultGroup=splunk2

[tcpout:splunk2]
server=10.1.2.3:9997
```

6. Create `$SPLUNK_HOME/etc/deployment-apps/winevt/default/inputs.conf` with the following settings:

```
[WinEventLog:Application]
disabled=0

[WinEventLog:Security]
disabled=0

[WinEventLog:System]
disabled=0
```

For information on monitoring Windows event log data, see "Monitor Windows event log data" in this manual.

7. Create `$SPLUNK_HOME/etc/deployment-apps/linmess/default/inputs.conf` with the following settings:

```
[monitor:///var/log/messages]
disabled=false
followTail=1
sourcetype=syslog
```

For information on monitoring files, such as message logs, see "Monitor files and directories" in this manual.

8. Restart Splunk.

Note: Because the deployment server in this example is newly configured, it requires a restart for its configuration to take effect. When clients poll the server for the first time, they'll get all the content designated for them. To deploy subsequent content, you generally do not need to restart the server. Instead, you just invoke the Splunk CLI `reload` command on the server, as described in "Deploy apps and configurations". By doing so, you ensure that the server will inform its clients of content changes. However, whenever you edit `serverclass.conf`, you must always restart the deployment server for the configuration changes to take effect.

On each receiver, Fflanda-SPLUNK1 and Fflanda-SPLUNK2:

1. Install Splunk, if you haven't already done so.

2. Run the following CLI command:

./splunk enable listen 9997 -auth <username>:<password>

This specifies that the receiver will listen for data on port 9997. With proper authorization, any forwarder can now send data to the receiver by designating the receiver's IP address and port number. You must enable receivers *before* you enable the forwarders that will be sending data to them.

For information on enabling receivers, see "Enable forwarding and receiving" in this manual.

3. Restart Splunk.

On each forwarder, Fflanda-WIN1, Fflanda-LINUX1, and Fflanda-LINUX2:

1. Install Splunk, if you haven't already done so.

2. Create $SPLUNK_HOME/etc/system/local/deploymentclient.conf with the following settings:

```
[deployment-client]

[target-broker:deploymentServer]
# Specify the deployment server that the client will poll.
targetUri= 10.1.2.4:8089
```

See "Configure deployment clients" for details on how to configure this file.

3. Restart Splunk.

Each forwarder will now poll the deployment server, download its configuration files, restart, and begin forwarding data to its receiving indexer.

For a follow-up example showing how to use the deployment server to update forwarder configurations, see "Example: Add an input to forwarders".

What the communication between the deployment server and its clients looks like

Using the above example, the communication from Fflanda-WIN1 to Fflanda-SPLUNK3 on port 8089 would look like this:

Fflanda-WIN1: Hello, I am Fflanda-WIN1.

Fflanda-SPLUNK3: Hello, Fflanda-WIN1. I have been expecting to hear from you. I have you down as a member of the Fflanda-WIN server class, and you should have the fwd_to_splunk1 (checksum=12345) and winevt (checksum=12378) apps.

Fflanda-WIN1: Hmmm, I don?t have those configs. Using this connection I just opened up to you, can I grab the configs from you?

Fflanda-SPLUNK3: Sure! I have them ready for you.

Fflanda-WIN1: Thanks! I am going to back off a random number of seconds between 1 and 60 (in case you have a lot of clients that are polling you at the moment) ... OK, now send me the files.

Fflanda-SPLUNK3: Done! You now have fwd_to_splunk1-timestamp.bundle and winevt-timestamp.bundle.

Fflanda-WIN1: Awesome! I am going to store them in my $SPLUNK_HOME/etc/apps directory. Now I am going to restart myself, and when I come back up I am going to read the configurations that you sent me directly out of the .bundle files, which I know are just tar balls with a different extension.

A couple of minutes go by....

Fflanda-WIN1: Hello, I am Fflanda-WIN1.

Fflanda-SPLUNK3: Hello, Fflanda-WIN1. I have been expecting to hear from you. I have you down as a member of the Fflanda-WIN server class, and you should have the fwd_to_splunk1 (checksum=12345) and winevt (checksum=12378) Apps.

Fflanda-WIN1: Hmmm, I already have both of those, but thanks anyway!

Later on, an admin modifies the `winevt/inputs.conf` *file on Fflanda-SPLUNK3 to disable the collection of system event logs, and then runs the CLI command* `splunk reload deploy-server` *to force the deployment server to rescan serverclass.conf and the app directories. The next time Fflanda-WIN1 talks to Fflanda-SPLUNK3, it goes like this:*

Fflanda-WIN1: Hello, I am Fflanda-WIN1.

Fflanda-SPLUNK3: Hello, Fflanda-WIN1. I have been expecting to hear from you. I have you down as a member of the Fflanda-WIN server class, and you should have the fwd_to_splunk1 (checksum=12345) and winevt (checksum=13299) Apps.

Fflanda-WIN1: Hmmm, I know I have those configs, but the checksum I have for the winevt configs is different than the one you just told me about. Using this connection I just opened up to you, can I grab the updated winevt config from you?

Fflanda-SPLUNK3: Sure! I have it ready for you.

Fflanda-WIN1: Thanks! I am going to back off a random number of seconds between 1 and 60 (in case you have a lot of clients that are polling you at the moment) ... Ok, now send me the updated config.

Fflanda-SPLUNK3: Done! You now have winevt-newer_timestamp.bundle.

Fflanda-WIN1: Awesome! I am going to store it my $SPLUNK_HOME/etc/apps directory and move the old winevt.bundle I had out of the way. Now I am going to restart myself, and when I come back up, I'll have the most up-to-date config.

Example: add an input to forwarders

Example: add an input to forwarders

The previous topic, "Extended example: deploy several standard forwarders", described setting up a deployment environment to manage a set of forwarders. It showed how to configure a new deployment server to deploy content to a new set of deployment clients. The current example follows on directly from there, using the configurations created in that topic. It shows how to update a forwarder configuration file and deploy the updated file to a subset of forwarders, defined by a server class.

Overview of the update process

This example starts with the set of configurations and Splunk instances created in the topic "Extended example: deploy several standard forwarders". The Linux forwarders now need to start monitoring data from a second source. To accomplish this:

1. Edit the `inputs.conf` file for the Linux server class to add the new source, overwriting the previous version in its apps directory.

2. Use CLI to reload the deployment server, so that it becomes aware of the change and can deploy it to the appropriate set of clients.

You need make changes only on the deployment server. When the deployment clients in the Linux server class next poll the server, they'll be notified of the new `inputs.conf` file. They'll download the file, enable it, restart Splunk, and immediately begin monitoring the second data source.

Detailed configuration steps

1. Edit `$SPLUNK_HOME/etc/deployment-apps/linmess/default/inputs.conf` to add a new input:

```
[monitor:///var/log/messages]
disabled=false
followTail=1
sourcetype=syslog

[monitor:///var/log/httpd]
disabled=false
sourcetype = access_common
```

2. Use Splunk CLI to reload the deployment server:

```
./splunk reload deploy-server -class Fflanda-LINUX
```

Once this command has been run, the deployment server notifies the clients that are members of the Fflanda-LINUX server class of the changed file. Since the change doesn't affect the Fflanda-WIN

server class, its members don't need to know about it.

Set up distributed search

What is distributed search?

What is distributed search?

In **distributed search**, Splunk servers send search requests to other Splunk servers and merge the results back to the user. In a typical scenario, one Splunk server searches indexes on several other servers.

These are some of the key use cases for distributed search:

- **Horizonal scaling for enhanced performance.** Distributed search provides horizontal scaling by distributing the indexing and searching loads across multiple indexers, making it possible to search and index large quantities of data.
- **Access control.** You can use distributed search to control access to indexed data. In a typical situation, some users, such as security personnel, might need access to data across the enterprise, while others need access only to data in their functional area.
- **Managing geo-dispersed data.** Distributed search allows local offices to access their own data, while maintaining centralized access at the corporate level. Chicago and San Francisco can look just at their local data; headquarters in New York can search its local data, as well as the data in Chicago and San Francisco.
- **Maximizing data availability.** Distributed search, combined with load balancing and cloning from forwarders, is a key component of high availability solutions.

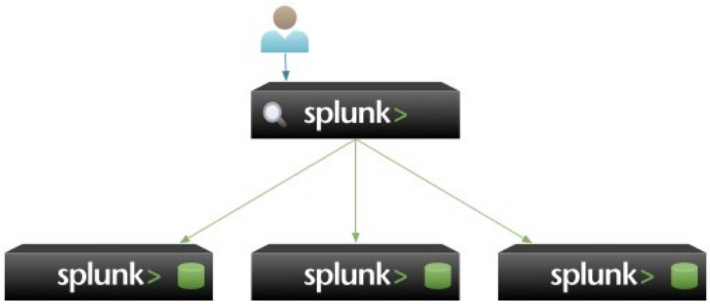
The Splunk instance that does the searching is referred to as the **search head**. The Splunk instances that do the indexing are called **search peers** or **indexer nodes**. Together, the search head and search peers constitute the nodes in a **distributed search cluster**.

A search head can also index and serve as a search peer. However, in performance-based use cases, such as horizontal scaling, it is recommended that the search head only search and not index. In that case, it is referred to as a **dedicated search head**.

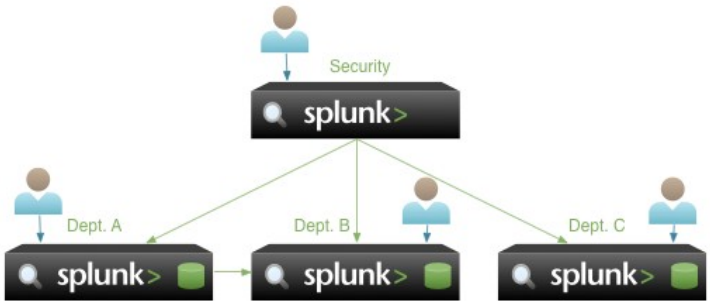
A search head by default runs its searches across all search peers in its cluster. You can limit a search to one or more search peers by specifying the `splunk_server` field in your query. See Search across one or more distributed servers in the User manual.

Some search scenarios

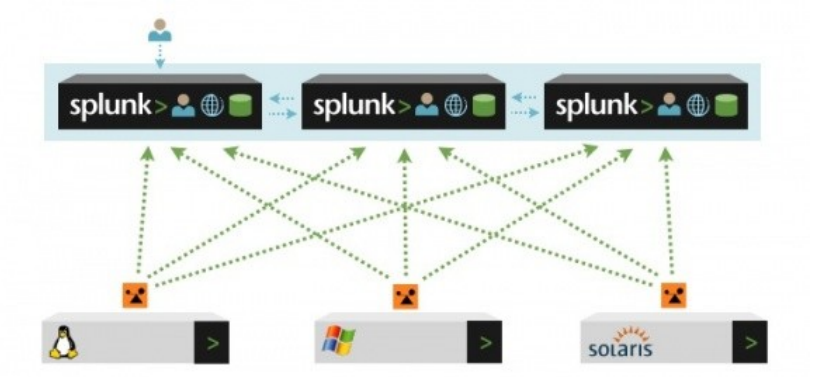
This diagram shows a simple distributed search scenario for horizontal scaling, with one search head searching across three peers:



In this diagram showing a distributed search scenario for access control, a "security" department search head has visibility into all the indexing search peers. Each search peer also has the ability to search its own data. In addition, the department A search peer has access to both its data and the data of department B:



Finally, this diagram shows the use of load balancing and distributed search to provide high availability access to data:



For more information on load balancing, see Set up load balancing in this manual.

For information on Splunk distributed searches and capacity planning, see Dividing up indexing and searching in the Installation manual.

What search heads send to search peers

When initiating a distributed search, the search head distributes its **knowledge objects** to its search peers. Knowledge objects include saved searches, event types, and other entities used in searching across indexes. The search head needs to distribute this material to its search peers so that they can properly execute queries on its behalf. See What is Splunk knowledge? for detailed information on knowledge objects.

The indexers use the search head's knowledge to execute queries on its behalf. When executing a distributed search, the indexers are ignorant of any local knowledge objects. They have access only to the search head's knowledge.

The process of distributing search head knowledge means that the indexers by default receive nearly the entire contents of all the search head's apps. This set of data is referred to as the **distributed bundle.** If an app contains large binaries that do not need to be shared with the indexers, you can reduce the size of the bundle by means of the `[replicationWhitelist]` stanza in `distsearch.conf`. See Limit distributed bundle size in this manual.

The distributed bundle gets distributed to `$SPLUNK_HOME/var/run/searchpeers/` on each search peer.

Because the search head distributes its knowledge, search scripts should not hardcode paths to resources. The distributed bundle will reside at a different location on the search peer's file system, so hardcoded paths will not work properly.

User authorization

All authorization for a distributed search originates from the search head. At the time it sends the search request to its search peers, the search head also distributes the authorization information. It tells the search peers the name of the user running the search, the user's role, and the location of the distributed `authorize.conf` file containing the authorization information.

Licenses for distributed deployments

Each indexer node in a distributed deployment requires a unique license key.

Search heads performing no indexing or only summary indexing can use the forwarder license. If the search head performs any other type of indexing, it requires a unique key.

See Search head license in the Installation manual for a detailed discussion of licensing issues.

Cross-version compatibility

All search nodes must be running Splunk 4.x to participate in the distributed search. Distributed search is not backwards or forwards compatible with Splunk 3.x.

Install a dedicated search head

Install a dedicated search head

Distributed search is enabled by default on every Splunk instance, with the exception of forwarders. This means that every Splunk server can function as a **search head** to a specified group of indexers, referred to as **search peers**.

In some cases, you might want a single Splunk instance to serve as both a search head and a search peer. In other cases, however, you might want to set up a **dedicated** search head. A dedicated search head performs only searching; it does not do any indexing. If you do not intend to perform any indexing on your search head, you can license it as a dedicated search head.

Note: If you **do** want to use a Splunk instance as both a search head and a search peer, or otherwise perform indexing on the search head, just install the search head as a regular Splunk instance with a normal license, as described in "About Splunk licenses" in the Installation manual.

To install a dedicated search head, follow these steps:

1. Determine your hardware needs by reading this topic in the Installation manual.

2. Install Splunk, as described in the topic in the Installation manual specific to your operating system.

3. Install a dedicated search head license (identical to a forwarder license), as described here in the Installation manual.

Note: Use the forwarder license for dedicated search heads only. **If the search head also performs indexing, it will need a full Splunk license.**

4. Establish distributed search from the search head to all the indexers, or "search peers", you want it to search. See "Configure distributed search" for how to do this.

5. Log in to your search head and do a search for *. Look at your results and check the `splunk_server` field. Veryify that all your search peers are listed in that field.

6. Set up the authentication method you want to use on the search head, just as you would for any other Splunk instance. Do **not** set up any indexing on your search head, since that will violate its license.

Configure distributed search

Configure distributed search

Distributed search is available on every Splunk server, with the exception of forwarders. This means that every Splunk server can function as a **search head** to a specified group of indexers, referred to as **search peers**. The distributed search capability is enabled by default.

To activate distributed search, a Splunk instance that you designate as a search head just needs to add search peers. Ordinarily, you do this by specifying each search peer manually. A number of other configuration options are available, but ordinarily you do not need to alter their default values.

No configuration is necessary on the search peers themselves to make them available to search heads. Access is controllable through public key authentication.

When configuring Splunk instances as search heads or search peers, keep this key distinction in mind:

- A **search head** must maintain a list of search peers, or it will have nothing to search on. A **dedicated search head** does not have additional data inputs, beyond the default ones.
- A **search peer** must have specified data inputs, or it will have nothing to index. A search peer does not maintain a list of other search peers.

These roles are not necessarily distinct. A Splunk instance can, and frequently does, function simultaneously as both a search head and a search peer.

You can set up a distributed search head via Splunk Web, the Splunk CLI, or by editing the `distsearch.conf` configuration file. Splunk Web is the recommended configuration method for most purposes.

Use Splunk Web

The main step in configuring a distributed search head is to specify its search peers. You can add these either manually (the usual case) or by enabling automatic discovery.

Add search peers manually

To specify search peers manually:

1. Log into Splunk Web and click **Manager**.

2. Click **Distributed search**.

3. On the **Search peers** line, select **Add new**.

4. Specify the search peer, along with any authentication settings.

5. Click **Save**.

Add search peers automatically, or otherwise configure distributed search

To configure automatic discovery, you need to go into **Distributed search setup** on both the search head and the search peers. You can configure other settings there as well.

On the search head:

1. Log into Splunk Web and click **Manager**.

2. Click **Distributed search**.

3. Click **Distributed search setup**.

4. Specify "Yes" for the option: "Automatically add other Splunk servers?".

5. Change any other settings as needed and click **Save**.

On each search peer:

1. Log into Splunk Web and click **Manager**.

2. Click **Distributed search**.

3. Click **Distributed search setup**.

4. Specify "Yes" for the option: "Broadcast to other Splunk servers?"

5. Change any other settings as needed and click **Save**.

Use the CLI

Follow these instructions to enable distributed search via Splunk's CLI.

To use Splunk's CLI, navigate to the `$SPLUNK_HOME/bin/` directory and use the `./splunk` command.

Enable distributed search

Distributed search is enabled by default, so this step is ordinarily not required:

```
splunk enable dist-search -auth admin:changeme
```

Add a search peer manually

Use the `splunk add search-server` command to add a search peer. When you run this command, make sure you specify the splunkd management port of the peer server. By default, this is `8089`, although it might be different for your deployment.

Be sure to provide credentials for both the local and the remote machines. Use the `-auth` flag for your local credentials and the `-remoteUsername` and `-remotePassword` flags to specify the remote credentials (in this example, for search peer `10.10.10.10`):

```
splunk add search-server -host 10.10.10.10:8089 -auth admin:changeme -remoteUsername admin -rem
```

Specify automatic discovery

```
splunk enable discoverable -auth admin:changeme
```

Each of these commands will elicit a response from the server indicating success and the need to restart the server for the changes to take effect.

Edit distsearch.conf

In most cases, the settings available through Splunk Web provide sufficient options for configuring distributed search environments. Some advanced configuration settings, however, are only available through `distsearch.conf`. Edit this file in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`.

For the detailed specification and examples, see distsearch.conf.

For more information on configuration files in general, see "About configuration files".

Note: As of 4.1, the `blacklistNames` and `blacklistURLs` attributes no longer have any effect on distributed search behavior.

Distribute the key files

If you add search peers via Splunk Web or the CLI, Splunk automatically handles authentication. However, if you add peers by editing `distsearch.conf`, you must distribute the key files manually.

After enabling distributed search on a Splunk instance (and restarting the instance), you will find the keys in this location: `$SPLUNK_HOME/etc/auth/distServerKeys/`

Distribute the file `$SPLUNK_HOME/etc/auth/distServerKeys/trusted.pem` on the search head to `$SPLUNK_HOME/etc/auth/distServerKeys/<searchhead_name>/trusted.pem` on the indexer nodes.

Support for keys from multiple Splunk instances

Any number of Splunk instances can have their certificates stored on other instances for authentication. The instances can store keys in `$SPLUNK_HOME/etc/auth/distServerKeys/<peername>/trusted.pem`

For example, if you have Splunk search heads A and B and they both need to search Splunk index node C, do the following:

1. On C, create `$SPLUNK_HOME/etc/auth/distServerKeys/A/` and `etc/auth/distServerKeys/B/`.

2. Copy A's `trusted.pem` to `$SPLUNK_HOME/etc/auth/distServerKeys/A/` and B's `trusted.pem` to `$SPLUNK_HOME/etc/auth/distServerKeys/B/`.

3. Restart C.

Limit distributed bundle size

The distributed knowledge bundle is the data that the search head replicates to each search peer to enable its searches. For information on the contents and purpose of this bundle, see "What search heads send to search peers".

To limit the size of the distributed bundle, you can create a replication whitelist. To do this, edit `distsearch.conf` and specify a `[replicationWhitelist]` stanza:

```
[replicationWhitelist]
<name> = <whitelist_regex>
...
```

All files that satisfy the whitelist regex will be included in the bundle that the search head distributes to its search peers. If multiple regex's are specified, the bundle will inlucde the union of those files.

In this example, the knowledge bundle will include all files with extensions of either ".conf" or ".spec":

```
[replicationWhitelist]
allConf = *.conf
allSpec = *.spec
```

The names, such as allConf and allSpec, are used only for layering. That is, if you have both a global and a local copy of `distsearch.conf`, the local copy can be configured so that it overrides only one of the regex's. For instance, assume that the example shown above is the global copy. Assume you then specify a whitelist in your local copy like this:

```
[replicationWhitelist]
allConf = *.foo.conf
```

The two conf files will be layered, with the local copy taking precedence. Thus, the search head will distribute only files that satisfy these two regex's:

```
allConf = *.foo.conf
allSpec = *.spec
```

For more information on attribute layering in configuration files, see "Attribute precedence" in this manual.

Manage distributed server names

The name of each search head and search peer is determined by its `serverName` attribute, specified in server.conf. serverName defaults to the server's machine name.

In a distributed search cluster, all nodes must have unique names. The serverName has three specific uses:

- For authenticating search heads. When search peers are authenticating a search head, they look for the search head's key file in `/etc/auth/distServerKeys/<searchhead_name>/trusted.pem`.
- For identifying search peers in search queries. serverName is the value of the `splunk_server` field that you specify when you want to query a specific node. See "Search across one or more distributed search peers" in the User manual.
- For identifying search peers in search results. serverName gets reported back in the `splunk_server` field.

Note: serverName is *not* used when adding search peers to a search head. In that case, you identify the search peers through their domain names or IP addresses.

The only reason to change serverName is if you have multiple instances of Splunk residing on a single machine, and they're participating in the same distributed search cluster. In that case, you'll need to change serverName to distinguish them.

Use distributed search

Use distributed search

From the user standpoint, specifying and running a **distributed search** is essentially the same as running any other search. Behind the scenes, the **search head** distributes the query to its **search peers** and consolidates the results when presenting them to the user.

Your users do not have the ability to specify which search peers participate in a search. They do need to be aware of the distributed search configuration to troubleshoot.

Perform distributed searches

In general, you specify a distributed search through the same set of commands as for a local search. However, Splunk provides several additional commands and options to assist with controlling and limiting a distributed search.

A search head by default runs its searches across all search peers in its cluster. You can limit a search to one or more search peers by specifying the `splunk_server` field in your query. See Search across one or more distributed servers in the User manual.

The search command `localop` is also of use in defining distributed searches. It enables you to limit the execution of subsequent commands to the search head. See the description of localop in the Search Reference for details and an example.

In addition, the `lookup` command provides a `local` argument for use with distributed searches. If set to `true`, the lookup occurs only on the search head; if `false`, the lookup occurs on the search peers as well. This is particularly useful for scripted lookups, which replicate lookup tables. See the description of lookup in the Search Reference for details and an example.

Troubleshoot the distributed search

This table lists some of the more common search-time error messages associated with distributed search:

Error message	Meaning
`status=down`	The specified remote peer is not available.
`status=not a splunk server`	The specified remote peer is not a Splunk server.
`duplicate license`	The specified remote peer is using a duplicate license.
`certificate mismatch`	Authentication with the specified remote peer failed.

Manage search jobs

About jobs and job management

About jobs and job management

When a user runs a search in Splunk, it is created as a "job" in the system. This job also includes the artifacts (like search results) that are returned by a given search. Users can pause and resurrect their own jobs in the Job Manager. As an admin, you can manage the jobs of all users in the system.

To access the Jobs manager, click **Jobs** in the upper right of Splunk Web.

Note: The number of jobs shown in parentheses next to the **Jobs** link is the number of jobs that the user you're logged in as is currently running, not the number of jobs running on the system as a whole, even if you're logged in as admin.

You can also manage jobs through the command line of your OS.

Restrict the jobs users can run

The way to restrict how many jobs a given user can run, and how much space their job artifacts can take up is to define a role with these restrictions and assign them to it. You can do this at a very high level of granularity; each user in your system can have their own role.

Create a capability in a copy of `authorize.conf` in `$SPLUNK_HOME/etc/system/local` and give it appropriate values of:

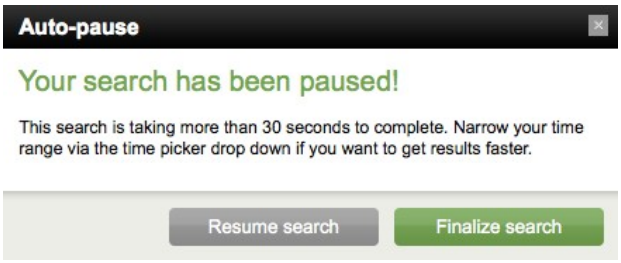
- srchDiskQuota: Maximum amount of disk space (MB) that can be taken by search jobs of a user that belongs to this role.
- srchJobsQuota: Maximum number of concurrently running searches a member of this role can have.

For more information, refer to the topic about creating roles in this manual.

Autopause long-running jobs

To handle inadvertently long-running search jobs, Splunk provides an autopause feature. The feature is enabled by default only for summary dashboard clicks, to deal with the situation where users mistakenly initiate "all time" searches.

When autopause is enabled for a particular search view, the search view includes an autopause countdown field during a search. If the search time limit has been reached, an information window will appear to inform the user that the search has been paused. It offers the user the option of resuming or finalizing the search. By default, the limit before autopause is 30 seconds.



Auto-pause is configurable only by view developers. It is not a system-wide setting nor is it configurable by role. The autopause feature can be enabled or disabled by editing the appropriate view. See How to turn off autopause in the Developer manual. Also, see the `host, source,` and `sourcetypes` links on the summary dashboard for examples of autopause implementation.

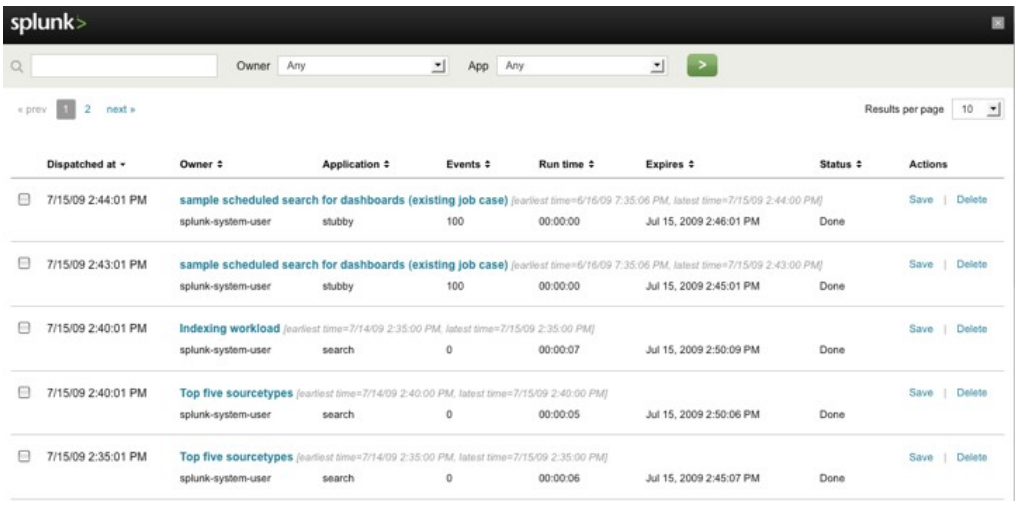
Manage jobs in Splunk Web

Manage jobs in Splunk Web

As the admin user, you can manage jobs run by all other users on the system. You can access these jobs through the **Jobs** manager in Splunk Web.

Note: The number of jobs shown in parentheses next to the Jobs link is the number of jobs that the user you're logged in as is currently running, not the number of jobs running on the system as a whole, even if you're logged in as admin.

The Jobs manager launches a pop-up window showing all the jobs currently running on the system.



Use the controls to save, pause, delete, resume, delete, and finalize jobs on the system. Select the checkbox to the left of the item you want to act on, and click the relevant button at the bottom of the page.

Unsaved search jobs expire within a set period of time after they complete. This means that the artifacts (including their results) are removed from the filesystem and cannot be retrieved unless

you've explicitly saved the job.

The default lifespan for manually run search jobs is 15 minutes (search jobs resulting from scheduled searches typically have much shorter lifespans). The Expires column tells you how much time each listed job has before it is deleted from the system. If you want to be able to review a search job after that expiration point, or share it with others, save it.

Manage jobs in the OS

Manage jobs in the OS

When Splunk is running a job, it will manifest itself as a process in the OS called `splunkd search`. You can use Manager to act on this job, but you can also manage the job's underlying processes at the OS commandline as well.

To see the job's processes and its arguments, type:

```
> top
> c
```

This will show you all the processes running and all their arguments.

Typing `ps -ef | grep "splunkd search"` will isolate all the splunk search processes within this list. It looks like this:

```
[pie@fflanda ~]$ ps -ef | grep "splunkd search"
pie  21368 19460 96 13:51 ?        00:01:18 splunkd search --search=search sourcetype="access_c
pie  21371 21368  0 13:51 ?        00:00:00 splunkd search --search=search sourcetype="access_c
pie  22804 20379  0 13:52 pts/9    00:00:00 grep splunk-search
```

There will be two processes for each search job; the second one is a 'helper' process used by the <codesplunkd</code> process to do further work as needed. The main job is the one using system resources. The helper process will die on its own if you kill the main process.

The process info includes:

- the search string (search=)
- the job ID for that job (id=)
- the ttl, or length of time that job's artifacts (the output it produces) will remain on disk and available (ttl=)
- the user who is running the job (user=)
- what role(s) that user belongs to (roles=)

When a job is running, its data is being written to `$SPLUNK_HOME/var/run/splunk/dispatch/<job_id>/` Scheduled jobs (scheduled saved searches) include the saved search name as part of the directory name.

The value of `ttl` for a process will determine how long the data remains in this spot, even after you kill a job. When you kill a job from the OS, you might want to look at its job ID before killing it if you want to also remove its artifacts.

Use the filesystem to manage jobs

Splunk allows you to manage jobs via creation and deletion of items in that job's artifact directory:

- To cancel a job, go into that job's artifact directory create a file called 'cancel'.
- To preserve that job's artifacts (and ignore its ttl setting), create a file called 'save'.
- To pause a job, create a file called 'pause', and to unpause it, delete the 'pause' file.

Use Splunk's command line interface (CLI)

About the CLI

About the CLI

You can use the Splunk CLI to monitor, configure, and execute searches on your Splunk server. Your Splunk role configuration dictates what actions (commands) you can execute. Most actions require you to be a Splunk administrator.

How to access the CLI

To access Splunk CLI, you need either:

- Shell access to a Splunk server, or
- Permission to access the correct port on a remote Splunk server.

If you have administrator or root privileges you can simplify CLI usage by adding the top level directory of your Splunk installation to your shell path. The `$SPLUNK_HOME` variable refers to the top level directory. Set a `SPLUNK_HOME` environment variable and add `$SPLUNK_HOME/bin` to your shell's path.

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

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

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

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

CLI commands

If you have administrator privileges, you can use the CLI not only to search but also to configure and monitor your Splunk server (or servers). The CLI commands used for configuring and monitoring Splunk are not search commands. Search commands are arguments in the `search` and `dispatch` CLI commands.

You can find all CLI documentation in the CLI help reference. For the list of CLI commands, type:

```
./splunk help commands
```

Or, access the help page about Splunk search commands with:

```
./splunk help search-commands
```

- For more information, see "Get help with the CLI".

- For details on syntax for searching using the CLI, refer to "About CLI Searches" in the Search Reference Manual.

Note for Mac users

Mac OS X requires you to have 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.)

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

Get help with the CLI

Find a complete CLI help reference by using the command `help`.

Access the default CLI help page by typing the following in the command line while Splunk is running:

```
./splunk help
```

Access help on a specific CLI command, or topic by typing:

```
./splunk help command name | topic name
```

For example, access a help page about Splunk **CLI commands**:

```
./splunk help commands
```

Or, access a help page about Splunk **search commands**:

```
./splunk help search-commands
```

Note: Notice the dash (-) between the words: "search" and "commands". This is because the Splunk CLI interprets spaces as breaks. Use dashes between multiple words for topic names that are more than one word.

Working with the CLI on Windows

To access CLI help and run CLI commands on Windows, be sure to run cmd.exe as administrator first. It's also not necessary to type the ./ before each "splunk" command.

Some commands require authentication or target host information

Use the `auth` and `uri` parameters with any CLI command.

auth

Use `auth` with commands that require authentication to execute. `auth` is 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

Use `uri` to send commands to another Splunk server.

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
```

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
```

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

Use the CLI to administer a remote Splunk server

Use the CLI to administer a remote Splunk server

You can use the `uri` parameter with any CLI command to send that command to another Splunk server 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: Starting in 4.1.4, 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 (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
```

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 server.

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

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

Search a remote server

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 server

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
- Resurrect, unressurect

You can view all CLI commands by accessing the CLI help reference. For more information, see "Get help with the CLI" in this manual.

CLI admin commands

CLI admin commands

This topic contains information on using administrative commands via the Splunk CLI.

- For more general information, see "Get help with the CLI".
- For details on syntax for searching using the CLI, refer to "About CLI searches" in the Search Reference Manual.

```
Splunk CLI command syntax:

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

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

Commands and objects:

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

Supported commands and objects:

    [command]       [objects]

    add             [exec|forward-server|index|monitor|oneshot|
                    saved-search|search-server|tcp|udp|user]

    anonymize       source

    clean           [eventdata|globaldata|userdata|all]

    create          app

    disable         [app|deploy-client|deploy-server|discoverable|
                    dist-search|index|listen|local-index|boot-start|
                    webserver|web-ssl]

    edit            [app|exec|forward-server|index|monitor|saved-search|
                    search-server|tcp|udp|user]
```

```
   enable          [app|deploy-client|deploy-server|discoverable|
                   dist-search|index|listen|local-index|boot-start|
                   webserver|web-ssl]

   display         [app|deploy-clients|deploy-server|discoverable|
                   dist-search|index|jobs|listen|local-index|boot-start|
                   webserver|web-ssl]

   export,import   [eventdata|userdata]

   find            logs

   help            NONE

   list            [deploy-clients|exec|forward-server|index|jobs|monitor|
                   saved-search|search-server|source|sourcetype|tcp|udp|
                   user]

   login,logout    NONE

   package         app

   refresh         deploy-clients

   reload          [auth|deploy-server]

   remove          [app|exec|forward-server|jobs|monitor|saved-search|
                   search-server|source|sourcetype|tcp|udp|user]

   resurrect,unresurrect [archive_directory|index|from_time|end_time]

   search          NONE

   set             [datastore-dir|deploy-poll|default-hostname|default-index|
                   minfreemb|servername|server-type|splunkd-port|web-port]

   show            [config|datastore-dir|deploy-poll|default-hostname|
                   default-index|jobs|license|minfreemb|servername|
                   splunkd-port|web-port]

   spool           NONE

   start,stop,restart  [monitor|splunkd|splunkweb]

   status          [monitor|splunkd|splunkweb]


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

Configuration file reference

admon.conf

admon.conf

The following are the spec and example files for admon.conf.

admon.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains potential attribute/value pairs to use when configuring Windows active
# directory monitoring.
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[<stanza name>]
        * There can be multiple configuration for any given Domain Controller,
        so this is a unique name related to that  particular set of configuration.

targetDC = <string>
        * Fully qualified domain name. This can also be empty, which then it will
        obtain the local computer DC and bind to its root DN.

startingNode = <string>
        * Specify a path to the directory tree in AD where to start monitoring,
        or else if left empty it will start at the root of the directory tree

monitorSubtree = <int 0|1>
        * Given the DC path, monitor subtree instead of a single level

disabled = <in 0|1>
        * Enables or disables this particular configuration
```

admon.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains an example configuration for monitoring changes
# to the Windows active directory monitor. Refer to admon.conf.spec for details.
# The following is an example of a active directory monitor settings.
#
# To use one or more of these configurations, copy the configuration block into
# admon.conf in $SPLUNK_HOME/etc/apps/windows/local/. You must restart Splunk to enable configu
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[default]
monitorSubtree = 1
disabled = 0

[NearestDC]
```

```
targetDc =
startingNode =
```

alert_actions.conf

alert_actions.conf

The following are the spec and example files for alert_actions.conf.

alert_actions.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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 configurat
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

############################################################################
# Global options: these settings do not need to be prefaced by a stanza name
# If you do not specify an entry for each attribute, Splunk will use the default value.
############################################################################

maxresults = <int>
        * Set the global maximum number of search results sent via alerts.
        * Defaults to 100.

hostname = <string>
        * Set the hostname that is displayed in the link sent in alerts.
        * This is useful when the machine sending the alerts does not have a FQDN.
        * Defaults to current hostname (set in Splunk) or localhost (if none is set).

ttl     = <int>[p]
 * optional argument specifying the minimum ttl in seconds (or if p follows the number, the num
 * of scheduled periods) of the search artifact's if this  action is triggered.
 * If no actions are triggered, the artifacts will have their ttl determined by dispatch.ttl (i
 * 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 = <int>[mshd]
 * the maximum amount of time the execution of an action should be allowed before the action is
 * Defaults to 5m
 * Defaults to 1m for: rss

############################################################################
# EMAIL: these settings are prefaced by the [email] stanza name
############################################################################

[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 originating alert.
     * Defaults to splunk@$LOCALHOST.

subject = <string>
     * Specify an alternate email subject.
     * Defaults to SplunkAlert-<savedsearchname>.

format = <string>
     * Specify the format of text in the email.
      * Possible values:  plain, html, raw and csv.
     * This value will also apply to any attachments.

sendresults = <bool>
     * Specify whether to include the search results in the email. The results can be attached
     * Defaults to false.

inline = <true | false>
        * Specify whether the search results are contained in the body of the alert email.
        * Defaults to false.

mailserver = <string>
        * The SMTP mail server to use when sending emails.
        * Defaults to $LOCALHOST.

reportServerURL = <url>
    * The URL of the PDF report server, if one is setup and available on the network
    * For a default locally installed report server, the url would be http://localhost:8091/
    * Defaults to false

reportPaperSize = <string>
    * Default paper size for PDFs
    * Can be one of letter, legal, ledger, a2, a3, a4, a5
    * Defaults to letter

reportPaperOrientation = <string>
    * Paper orientation: portrait or landscape
    * Defaults to portrait

preprocess_results = <search-string>
    * a search string to preprocess results before emailing them. Usually the pre processing
    * consists of filtering out unwanted internal field
    * Defaults to empty string

###############################################################################
# RSS: these settings are prefaced by the [rss] stanza
###############################################################################

[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 [email] stanza).
     * Defaults to 30.
```

```
############################################################################
# script:
############################################################################
[script]
command = <string>
        * command template to be realized with information from the saved search that
        * triggered the script action.


############################################################################
# summary_index: these settings are prefaced by the [summary_index] stanza
############################################################################
[summary_index]
command = <string>
        * command template to be realized with information from the saved search that
        * triggered the summary indexing action.


############################################################################
# populate_lookup: these settings are prefaced by the [populate_lookup] stanza
############################################################################
[populate_lookup]
command = <string>
        * command template to be realized with information from the saved search that
        * triggered the populate lookup action.
```

alert_actions.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This is an example alert_actions.conf.  Use this file to configure alert actions for saved se
#
# To use one or more of these configurations, copy the configuration block into alert_actions.c
# 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://www.splunk.com/base/Documentation/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

# $name$ will be substituted from the scheduled search
subject = Splunk Alert: $name$.

format = html

reportServerURL = http://localhost:8091/

inline = false

sendresults = true
```

```
hostname = CanAccessFromTheWorld.com

command = sendemail "to=$action.email.to$" "server=$action.email.mailserver{default=localhost}$
_validate-1 = action.email.sendresults, validate( is_bool('action.email.sendresults'), "Value c


[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:

[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="$r
```

app.conf

app.conf

The following are the spec and example files for app.conf.

app.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains defines configuration for a custom app inside Splunk.
# There is no default app.conf. Instead, an app.conf is placed inside the /default
# directory of each app. For examples, see app.conf.example.
# You must restart Splunk to reload manual changes to app.conf.
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles


#
# Define how this app appears in Launcher (and online on Splunkbase)
#

[launcher]

version = <version string>
* Version numbers are a number followed by a sequence of numbers or dots.
* Pre-release versions can append a space and a single-word suffix like "beta2". Examples:
* 1.2
* 11.0.34
* 2.0 beta
* 1.3 beta2
* 1.0 b2
* 12.4 alpha
* 11.0.34.234.254
```

```
description = <string>
* short explanatory string which is displayed underneath the app's title in Launcher.
* descriptions should be 200 characters or less, becase 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, images for
# icon and screenshot should be placed in the /appserver/static dir of
# your app. They will automatically be detected by Launcher and Splunkbase.
# eg. /appserver/static/appIcon.png, (the capital "I" is required!)
# /appserver/static/screenshot.png
# app icon image must be 36px x 36px, in PNG format
# app screenshot must be 623px x 350px, in PNG format


#
# [package] defines upgrade-related metadata, and will be
# used in future versions of Splunk to streamline app upgrades.
#

[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 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


#
# Set install settings for this app
#

[install]

state = disabled | enabled
* Set whether app is disabled or enabled.
* If an app is disabled, its configs are ignored.
* By default, apps are enabled.

state_change_requires_restart = true | false
* Set whether removing/enabling/disabling app requires a restart of Splunk.
```

```
* Defaults to true.

is_configured = true | false
* whether the application's custom setup has been performed
* Defaults to false

build = integer-build-number
* must increment this whenever you change files in /appserver/static
* every release must change both "version" and "build" settings
* ensures browsers don't see cached copies of old static files in
* new versions of your app
* note that build is a single integer, unlike version which can be
* a more complex string like 1.5.18


#
# Set UI-specific settings for this app
#

[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

is_manageable = true | false
* Indicates if Splunk Manager should be used to manage this app
* Defaults to true

label = <string>
* Defines the name of the app shown in the Splunk GUI and Launcher
* Must be between 5 and 80 characters.
* Must not include "Splunk For" prefix.
* Label is required.
* Examples of good labels:
* IMAP
* SQL Server Integration Services
* FISMA Compliance

#
# Set custom configuration file settings for this app
#

[config:$STRING]
* Name your stanza.
* Preface with config:.
* Set $STRING to any arbitrary identifier.

targetconf = <$CONFIG_FILE>
* Target configuration file for changes.
* There can be only one.
* Any configuration file that is included in the application.
* For example indexes, for indexes.conf.

targetstanza = <$STANZA_NAME>
* Stanza name from application.

targetkey = <$ATTRIBUTE>
* Attribute to set.
```

```
targetkeydefault = <$VALUE>
* Default setting for attribute.
* Can be empty for no default.
```

app.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# The following are example app.conf configurations. Configure properties for your custom appli
#
# There is NO DEFAULT app.conf.
#
# 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 configuration
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[config:coldindexpath]
targetconf=indexes
targetstanza=sampledata
targetkey=coldPath
targetkeydefault=$SPLUNK_DB/sampledata/colddb
conflabel=Cold DB Path for Sample Data Index

[config:thawedindexpath]
targetconf=indexes
targetstanza=sampledata
targetkey=thawedPath
targetkeydefault=$SPLUNK_DB/sampledata/thaweddb
conflabel=Thawed DB Path for Sample Data Index

[config:homeindexpath]
targetconf=indexes
targetstanza=sampledata
targetkey=homePath
targetkeydefault=$SPLUNK_DB/sampledata/db
conflabel=Home DB Path for Sample Data Index

[launcher]
author=<author of app>
description=<textual description of app>
version=<version of app>
```

audit.conf

audit.conf

The following are the spec and example files for audit.conf.

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles


############################################################################################
# EVENT HASHING: turn on SHA256 event hashing.
############################################################################################

[eventHashing]
        * This stanza turns on event hashing -- every event is SHA256 hashed.
        * The indexer will encrypt all the signatures in a block.
        * Follow this stanza name with any number of the following attribute/value pairs.


filters=mywhitelist,myblacklist...
        * (Optional) Filter which events are hashed.
        * Specify filtername values to apply to events.
        * NOTE: The order of precedence is left to right. Two special filters are provided by d
        * blacklist_all and whitelist_all, use them to terminate the list of your filters. For
        * if your list contains only whitelists, then terminating it with blacklist_all will re
        * signing of only events that match any of the whitelists. The default implicit filter
        * terminator is whitelist_all

# FILTER SPECIFICATIONS FOR EVENT HASHING

[filterSpec:<event_whitelist | event_blacklist>:<filtername>]
        * This stanza turns on whitelisting or blacklisting for events.
        * Use filternames in "filters" entry (above).
        * For example [filterSpec:event_whitelist:foofilter].

all=<true | false>
        * The 'all' tag tells the blacklist to stop 'all' events.
        * Defaults to 'false.'

Optional list of blacklisted/whitelisted sources, hosts or sourcetypes (in order from left to r
Exact matches only, no wildcarded strings supported.
        * For example:
        source=s1,s2,s3...
        host=h1,h2,h3...
        sourcetype=st1,st2,st3...


############################################################################################
# KEYS: specify your public and private keys for encryption.
############################################################################################

[auditTrail]
        * This stanza turns on cryptographic signing for audit trail events (set in inputs.conf
```

```
        and hashed events (if event hashing is enabled above).

privateKey=/some/path/to/your/private/key/private_key.pem
publicKey=/some/path/to/your/public/key/public_key.pem
        * You must have a private key to encrypt the signatures and a public key to decrypt the
        * Set a path to your own keys
         * Generate your own keys using openssl in $SPLUNK_HOME/bin/.

queuing=<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.
        * Defaults to 'true.'
```

audit.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This is an example audit.conf.  Use this file to configure auditing and event hashing.
#
# 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://www.splunk.com/base/Documentation/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/.



# EXAMPLE #1 - hash all events:

[eventHashing]

# This performs a SHA256 hash on every event other than ones going the _audit index (which are
# handled their own way).
# NOTE: All you need to enable hashing is the presence of the stanza 'eventHashing'.



# EXAMPLE #2 - simple blacklisting

[filterSpec:event_blacklist:myblacklist]
host=somehost.splunk.com, 45.2.4.6, 45.3.5.4

[eventHashing]
filters=myblacklist

# Splunk does NOT hash any events from the hosts listed - they are 'blacklisted'. All other
```

```
# events are hashed.



# EXAMPLE #3 - multiple blacklisting

[filterSpec:event_blacklist:myblacklist]
host=somehost.splunk.com, 46.45.32.1
source=/some/source
sourcetype=syslog, apache.error

[eventHashing]
filters=myblacklist

# DO NOT hash all events with the following, sources, sourcetypes and hosts - they are all
# blacklisted.  All other events are hashed.



# EXAMPLE #4 - whitelisting

[filterspec:event_whitelist:mywhitelist]
sourcetype=syslog
#source=aa, bb  (these can be added as well)
#host=xx, yy

[filterspec:event_blacklist:nothingelse]
#The 'all' tag is a special boolean (defaults to false) that says match *all* events
all=True

[eventSigning]
filters=mywhitelist, nothingelse

# Hash ONLY those events which are of sourcetype 'syslog'.  All other events are NOT hashed.
# Note that you can have a list of filters and they are executed from left to right for every e
# If an event passed a whitelist, the rest of the filters do not execute.  Thus placing
# the whitelist filter before the 'all' blacklist filter says "only hash those events which
# match the whitelist".
```

authentication.conf

authentication.conf

The following are the spec and example files for authentication.conf.

authentication.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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 configura
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[authentication]
    * Follow this stanza name with any number of the following attribute/value pairs.

authType = <string>
    * Specify which authentication system to use.
    * Currently available: Splunk, LDAP, Scripted.
    * Defaults to Splunk.

authSettings = <string>
    * Key to look up the specific configurations of chosen authentication system.
    * <string> is the name of the stanza header [<authSettingsKey>].
    * This is used by LDAP and Scripted Authentication.

####################
# LDAP settings
####################

[<authSettings-key>]
    * Follow this stanza name with the following attribute/value pairs.

host = <string>
    * REQUIRED - Hostname of LDAP server.
    * Be sure that your Splunk server can resolve the host name.

SSLEnabled = <boolean>
    * OPTIONAL - 0 for disabled (default)
    * 1 for enabled.
        * See the file $SPLUNK_HOME/etc/openldap/openldap.conf for SSL LDAP settings

port = <integer>
    * OPTIONAL - 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>
    * Distinguished name of the user that will be retrieving the LDAP entries
    * This user needs to have read access to all LDAP users and groups you wish to use in Splun
    * Optional, but usually required due to LDAP security settings.
        * Leave this blank if your LDAP entries can be retrieved with anonymous bind

bindDNpassword = <string>
    * Password for the bindDN user.
    * Optional - leave this blank if anonymous bind is sufficient

userBaseDN = <string>
    * REQUIRED - Distinguished names of LDAP entries whose subtrees contain the users
    * Enter a ';' delimited list to search multiple trees.

userBaseFilter = <string>
    * OPTIONAL - 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.
```

```
groupBaseDN = <string>
    * REQUIRED - 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 t
        * Set groupBaseDN to the same as userBaseDN, which means you will search for groups in
        * Next, set the groupMemberAttribute and groupMappingAttribute to the same attribute as
            * This means the entry, when treated as a group, will use the username value as its
        * For clarity, you should probably also set groupNameAttribute to the same as userNameA

groupBaseFilter = <string>
    * OPTIONAL - The LDAP search filter you wish to use when searching for groups
    * Like userBaseFilter, this is highly recommended to speed up LDAP queries
    * See RFC 2254 for more information
    * This defaults to no filtering

userNameAttribute = <string>
    * REQUIRED - User entry attribute whose value is the username
    * NOTE: This attribute should use case insensitive matching for its values, and the values
        * Users are case insensitive in Splunk
    * In Active Directory, this is 'sAMAccountName'
    * A typical attribute for this is 'uid'

realNameAttribute = <string>
    * REQUIRED - User entry attribute whose value is their real name (human readable)
    * A typical attribute for this is 'cn'

groupMappingAttribute  = <string>
    * OPTIONAL - User entry attribute whose value is used by group entries to declare membershi
    * Groups are often mapped with user DN, so this defaults to 'dn'
    * Set this if groups are mapped using a different attribue
        * 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

groupNameAttribute = <string>
    * REQUIRED - 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 en

groupMemberAttribute = <string>
    * REQUIRED - 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
        * To declare 'splunkuser' as a group member, its attribute 'member' must have the value

charset = <string>
    * OPTIONAL - ONLY set this for an LDAP setup that returns non-UTF-8 encoded data. LDAP is s
    * Follows the same format as CHARSET in props.conf (see props.conf.spec)
    * An example value would be "latin-1"

######################
# Map roles
######################

[roleMap]
    * Follow this stanza name with several Role to Group mappings as defined below.

<RoleName> = <LDAP group string>
```

```
    * Maps a Splunk role (from authorize.conf) to LDAP groups
    * This list is semi-colon delimited (no spaces).
    * List several of these attribute value pairs to map all Splunk roles to Groups

####################
# Scripted authentication
####################

[<authSettings-key>]
        * Follow this stanza name with the following attribute/value pairs:

scriptPath = <string>
        * REQUIRED - Full path to the script, including the path to the program that runs it (p
        * ex: "$SPLUNK_HOME/bin/python" "$SPLUNK_HOME/etc/system/bin/$MY_SCRIPT"
        * Note that if a path contains spaces, it must be quoted. Our example above handles the

scriptSearchFilters = <boolean>
        * OPTIONAL - Only set this to 1 to call the script to add search filters.
        * 0 disables (default)

# Cache timing:
# Use these settings to adjust the maximum frequency at which Splunk calls your script function
# Caching is disabled by default

# 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.
[cacheTiming]
getUserInfoTTL = <time range string>
    * Timeout for getUserInfo.

getUsersTTL = <time range string>
      * Timeout for getUsers.

userLoginTTL = <time range string>
      * Timeout for userLogin.
```

authentication.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This is an example authentication.conf.  Use this file to configure LDAP or toggle between LD
# and Splunk's native authentication system.
#
# To use one or more of these configurations, copy the configuration block into authentication.
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

# Use Splunk's built-in authentication:
[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

# This stanza maps roles you have created in authorize.conf to LDAP Groups
[roleMap]
admin = SplunkAdmins


#### 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 = (objectclass=*)
groupMappingAttribute = dn
groupMemberAttribute = uniqueMember
groupNameAttribute = cn
host = ldapbogus.splunksupport.com
port = 389
realNameAttribute = givenName
userBaseDN = ou=People,dc=splunksupport,dc=com
userBaseFilter = (objectclass=*)
userNameAttribute = uid

[roleMap]
admin = SplunkAdmins
power = SplunkPowerUsers
user = SplunkUsers

#### Sample Configuration for Active Directory
```

```
[authentication]
authSettings = AD
authType = LDAP

[AD]
SSLEnabled = 0
bindDN = ldap_bind@splunksupport.kom
bindDNpassword = ldap_bind_user_password
groupBaseDN = CN=Groups,DC=splunksupport,DC=kom
groupBaseFilter = (objectclass=*)
groupMappingAttribute = dn
groupMemberAttribute = member
groupNameAttribute = cn
host = ADbogus.splunksupport.kom
port = 389
realNameAttribute = cn
userBaseDN = CN=Users,DC=splunksupport,DC=kom
userBaseFilter = (objectclass=*)
userNameAttribute = sAMAccountName

[roleMap]
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 = (objectclass=*)
groupNameAttribute = cn
SSLEnabled = 0
port = 389
userBaseDN = cn=users,dc=osx,dc=company,dc=com
host = hostname_OR_IP
userBaseFilter = (objectclass=*)
userNameAttribute = uid
groupMappingAttribute = uid
groupBaseDN = dc=osx,dc=company,dc=com
groupMemberAttribute = memberUid
realNameAttribute = cn
failsafeLogin = splunk_failsafe
failsafePassword = you_specify_the_password

[roleMap]
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/radiusScrip

# 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

# Cache results for different times per function
[cacheTiming]
userLoginTTL    = 30s
getUserInfoTTL  = 1min
getUsersTTL     = 5mins
```

authorize.conf

authorize.conf

The following are the spec and example files for authorize.conf.

authorize.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[capability::<capability>]
        * Defines a Splunk system capability
        * Splunk adds all of its capabilities this way
        * For the default list of capabilities and assignments, see authorize.conf under the 'd
        * Descriptions of specific capabilities are listed below

[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
        * Roles inherit all capabilities from imported roles, and inherited capabilities cannot

importRoles = <string>
        * Semicolon delimited list of other role capabilities that should be imported.
        * Importing other roles also imports the other aspects of that role, such as allowed in

srchFilter = <string>
        * Semicolon delimited list of search filters for this Role.

srchTimeWin = <number>
        * Maximum time span of a search, in seconds.

srchDiskQuota = <number>
        * Maximum amount of disk space (MB) that can be taken by search jobs of a user
          that belongs to this role

srchJobsQuota = <number>
        * Maximum number of concurrently running historical searches a member of this role can

rtSrchJobsQuota = <number>
        * Maximum number of concurrently running real-time searches a member of this role can h

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
        * To match internal indexes, start with '_'. All internal indexes are represented by '_

srchIndexesAllowed = <string>
        * Semicolon delimited list of indexes this role is allowed to search
        * Follows the same wildcarding semantics as srchIndexesDefault

### Descriptions of Splunk system capabilities
[capability::admin_all_objects]
        * A role with this capability has access to objects in the system (user objects, search
        * 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]
        * Required to change authentication settings through the various authentication endpoin
        * Also controls whether authentication can be reloaded

[capability::change_own_password]
        * Self explanatory. Some auth systems prefer to have passwords be immutable for some us

[capability::delete_by_keyword]
        * Required to use the 'delete' search operator. Note that this does not actually delete
        * Delete merely masks the data (via the index) from showing up in search results.

[capability::edit_deployment_client]
        * Self explanatory. The deployment client admin endpoint requires this cap for edit.

[capability::edit_deployment_server]
        * Self explanatory. The deployment server admin endpoint requires this cap for edit.

[capability::edit_dist_peer]
        * Required to add and edit peers for distributed search.

[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]
        * Required to edit and end user sessions through the httpauth-tokens endpoint

[capability::edit_input_defaults]
        * Required to change the default hostname for input data in the server settings endpoin

[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_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_scripted]
        * Required to create and edit scripted inputs.

[capability::edit_search_server]
        * Required to edit general distributed search settings like timeouts, heartbeats, and h

[capability::edit_server]
        * Required to edit general server settings such as the server name, log levels, etc.

[capability::edit_splunktcp]
        * Required to change settings for receiving TCP input from another Splunk instance.

[capability::edit_splunktcp_ssl]
        * Required to list or edit any SSL specific settings for Splunk TCP input.

[capability::edit_tcp]
        * Required to change settings for receiving general TCP inputs.

[capability::edit_udp]
        * Required to change settings for UDP inputs.

[capability::edit_user]
        * Required to create, edit, or remove users.
        * Note that Splunk users may edit certain aspects of their information without this cap
        * Also required to manage certificates for distributed search.

[capability::edit_web_settings]
        * Required to change the settings for web.conf through the system settings endpoint.

[capability::get_metadata]
        * Required to use the 'metadata' search processor.

[capability::get_typeahead]
        * Required for typeahead. This includes the typeahead endpoint and the 'typeahead' sear

[capability::indexes_edit]
        * Required to change any index settings like file size and memory limits.

[capability::license_tab]
        * Required to access and change the license.
```

```
[capability::list_forwarders]
        * Required to show settings for forwarding data.
        * Used by TCP and Syslog output admin handlers.

[capability::list_httpauths]
        * Required to list user sessions through the httpauth-tokens endpoint.

[capability::list_inputs]
        * Required to view the list of various inputs.
        * This includes input from files, TCP, UDP, Scripts, etc.

[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]
        * Required to edit settings for entries and categories in the python remote apps handle
        * See restmap.conf for more information

[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]
        * Required to get information from the services/properties endpoint.

[capability::rest_properties_set]
        * Required to edit the services/properties endpoint.

[capability::restart_splunkd]
        * Required to restart Splunk through the server control handler.

[capability::rtsearch]
        * Required to run a realtime search.

[capability::schedule_search]
        * Required to schedule saved searches.

[capability::search]
        * Self explanatory - required to run a search.

[capability::use_file_operator]
        * Required to use the 'file' search operator.
```

authorize.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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 configur
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/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
# 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
# ninja is allowed to take up 500 megabytes total on disk for all their jobs.
```

commands.conf

commands.conf

The following are the spec and example files for commands.conf.

commands.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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/searchs
# or $SPLUNK_HOME/apps/MY_APP/bin/.  For the latter, put a custom commands.conf in
# $SPLUNK_HOME/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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles


[$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 defa

type = <string>
        * Type of script: python, perl
        * Defaults to python.

filename = <string>
        * Name of script file for command.
        * <stanza-name>.pl for perl.
        * <stanza-name>.py for python.

streaming = <true/false>
        * Is the command streamable.
        * Defaults to false.
```

```
maxinputs = <integer>
        * Maximum number of events that can be passed to the command for each invocation.
        * 0 for no limit.
        * Defaults to 50000.

passauth = <true/false>
        * If set to true, passes an authentication token on the start of input.
        * Defaults to false.

run_in_preview = <true/false>
        * Run this command if we 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>
        * Does command retain events?
                * e.g. sort/dedup/cluster.
        * Or does it transform them
                e.g. stats.
        * Defaults to false.

generating = <true/false>
        * Does your command generate new events
                * eg if the 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 f
        * Defaults to false.

overrides_timeorder = <true/false>
        * If generating = false, does command change the order of events with respect to time
        * Defaults to true.

requires_preop
        * require pre-streaming operations
        * Defaults to false.

streaming_preop = <string>
        * A string that denotes the requested pre-streaming search string.

required_fields = <string>
        * A comma separated list of fields required by this command
        * Defaults to '*'

supports_multivalues = <true/false>
        * Does the command support 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
```

```
          (i.e., isinstance(val, basestring) == True).

supports_getinfo = <true/false>
        * Does the command support dynamic probing for settings via the first argument,
          being invoked == __GETINFO__ or __EXECUTE__.
```

commands.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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-n

# 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

[iplocation]
FILENAME = iplocation.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

crawl.conf

The following are the spec and example files for crawl.conf.

crawl.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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://www.splunk.com/base/Documentation/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 us


[default]

[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, shar

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,

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 fi
        * For example: days_sizek_pairs_list = 7-0, 30-1000 tells Splunk to crawl only files la
        modified within 7 days and at least 0kb in size, or modified within the last 30 days ar
        * 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_ba
        * Defaults to 100.




[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

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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 configuration
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[files]
bad_directories_list= bin, sbin, boot, mnt, proc, tmp, temp, home, mail, .thumbnails, cache, ol
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
```

default.meta.conf

default.meta.conf

The following are the spec and example files for default.meta.conf.

default.meta.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# *.meta files contain ownership information, access controls, and export
# settings for Splunk objects like saved searches, event types, and views.

# Set access controls on the app containing this metadata file.
[]
# 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.
access = read : [ * ], write : [ admin, power ]

# Set access controls on this app's views.
[views]
# Allow all users to read this app's views. Allow only admin users to create,
# remove, share, or unshare views in this app.
```

```
access = read : [ * ], write : [ admin ]

# Set access controls on a specific view in this app.
[views/index_status]
# Allow only admin users to read or modify this view.
access = read : [ admin ], write : [ admin ]
# Make this view available in all apps.
export = system
## To make this view available only in this app, set 'export = none' instead.
# Set admin as the owner of this view.
owner = admin
```

default.meta.conf.example

```
No example
```

deploymentclient.conf

deploymentclient.conf

The following are the spec and example files for deploymentclient.conf.

deploymentclient.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains possible attributes and values for configuring a deployment client to rece
# content (apps and configurations) from a deployment server.
#
# To customize the way a deployment client behaves, place a deploymentclient.conf in $SPLUNK_HO
# on that Splunk instance. Configure what apps/configuration content is deployed to a given dep
# serverclass.conf. Refer to serverclass.conf.spec and serverclass.conf.example for more inform
#
# 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://www.splunk.com/base/Documentation/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.
#*******************************************************************************

[deployment-client]

disabled = <false or true>
    * Defaults to false
    * Enable/Disable deployment client.

clientName=deploymentClient
    * also referred to as 'tag'. Defaults to 'deploymentClient'.
    * a name that can be used to filter on by deployment server, and takes precedence over dns
```

```
workingDir=$SPLUNK_HOME/var/run/deploy-client
    * temporary folder used by the deploymentClient to download apps and configuration content.

repositoryLocation = $SPLUNK_HOME/etc/apps
    * The location into which content is installed after being downloaded from a deployment ser
    * Apps and configuration content must be installed into the default location ($SPLUNK_HOME/
    * Note: Apps and configuration content to be deployed may be located in an alternate locati
    * The deployment client will use the 'serverRepositoryLocationPolicy' defined below to dete

serverRepositoryLocationPolicy = < one of acceptSplunkHome, acceptAlways, rejectAlways>
    * defaults to 'acceptSplunkHome'.
    * acceptSplunkHome - accept the repositoryLocation supplied by the deployment server, only
    * acceptAlways - always accept the repositoryLocation supplied by the deployment server.
    * rejectAlways - reject the server supplied value and only use the repositoryLocation speci

endpoint=$deploymentServerUri$/services/streams/deployment?name=$serverClassName$:$appName$
    * The HTTP endpoint from which content should be downloaded.
    * Note: The deployment server may specify a different endpoint from which to download each
    * The deployment client will use the 'serverEndpointPolicy' defined below to determine whic
    * $deploymentServerUri$ will resolve to 'targetUri' defined in the [target-broker] stanza b
    * $serverClassName$ and $appName$ mean what they say.

serverEndpointPolicy = < one of 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' defi

phoneHomeIntervalInSecs = <N>
    * defaults to 30
    * This determines how frequently this deployment client should check for new content.

# Advanced!
# You should use this property only when you have a hierarchical DS installation, and have a Sp
reloadDSOnAppInstall = <false or true>
    * Defaults to false
    * Setting this flag to true will cause the DeploymentServer on this Splunk to be reloaded,

# The following stanza specifies deployment server connection information

[target-broker:deploymentServer]

targetUri= <deploymentServer>:<mgmtPort>
    * URI of the deployment server.
```

deploymentclient.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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
# There is nothing in [deployment-client] because the deployment client is not overriding the v
# 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
# will not be recognized and run.
# This configuration rejects any location specified by the deployment server and replaces it wi
# 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 specifie
# 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
# 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
```

distsearch.conf

distsearch.conf

The following are the spec and example files for distsearch.conf.

distsearch.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains possible attributes and values you can use to configure distributed search
#
```

```
# There is NO DEFAULT distsearch.conf.
#
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[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 and on.
* Defaults to false (your distributed search stanza is enabled by default).

heartbeatFrequency = <in seconds>
* Heartbeat in seconds.
* 0 disables all heartbeats.
* If the heartbeat is disabled, no other Splunk server is able to auto-discover this instance.
* Defaults to 0.

heartbeatMcastAddr = <IP address>
* Set a multicast address.
* Defaults to 255.0.0.37.

heartbeatPort = <port>
* Set heartbeat port.
* Defaults to 60.

serverTimeout = <in seconds>
* How long to wait for a connection to a server.
* If a connection occurs, a search times out in 10x this value.
        * For example, if set to 10 seconds, the maximum search allowed is 100 seconds.
* This setting works in tandem with 'removeTimedOutPeers.'
* Defaults to 10.

statusTimeout = <in seconds>
* Set how long to wait for a server to return its status.
* Up this number if your peered servers are slow or if the server name disappears from Splunk W

removedTimedOutServers = true | false
* If true, remove a server connection that cannot be made within 'serverTimeout.'
* If false, every call to that server attempts to connect.
        * NOTE: This may result in a slow user interface.

checkTimedOutServersFrequency = <in seconds>
* This tag is ONLY relevant if 'removeTimedOutServers' is set to true.
        * If 'removeTimedOutServers' is false, this attribute is ignored.
* Rechecks servers at this frequency (in seconds).
* If this is set to 0, then no recheck will occur.
* Defaults to 60.

autoAddServers = true | false
* If this tag is set to 'true', this node will automatically add all discovered servers.
* Defaults to false.

skipOurselves = true | false
```

```
* If this is set to 'true', then this server will NOT participate as a server in any search or
* This is used for building a node that does nothing but merge the results from other servers.
* Defaults to false.

ttl = <integer>
* Time to live.
* Increasing this number allows the UDP multicast packets to spread beyond the current subnet t
* NOTE:  This only will work if routers along the way are configured to pass UDP multicast pack
* Defaults to 1 (this subnet).

servers = <comma separated list of servers)
* Initial list of servers.
* If operating completely in 'autoAddServers' mode (discovering all servers), there is no need

blacklistNames = <comma separated list of server names>
* List of server names that you do not want to peer with.
* Server names are the 'server name' that is created for you at startup time.

blacklistURLs = <comma separated list of server names or URIs>
* Specify servers to blacklist.
* You can blacklist on server name (above) or server URI (x.x.x.x:port).

shareBundles = true | false
* Indicates if this server will share its app with any of its peers.
* This flag is required on the search distributor.
* Defaults to true.

connectionTimeout = <integer>
* Amount of time in seconds to use as a timeout during search peer connection establishment

sendTimeout = <integer>
* Amount of time in seconds to use as a timeout while trying to write/send data to a search pee

receiveTimeout = <integer>
* Amount of time in seconds to use as a timeout while trying to read/receive data from a search


#*******************************************************************************
# REPLICATION SETTING OPTIONS
# These options must be set under an [replicationSettings] entry
#*******************************************************************************

connectionTimeout = <number>
* The maximum number of seconds to wait before timing out on inital connection to a peer

sendRcvTimeout = <number>
* The maximum number of seconds to wait for the sending of a full replication to a peer

replicationThreads = <number>
* The maximum number of threads to use when performing bundle replication to peers
* Defaults to 1

#*******************************************************************************
# REPLICATION WHITELIST OPTIONS
# These options must be set under an [replicationWhitelist] entry
#*******************************************************************************
<name> = <whitelist_regex>
* A pattern that if it matches a candidate file for replication (ie is under $SPLUNK_HOME/etc )
* Note: Wildcards and replication:
```

```
* You can use wildcards to specify your path for replicated files. Use ... for paths and * for
* ... recurses through directories until the match is met. This means that /foo/.../bar will ma
* To recurse through a subdirectory, use another .... For example /foo/.../bar/....
* matches anything in that specific path segment. It cannot be used inside of a directory path;
* Combine * and ... for more specific matches:
* foo/.../bar/* matches any file in the bar directory within the specified path.

#*******************************************************************************
# REPLICATION BLACKLIST OPTIONS
# These options must be set under an [replicationBlacklist] entry
#*******************************************************************************
<name> = <blacklist_regex>
* All comments from the replication whitelist notes above apply here.
* replication blacklist takes precedence over the whitelist, meaning that a file that matches B
```

distsearch.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This is an example 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[distributedSearch]
heartbeatFrequency = 10
servers = 192.168.1.1:8059,192.168.1.2:8059
blacklistNames = the-others,them
blacklistURLs = 192.168.1.3:8059,192.168.1.4:8059

# This entry distributes searches to 192.168.1.1:8059,192.168.1.2:8059.
# The server sends a heartbeat every 10 seconds.
# There are four blacklisted instances, listed across blacklistNames and blacklistURLs.
# 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
```

eventdiscoverer.conf

eventdiscoverer.conf

The following are the spec and example files for eventdiscoverer.conf.

eventdiscoverer.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5

# This file contains possible attributes and values you can use to configure event discovery th
# the search command "typelearner."
#
# There is an eventdiscoverer.conf in $SPLUNK_HOME/etc/system/default/.  To set custom configur
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

ignored_keywords = <comma-separated list of terms>
        * Terms in this list are never considered for defining an event type.
        * If you find that eventtypes have terms you do not want considered (e.g., "mylaptopnam
        that term to this list.
        * Default = "sun, mon, tue,..." (see $SPLUNK_HOME/etc/system/default/eventdiscover.conf

ignored_fields = <comma-separated list of fields>
        * Similar to ignored_keywords, except fields as defined in Splunk.
        * Defaults include time-related fields that would not be useful for defining an event t

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,..." (see $SPLUNK_HOME/etc/system/default/eventdis
```

eventdiscoverer.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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
# 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://www.splunk.com/base/Documentation/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

event_renderers.conf

The following are the spec and example files for event_renderers.conf.

event_renderers.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains possible attribute/value pairs for configuring event rendering properties.
#
# There is an event_renderers.conf in $SPLUNK_HOME/etc/system/default/.  To set custom configur
# place an event_renderers.conf in $SPLUNK_HOME/etc/system/local/, or your own custom app direc
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[<name>]
* Stanza name.
* Unique.

eventtype = <eventtype>
* Specify eventtype 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 woul
element of the event having an html attribute class with a value of splEvent-foo (eg., class="s
externalize your css style rules for this in $APP/appserver/static/application.css. So for exam
would add to application.css:
.splEvent-foo { color:red; }
```

event_renderers.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
# 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/defa
# into ../local and edit there.
#
# This file contains mappings between Splunk eventtypes and 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

eventtypes.conf

The following are the spec and example files for eventtypes.conf.

eventtypes.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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 wri
# to $SPLUNK_HOME/etc/systems/local/eventtypes.conf.
#
# There is an eventtypes.conf in $SPLUNK_HOME/etc/system/default/.  To set custom configuration
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles


[$EVENTTYPE]
        * Header for the event type
        * $EVENTTYPE is the name of your event type.
        * You can have any number of event types, each represented by a stanza and any number
        of the following attribute/value pairs.
              * NOTE: If the name of the event type includes field names surrounded by the perc
              character (e.g. "%$FIELD%") then the value of $FIELD is substituted into the even
          name for that event.  For example, an event type with the header [cisco-%code%] that
          "code=432" becomes labeled "cisco-432".

disabled = <1 or 0>
        * Toggle event type on or off.
        * Set to 0 to disable.

search = <string>
        * Search terms for this event type.
        * For example: error OR warn.
```

eventtypes.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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://www.splunk.com/base/Documentation/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

fields.conf

The following are the spec and example files for fields.conf.

fields.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains possible attribute and value pairs for creating dynamic field extractions.
# This file contains possible attribute and value pairs for configuring additional information
# Use this file if you are creating a field at index time (not advised).
# Also, use this file to indicate that your configured field's value is a smaller part of a tok
# For example, your field's value is "123" but it occurs as "foo123" in your event.
# Configure fields.conf to:
#   * Tell Splunk how to handle multi-value fields.
#   * Distinguish indexed and extracted fields.
#   * Improve search performance by telling the search processor how to handle field values.
#
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[<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 = <regular expression>
* A regular expression that indicates how the field can take on multiple values at the same tim
 * Use this setting to configure multi-value fields (http://www.splunk.com/doc/current/admin/Mu
* 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 search/where (the search command), the summary and XML outputs of the
* Tokenization of indexed fields (INDEXED = true) is not supported thus this attribute is ignor
* 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
* Set {{indexed_value}} to true if the value is in the raw text of the event.
* Set it 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 (s
* Defaults to true.
* NOTE: You only need to set indexed_value if indexed = false.
```

fields.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains an example fields.conf.  Use this file to configure dynamic field extracti
#
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles
#
# The following example can be used with the IMAP bundle (found here:
# http://www.splunkbase.com/addons/All/Technologies/Mail/addon:IMAP+Addon .)
# 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

indexes.conf

The following are the spec and example files for indexes.conf.

indexes.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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://www.splunk.com/base/Documentation/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.
#
# DO NOT change the attribute QueryLanguageDefinition without consulting technical support.

#*******************************************************************************
# GLOBAL OPTIONS
# These options affect every index
#*******************************************************************************

sync = <integer>
* The index processor syncs events every <integer> number of events.
* Must be non-negative.
* Set to 0 to disable.
* Defaults to 0.

defaultDatabase = <database name>
* If no index is specified during search, Splunk searches default database.
* Also the database displays by default on the homepage.
* Defaults to main.

queryLanguageDefinition = <path to file>
* The path to the search language definition file.
* DO NOT EDIT THIS SETTING.
* Defaults to $SPLUNK_HOME/etc/searchLanguage.xml.

blockSignatureDatabase = <database name>
* This is the database that stores block signatures of events.
* Defaults to _blocksignature.

memPoolMB = <number or "auto">
* Specifying "auto" or an invalid value will cause Splunk to autotune this param based on your
* How much memory is given to indexer memory pool that restricts number of outstanding events i
* Has to be greater than 0 and has a maximum of 1048576 (which corresponds to 1 TB)
* Setting this too high may lead to splunkd memory usage going up substantially
* Setting this too low may degrade splunkd indexing performance
* Please only set this value if you are an expert user or are advised by Splunk Support
```

```
* CARELESSNESS IN SETTING THIS MAY LEAD TO PERMANENT BRAIN DAMAGE OR LOSS OF JOB

indexThreads = <number or "auto">
* Specifying "auto" or an invalid value will cause Splunk to autotune this param based on your
* The number of threads to use for indexing
* The number has to be at least 1 and at most 16
* If specified as < 0 or more than 16, it will be autotuned
* This number should not be set higher than the number of processors in the box.
* If splunkd is also doing parsing and aggregation, the number should be lower than the total n
* Please only set this value if you are an expert user or are advised by Splunk Support
* CARELESSNESS IN SETTING THIS MAY LEAD TO PERMANENT BRAIN DAMAGE OR LOSS OF JOB

assureUTF8 = true | false
* Verifies that all data retreived from the index is proper UTF8.
* Will degrade indexing performance when enabled (set to true)
* Can only be enabled/disabled globally by use on the [default] stanza.
* Defaults to false

enableRealtimeSearch = true | false
* Enables real time searches
* Defaults to true

#*******************************************************************************
# PER INDEX OPTIONS
# These options may be set under an [$INDEX] entry
#*******************************************************************************

disabled = true | false
* Toggle your index entry off and on.
* Set to true to disble an index.
* Defaults to false.

homePath = <path on server>
* The path that contains the hot and warm databases and fields for the index.
* Splunkd keeps a file handle open for warm databases at all times .
* CAUTION: Path MUST be writable.

coldPath = <path on server>
* The path that contains the cold databases for the index.
* Cold databases are opened as needed when searching.
* CAUTION: Path MUST be writable.

thawedPath = <path on server>
* The path that contains the thawed (resurrected) databases for the index.

# The following options can be set either per index or at the top of the file as defaults for a
# Defaults set at the top of the file are overridden if set on a per-index basis.

maxWarmDBCount = <integer>
* The maximum number of warm DB_N_N_N directories.
* All warm DBs are in the <homePath> for the index.
* Warm DBs are kept in open state.
* Defaults to 300.

maxTotalDataSizeMB = <integer>
* The maximum size of an index (in MB).
* If an index grows larger, the oldest data is frozen.
* Defaults to 500000.
```

```
rotatePeriodInSecs = <integer>
* Frequency (in seconds) to check if a new hot DB needs to be created.
* Also the frequency to check if there are any cold DBs that need to be frozen.
* Defaults to 60.

frozenTimePeriodInSecs = <integer>
* Number of seconds after which indexed data rolls to frozen.
* If you do not specify a coldToFrozenScript, this data is erased.
* IMPORTANT: Every event in the DB must be older than frozenTimePeriodInSecs before it will rol
* frozenTimePeriodInSecs will be frozen the next time splunkd checks.
* Defaults to 188697600.

warmToColdScript = <script>
* Specifies a script to run when moving data from warm to cold.
* Should not be necessary for any purpose in Splunk 4.  Migrating data across filessystems is h
* If you specify a script here, the script becomes responsible for moving the data, and splunk'
* The script must accept two arguments:
 * First: the warm directory (bucket) to be rolled to cold.
 * Second: the destination in the cold path.
* Splunk searches and other activities will be paused while the script works.
* Please contact Splunk Support (http://www.splunk.com/page/submit_issue) if you need help conf
* Defaults to empty.

coldToFrozenScript = <script>
* Specify an archiving script by changing <script>.
* Splunk ships with two default archiving scripts (or create your own):
 * compressedExport.sh - Export with tsidx files compressed as gz.
 * flatfileExport.sh - Export each source as a flat text file.
 * <$script> path is relative to $SPLUNK_HOME/bin, and must be in that directory or a subdirect
* WINDOWS users use this notation:
  coldToFrozenScript = <script> "$DIR"
 * <script> can be either compressedExport.bat or flatfileExport.bat
 * flatfileExport (bat or sh) is not currently recommended for performance and resource issues.

compressRawdata = true | false
* This param is ignored.  Splunkd will always compress raw data now

maxConcurrentOptimizes = <integer>
* The number of concurrent optimize processes that can be run against the hot DB.
* This number should be increased if:
 1. There are always many small tsidx files in the hot DB.
 2. After rolling, there are many tsidx files in warm or cold DB.

maxDataSize = <integer, "auto", or "auto_high_volume">
* The maximum size in MBs for a hot db to grow before a roll to warm is triggered
* Specifying "auto" or "auto_high_volume" will cause Splunk to autotune this param based on you
* 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.
* "auto" sets the size to 750MB, "auto_high_volume" to 10GB
* Although the maximum value you can set this is 1048576 MB, which corresponds to 1 TB, a reaso
 * any number outside this range should be approved by Splunk support before proceeding
* If you specify an invalid number or string for maxDataSize, maxDataSize will be auto tuned
* NOTE: The precise size of your warm buckets may vary from maxDataSize due to post processessi

rawFileSizeBytes = <positive integer>
* Target uncompressed size in bytes for individual raw file in the rawdata directory of the inc
* Defaults to 10485760 (=10*1024*1024)
* 0 is NOT a valid value, if 0 is specified, rawFileSizeBytes will be set to default value
```

```
* NOTE: rawFileSizeBytes only specifies a target file size, the file size may be slightly large
* WARNING: This is an advanced parameter, please only tune if you are instructed to do so by Sp

maxMemMB = <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 what is available.
* The default is recommended for all environments.
* Defaults to 5.

blockSignSize = <integer>
* Controls how many events make up a block for block signatures.
* If it is set to 0 block signing is disabled for this index.
* Defaults to 0.
* A recommended value of this variable is 100.

maxHotSpanSecs = <non-negative number>
   * Upper bound of target max timespan of hot/warm buckets in seconds
   * Defaults to 90 days
   * NOTE: if you set this too small, you may get an explosion of hot/warm
     buckets in the filesystem.  The system sets a lower bound implicitly for
     this parameter at 3600, but this is an advanced parameter that should be set
     with care and understanding of the characteristics of your data

maxHotIdleSecs = <non-negative number>
   * If the number of hot buckets is growing and is exceeding the maxHotBuckets
     count, this parameter will take effect, otherwise, it has no effect
   * Upper bound of life in seconds of a hot bucket
   * Once this time expires, hot bucket will be "rolled" into a warm bucket
   * A value of 0 turns of the Idle check (equivalent to INFINITE idle time)
   * Defaults to 0

maxHotBuckets = <non-negative number>
   * Maximum hot buckets that can exist per index
   * LRU policy will be used to age out hot buckets when this number is exceeded
   * Defaults to 1

quarantinePastSecs = <non-negative number>
   * events with timestamp of quarantinePastSecs older than "now" will be
     dropped into quarantine bucket
   * defaults to 77760000 (300 days)
   * this is a mechanism to prevent main hot buckets from being poluted with
     fringe events

quarantineFutureSecs = <non-negative number>
   * events with timestamp of quarantineFutureSecs newer than "now" will be
     dropped into quarantine bucket
   * defaults to 2592000 (1 month)
   * this is a mechanism to prevent main hot buckets from being poluted with
     fringe events

syncMeta = <true/false>
   * when specified, a sync operation is called before file descriptor is closed on metadata fi
   * this functionality was introduce to improve integrity of metadata files, especially in req
   * defaults to true
   * NOTE: This is an advanced parameter that should be left alone unless instructed by a Splun
```

```
isReadOnly = true | false
* Set to true to make an index read-only.
* If true, no new events can be added to the index, but it is still searchable.
```

indexes.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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://www.splunk.com/base/Documentation/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 = /opt/bin/compressedExport.sh
```

inputs.conf

inputs.conf

The following are the spec and example files for inputs.conf.

inputs.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5

# This file contains possible attributes and values you can use to configure inputs,
# distributed inputs 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.exampl
# You must restart Splunk to enable new configurations.
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles
#

#*******
# GENERAL SETTINGS:
# The following attributes/value pairs are valid for ALL input types (except file system change
# You must first enter a stanza header, specifying the input type.
# Then, use any of the following attribute/value pairs.
#*******

host = <string>
* Set the default host to a static value.
* "host=" is automatically prepended to <string>.
* Defaults to the IP address of fully qualified domain name of the host where the data originat

index = <string>
* Set the index to store events from this input.
* "index=" is automatically prepended to <string>.
* Defaults to "index=main" (or whatever you have set as your default index).

source = <string>
* Set the source for events from this input.
* "source=" is automatically prepended to <string>.
* Defaults to the file path.

sourcetype = <string>
* Set the sourcetype for events from this input.
* "sourcetype=" is automatically prepended to <string>.
* Splunk automatically picks a source type based on various aspects of your data.  There is no

queue = parsingQueue | indexQueue
* Specify where the input processor should deposit the events it reads.
* Set to "parsingQueue" to apply props.conf and other parsing rules to your data.
* Set to "indexQueue" to send your data directly into the index.
* Defaults to parsingQueue.

_TCP_ROUTING = <tcpout_group_name>,<tcpout_group_name>,<tcpout_group_name>
* comma separated list of tcpout group names
* Using this you can selectively forward your data to specific indexer(s).
* Specify the tcpout group forwarder should use when forwarding your data.
```

```
  The tcpout group names are defined in outputs.conf with [tcpout:<tcpout_group_name>]
* Defaults to groups present in 'defaultGroup' in tcpout stanza in outputs.conf.

#*******
# Valid input types follow, with input-specific attributes listed as well:
#*******


#*******
# MONITOR:
#*******

[monitor://<path>]
* This directs Splunk to watch all files in <path>.
* <path> can be an entire directory or just a single file.
* You must specify the input type and then the path, so put three slashes in your path if you'r

# Additional attributes:

host_regex = <regular expression>
* If specified, <regular expression> extracts host from the filename of each input.
* Specifically, the first group of the regex is used as the host.
* If the regex fails to match, the default "host =" attribute is used.

host_segment = <integer>
* If specified, the '/' separated segment of the path is set as host.
* If the value is not an integer, or is less than 1, the default "host =" attribute is used.

whitelist = <regular expression>
* If set, files from this path are monitored only if they match the specified regex.
* Takes precedence over the deprecated _whitelist, which functions the same.

blacklist = <regular expression>
* If set, files from this path are NOT monitored if they match the specified regex.
* Takes precedence over the deprecated _blacklist, which functions the same.

Note: Wildcards and monitor:
* You can use wildcards to specify your input path for monitored input. Use ... for paths and *
* ... recurses through directories until the match is met. This means that /foo/.../bar will ma
 * To recurse through a subdirectory, use another .... For example /foo/.../bar/....
* * matches anything in that specific path segment. It cannot be used inside of a directory pat
 * Combine * and ... for more specific matches:
 * foo/.../bar/* matches any file in the bar directory within the specified path.

crcSalt = <string>
* Use this to force Splunk to consume files with matching CRCs.
* Set any string to add to the CRC.
* If set to "crcSalt = <SOURCE>", then the full source path is added to the CRC.

followTail = 0 | 1
* If set to 1, monitoring begins at the end of the file (like tail -f).
* This only applies to files the first time Splunk sees them.
* After that, Splunk's internal file position records keep track of the file.

alwaysOpenFile = 0 | 1
* Opens a file to check if it has already been indexed.
* Only useful for files that don't update modtime.
* Should only be used for monitoring files on Windows, and mostly for IIS logs.
* NOTE: This flag should only be used as a last resort, as it increases load and slows down inc
```

```
time_before_close = <integer>
* Modtime delta required before Splunk can close a file on EOF.
* Tells the system not to close files that have been updated in past <integer> seconds.
* Defaults to 3.

recursive = true|false
* if false, will not go into subdirectories found within a monitored directory
* defaults to true

followSymlink
* if false, will ignore symbolic links found within a monitored directory
* defaults to true

_whitelist = ...
* This setting is deprecated.  It is still honored, unless whitelist exists as well.

_blacklist = ...
* This setting is deprecated.  It is still honored, unless blacklist exists as well.

dedicatedFD = ...
* This setting has been removed.  It is no longer needed.


#*******
# BATCH:
#*******

NOTE: Batch should only be used for large archives of historic data. If you want to continuousl

[batch://<path>]
* One time, destructive input.
* For continuous, non-destructive inputs, use **monitor**.

# Additional attributes:

move_policy = sinkhole.
* Important = You must set move_policy = sinkhole.
* This loads the file destructively.
* Do not use this input type for files you do not want to consume destructively.

host_regex (see MONITOR, above)
host_segment (see MONITOR, above)

# IMPORTANT: The following are not used by batch:
source = <string>
<KEY> = <string>


#*******
# TCP:
#*******

[tcp://<remote server>:<port>]
* Configure Splunk to listen on a specific port.
* If a connection is made from <remote server>, this stanza is used to configure the input.
* If <remote server> is blank, this stanza matches all connections on the specified port.

# Additional attributes:
```

```
connection_host = ip | dns | none
* Set to "ip," "dns" or "none."
* "ip" (or "false") sets the TCP input processor to rewrite the host with the IP address of the
* "dns" sets the host to the DNS entry of the remote server.
* "none" leaves the host as specified for this stanza.
* Defaults to ip.


#*******
# Data distribution:
#*******

[splunktcp://<remote server>:<port>]
* This is the same as TCP, except the remote server is assumed to be a Splunk server.
* For SplunkTCP, the host or connection_host will be used if the remote Splunk server does not
* See documentation (http://www.splunk.com/doc/latest/admin/ForwardingReceiving) for help.

enableS2SHeartbeat = true | false
* This allows detection of dead forwarders due to network, firewall etc
* Splunk will monitor the connection for presense of heartbeat and if the heartbeat is
* not seen for s2sHeartbeatTimeout, receiver will close the connection.
* This overrides the default value specified at global splunktcp stanza.
* This is true by default

s2sHeartbeatTimeout = <seconds>
* Timeout value in seconds
* Splunk will monitor for forwarder connections and close the connection if
* heartbeat is not seen for s2sHeartbeatTimeout seconds
* This overrides the default value specified at global splunktcp stanza.
* Has default value of 600 seconds

[splunktcp]
route = has_key | absent_key:<key>:<queueName>;...
* Settings for the light forwarder.
* Splunk sets these parameters automatically -- you DO NOT need to set them.
* The property route is composed of rules delimited by ';'.
* Splunk checks each incoming data payload via cooked tcp port against the route rules.
* If a matching rule is found, Splunk sends the payload to the specified <queueName>.
* If no matching rule is found, Splunk 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.


compressed = true | false
* Send compressed data? y/n?
* Defaults to false.
* If this is set to true, the forwarder port should also have compression turned on.

enableS2SHeartbeat = true | false
* This specifies global keepalive setting for all splunktcp ports.
* This is true by default

s2sHeartbeatTimeout = <seconds>
* This specifies global keepalive setting for all splunktcp ports.
* Has default value of 600 seconds

inputShutdownTimeout = <seconds>
* This flag is used during shutdown. To minimize data loss when forwarders are connected
```

```
  to receiver, during shutdown the tcp input processor waits for given amount of
  seconds and then closes the connection after timeout. As a result, shutdown completion
  is delayed by given amount. If the connections close before the timeout period, the
  shutdown proceeds normally.

# SSL settings for data distribution:

[splunktcp-ssl:PORT]
* Use this stanza name if you are sending encrypted, cooked data from Splunk.
* Set PORT to the port on which your forwarder is sending cooked, encrypted data.
* Forwarder settings are set in outputs.conf on the forwarder-side.

enableS2SHeartbeat = true | false
* Please see doc for [splunktcp:PORT]

s2sHeartbeatTimeout = <seconds>
* Please see doc for [splunktcp:PORT]

compressed = true | false
* Send compressed data? y/n?
* Defaults to false.
* If this is set to true, the forwarder port should also have compression turned on.


[tcp-ssl:PORT]
* Use this stanza name if you are sending encrypted, raw data from a third-party system.
* Set PORT to the port on which your forwarder is sending raw, encrypted data.

[SSL]
* Set the following specifications for SSL underneath this stanza name:

serverCert = <path>
* Full path to the server certificate.

password = <string>
* Server certificate password, if any.

rootCA = <string>
* Certificate authority list (root file).

requireClientCert = true | false
* Toggle whether it is required for a client to authenticate.
supportSSLV3Only = <true|false>
* If true, tells the inputproc to only accept connections
* from SSLv3 clients.
* Default is false.

cipherSuite = <cipher suite string>
* If set, uses the specified cipher string for the input processors.
* 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.


#*******
# UDP:
#*******

[udp://<port>]
```

```
* Similar to TCP, except that it listens on a UDP port.


# Additional attributes:

_rcvbuf = <integer>
* Specify the receive buffer for the UDP port (in bytes).
* If the value is 0 or negative, it is ignored.
* Defaults to 1,572,864.
* Note: The default in the OS varies.

no_priority_stripping = true
* If this attribute is set to true, then Splunk does NOT strip the <priority> syslog field from
* NOTE: Do NOT include this key if you want to strip <priority>.

no_appending_timestamp = true
* If this attribute is set to true, then Splunk does NOT append a timestamp and host to receive
* NOTE: Do NOT include this key if you want to append timestamp and host to received events.


#*******
# FIFO:
#*******

[fifo://<path>]
* This directs Splunk to read from a FIFO at the specified path.


#*******
# Scripted Input:
#*******

[script://<cmd>]
* Runs <cmd> at a configured interval (below) and indexes the output.
* The command must reside in $SPLUNK_HOME/etc/system/bin/, $SPLUNK_HOME/etc/apps/$YOUR_APP/bin/
* We ship severeal Windows-only scripted inputs. Check towards the end of inputs.conf.example f
* <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 executed.  Whereas the pointer file is bound by the same location
restrictions mentioned above, the command referenced inside it can reside
anywhere on the file system.  This file must contain exactly one line, which
would be the path to the command to execute, optionally followed by command
line arguments.  Additional empty lines and lines that begin with '#' are also
permitted and will be ignored.

interval = <integer>|<cron schedule>
* How often to execute the specified command (in seconds), or a valid cron schedule.
* NOTE: when a cron schedule is specified the script is not executed on start up
* Defaults to 60 seconds.

passAuth = <username>
* User to run the script under.
* If you provide a username, Splunk generates an auth token for that user and passes it to the


#*******
# File system change monitor
#*******
```

```
NOTE: You cannot simultaneously watch a directory using fs change monitor and monitor (above).

[fschange:<path>]
* Monitors all add/update/deletes to this directory and sub directories.
* NOTE: <path> is the direct path.  You do not need to preface it with // like other inputs.
* Sends an event for every change.

# Additional attributes:
# NOTE: fschange does not use the same attributes as other input types (above).  Use only the f

index = <indexname>
* The index to store all events generated.
* Defaults to _audit, unless you do not set signedaudit (below) or set signedaudit = false, in

signedaudit = true | false
* Send cryptographically signed add/update/delete events.
* If set to true, events are *always* sent to the '_audit' index and will *always* have the sou
* If set to false, events are placed in the default index and the source type is whatever you s
* You must set signedaudit to false if you wish to set the index.
* NOTE: You MUST also enable auditing in audit.conf.
* Defaults to false.

filters = <filter1>,<filter2>,...<filterN>
* Each filter is applied left to right for each file or directory found during the monitor's po
* See "File System Monitoring Filters" below for help defining a filter.

recurse = true | false
* If true, recurse directories within the directory specified in [fschange].
* Defaults to true.

followLinks = true | false
* Follow symbolic links if true.
* It is recommended that you do not set this to true or file system loops may occur.
* Defaults to false.

pollPeriod = <integer>
* Check this directory for changes every <integer> seconds.
* Defaults to 3600.

hashMaxSize = <integer>
* Calculate a SHA256 hash for every file that is less than or equal to <integer> bytes.
* This hash is used as an addional method for detecting changes to the file/directory.
* Defaults to -1 (disabled).

fullEvent = true | false
* Set to true to send the full event if an add or update change is detected.
* Further qualified by the 'sendEventMaxSize' attribute.
* Defaults to false.

sendEventMaxSize  = <integer>
* 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 sourcetype for events from this input.
* "sourcetype=" is automatically prepended to <string>.
* Defaults to audittrail (if signedaudit=true) or fschange (if signedaudit=false).
```

```
host = <string>
* Set the host for events from this input
* Defaults to whatever host sent the event

index = <string>
* Set the index for events from this input
* Defaults to the main index

filesPerDelay = <integer>
* Injects a delay specified by 'delayInMills' after processing <integer> files.
* This is used to throttle file system monitoring so it doesn't consume as much CPU.

delayInMills = <integer>
* The delay in milliseconds to use after processing every <integer> files as specified in 'file
* This is used to throttle file system monitoring so it doesn't consume as much CPU.


#*******
# File system monitoring filters:
#*******

[filter:<filtertype>:<filtername>]
* Define a filter of type <filtertype> and name it <filtername>.

<filtertype>
* Filter types are either 'blacklist' or 'whitelist.'
* A whitelist filter processes all file names that match the regex list.
* A blacklist filter skips all file names that match the regex list.

<filtername>
* The filter name is used in the comma-separated list when defining a file system monitor.

regex<integer> = <regex>
* Blacklist and whitelist filters can include a set of regexes.
* The name of each regex MUST be 'regex<integer>', where <integer> starts at 1 and increments.
* Splunk applies each regex in numeric order:
  regex1=<regex>
  regex2=<regex>
  ...

#*******
# WINDOWS INPUTS:
#*******

* Windows platform specific input processor.
* Security, Application, System are enabled by default.  To disable an input type,
comment it out or set disabled = 1 in $SPLUNK_HOME\etc\apps\windows\local\inputs.conf
* You can configure Splunk to read other Windows event logs as well, but you must
import them to the Windows Event Viewer first, and then add them to your local
copy of inputs.conf (in $SPLUNK_HOME\etc\apps\windows\local\inputs.conf).
Just use the same format as the ones shown below [WinEventLog:<event log name>]
and the line disabled = 0.

[WinEventLog:<Log Name>]
* Define a windows event log to monitor

disabled = <integer> 1|0
* Enable or disable this input.
```

```
start_from = <string> oldest|newest
* oldest - Start reading windows event log chronologicaly from oldest to newest
* newest - Start reading windows event log in revers, from newest to oldest.  Once the
backlog of events is consumed, then it will start picking up the newest events

current_only = <integer> 1|0
* If set to 1 it emulates tail, only monitor new coming events.  If set to 0 it will first
get all existing events in the system and then monitor events coming in real time.

checkpointInterval = <integer> seconds
* An integer greater then 0.  Sets the interval of how often the windows event log checkpoint
it will be saved.  The default value is 5.

evt_resolve_ad_obj = <integer> 1|0  Enables/disables resolving active directory objects like
GUID/SID objects for a specific windows event log channel.  By default this option
it turned on for Security event logs.  Optionally you can specify the Domain Controller
name and/or DNS name of the domain to bind to which then splunk will use to resolve the
AD objects.

evt_dc_name = <string> Optional, this parameter can be left empty.
Domain Controller Name to bind to.  This name can be the 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.  All of 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"

evt_dns_name = <string> Optional, this parameter can be left empty.
Fully-qualified DNS name of the domain to bind to

* There are several Windows-only scripted inputs that we ship.  They are
defined in the default inputs.conf.  By default some of them are enabled and
others are disabled.  This is a list of the input stanzas:
    [script://$SPLUNK_HOME\bin\scripts\splunk-wmi.path]
    [script://$SPLUNK_HOME\bin\scripts\splunk-regmon.path]
    [script://$SPLUNK_HOME\bin\scripts\splunk-admon.path]
    [script://$SPLUNK_HOME\bin\scripts\splunk-perfmon.path]
* Use the "disabled=" parameter to enable/disable any of these inputs.
* Short summary of the inputs:
  * WMI: retreives events logs remotely and locally.  It can also gather
    performance data, as well as receive various system notifications.
  * RegMon: It uses a driver to track and report any changes that occur in the
    local system registry.
  * ADMon: It indexes existing AD object and listens for AD changes.
  * PerfMon: Retrieves performance data.
```

inputs.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles


# The following configuration directs Splunk to read all the files in the directory /var/log.

[monitor:///var/log]


# The following configuration directs Splunk to read all the files under /var/log/httpd and cla
# as sourcetype::access_common.

[monitor:///var/log/httpd]
sourcetype = access_common


# The following configuration directs Splunk to read all the files under /mnt/logs. When the pa
# /mnt/logs/<host>/... it sets the hostname (by file) to <host>.

[monitor:///mnt/logs]
host_segment = 3


# The following configuration directs Splunk to listen on TCP port 9997 for raw data from ANY r
# (not just a Splunk instance). The host of the data is set to the IP address of the remote ser

[tcp://:9997]


# The following configuration directs Splunk to listen on TCP port 9995 for raw data from ANY r
# 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 directs Splunk to listen on TCP port 9995 for raw data from 10.1.
# 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 sets a global default for data payloads sent from the light forwa
# The route parameter is an ordered set of rules that is evaluated in order for each payload of

[splunktcp]
route=has_key:_utf8:indexQueue;has_key:_linebreaker:indexQueue;absent_key:_utf8:parsingQueue;ab


# The following configuration directs Splunk to listen 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 directs Splunk to listen on TCP port 9998 for distributed search
# 10.1.1.100. The data is processed the same as locally indexed data.

[splunktcp://10.1.1.100:9996]


# The following configuration directs Splunk to listen 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


# Set up SSL:

[SSL]
serverCert=$SPLUNK_HOME/etc/auth/server.pem
password=password
rootCA=$SPLUNK_HOME/etc/auth/cacert.pem
requireClientCert=false

[splunktcp-ssl:9996]


# Enable Windows Registry monitoring (Windows only)
# This example shows how to enable Windows Registry monitoring as a scripted input.
# Because the Windows Registry can generate a high volume of events, Windows Registry monitorin
# is also affected by two other configuration files, sysmon.conf and regmon.conf:
# * sysmon.conf contains global settings for which event types (adds, deletes, renames, and so
# to monitor, which regular expression filters from the regmon-filters.conf file to use, and
# whether or not Windows registry events are monitored at all.
# * regmon-filters.conf contains the specific regular expressions you create to refine and filt
# the hive key paths you want Splunk to monitor.
# Splunk recommends that you refer to the documentation about Windows Registry monitoring at
# http://www.splunk.com/base for more details.
# You must make the change shown below in inputs.conf in $SPLUNK_HOME/etc/system/local/.
# You must restart Splunk to enable configurations.

[script://$SPLUNK_HOME\bin\scripts\splunk-regmon.py]
interval = 60
sourcetype = WinRegistry
source = WinRegistry
disabled = 0

# Enable WMI input (Windows only)
# This example shows how to enable WMI input as a scripted input.
# WMI input is also affected by configurations in wmi.conf.
# Splunk recommends that you refer to the documentation about WMI input at http://www.splunk.co
# for more details.
```

```
# You must make this change in inputs.conf in $SPLUNK_HOME/etc/apps/windows/local/.
# You must restart Splunk to enable configurations.

[script://$SPLUNK_HOME\bin\scripts\splunk-wmi.py]
disabled = 0


# Use file system change monitor:

[fschange:/etc/]
fullEvent=true
pollPeriod=60
recurse=true
sendEventMaxSize=100000
index=main

# Monitor Windows event logs Security:

[WinEventLog:Security]
disabled = 0
start_from = oldest
current_only = 0
evt_dc_name =
evt_dns_name =
evt_resolve_ad_obj = 1
checkpointInterval = 5

# Monitor Windows event logs ForwardedEvents:

[WinEventLog:ForwardedEvents]
disabled = 0
start_from = newest
current_only = 1
checkpointInterval = 5
```

limits.conf

limits.conf

The following are the spec and example files for limits.conf.

limits.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles
#
# 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.

[searchresults]
* This stanza controls search results for various Splunk search commands

maxresultrows = <integer>
* Configures the maximum number of events that will be generated by search commands which
grow the size of results (such as multikv) or that create events. Other search commands are exp
controlled in specific stanzas below.
* 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>
* Retry period.
* Defaults to 500.

[subsearch]
* This stanza controls subsearch results.

maxout = <integer>
* Maximum number of results to return from a subsearch.
* Defaults to 100.

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.
* Defaults to 300.

[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]

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.


[concurrency]
max_count = <integer>
* Maximum concurrency level to keep track of
* Defaults to 10000000


[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
and column field)
* Defaults to 1000.


[correlate]

maxfields = <integer>
* Maximum number of fields to correlate.
* Defaults to 1000.


[discretize]
* This stanza set attributes for bin/bucket/discretize.

maxbins = <integer>
* Maximum number of buckets to discretize into.
* If maxbins is not specified or = 0, it defaults to searchresults::maxresultrows (which is by

[inputcsv]
mkdir_max_retries = <integer>
* Maximum number of retries for creating a tmp directory (with random name as subdir of SPLUNK_
* Defaults to 100.


[join]
subsearch_maxout = <integer>
* Maximum result rows in output from subsearch that we 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]

maxdatapoints = <integer>
* Maximum data points to do kmeans clusterings for.
* Defaults to 100000000

maxkvalue = <integer>
```

```
* Maximum number of cluster 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]

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 50

maxchars = <integer>
* truncate _raw to to this size and then do auto KV
* Defaults to 10240

[lookup]

max_memtable_bytes = <integer>
* maximum size of static lookup file to use a in-memory index for
* Defaults to 10000000

max_matches = <integer>
* maximum matches for a lookup
* Defaults to 1000

max_reverse_matches = <integer>
* maximum reverse lookup matches (for search expansion)
* Defaults to 500

[metrics]

maxseries = <integer>
* The number of series to include in the per_x_thruput reports in metrics.log.
* Defaults to 10.

[rare]

maxresultrows = <integer>
* Maximum number of result rows to create.
* If not specified, defaults to searchresults::maxresultrows (which is by default 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]

maxresultrows = <integer>
* Maximum result rows to be returned by /events or /results getters from REST API.
```

```
* Defaults to 50000.

[search]

ttl = <integer>
* How long searches should be stored on disk once completed, in seconds.
* Defaults to 600, which is equivalent to 10 minutes.

status_buckets = 0
* The approximate maximum number of timeline buckets to maintain.
* Defaults to 0.

max_count = <integer>
* The last accessible event in a call that takes a base and bounds.
* Defaults to 10000.

truncate_report = <bool>
* Apply the max_count limit to report output?
* Defaults to false

min_prefix_len = <integer>
* The minimum length of a prefix before a * to ask the index about.
* Defaults to 1.

max_results_raw_size = <integer>
* The largest "_raw" volume that should be read in memory.
* If the total volume of _raw fields (the text of the events) exceeds this value, no more resul
will be returned for the search.
* Defaults to 100000000, which is 100MB.

cache_ttl = <integer>
* The length of time to persist search cache entries (in seconds).
* Defaults to 300.

reduce_freq = <integer>
* Attempt to reduce intermediate results every how many chunks (0 = never)
* 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
* we retry 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 the maximum number of searches computed as a multiplier of the CPUs
* Defaults to 4

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_searche
* Note: the maximum number of real-time searches is computed as: max_rt_searches = max_rt_searc
x max_hist_searches
* Defaults to 4
```

```
max_rt_search_multiplier = <decimal number>
* a number by which the maximum number of historical searches is multiplied to determine the ma
* number of concurrent real-time searches
* Note: the maximum number of real-time searches is computed as: max_rt_searches = max_rt_searc
x max_hist_searches
* Defaults to 3

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 >= 1, default is 100

realtime_buffer = <int>
* maximum number of accessible events to keep for real-time searches from the UI
* Acts as circular buffer once this limit is reached
* must be >= 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 2000

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 o
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 preview is computed as a ratio o
amount of time that the search has been running, or as a ratio of the length of the time window
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

replication_period_sec  = <int>
* the minimum amount of time in seconds between two successive configuration file replications
* defaults to 60

sync_bundle_replication = <bool>
* flag indicating whether configurtion file replication blocks searches or is run asynchronousl
* NOTE: setting this flag to false could cause searches to run with out-of-sync configuration f
on different search peers
* defaults to true

multi_threaded_setup = <bool>
* flag indicating whether to use multiple threads when setting up distributed search to multipl
* defaults to false

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

[realtime]
# default options for indexer support of real-time searches
# these can all be overriden for a single search via REST API arguments
queue_size = <int>
* size of queue for each real-time search (must be >0)
* default is 10000

blocking = <bool>
* should indexer block if a queue is full?
* defaults is false

max_blocking_secs = <int>
* maximum time to block if the queue is full (meaningless if blocking = false)
* default = 60
* 0 means no limit

indexfilter = <bool>
* should the indexer prefilter events for efficiency?
* default is true

[slc]

maxclusters = <integer>
* Maximum number of clusters to create.
* Defaults to 10000.

[stats|sistats]

maxresultrows = <integer>
* Maximum number of result rows to create.
* 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 100000 for stats and 1000 for sistats

maxvaluesize = <integer>
* Maximum length of a single value to consider.
* Defaults to 1000 for stats and 200 for sistats

# rdigest is a data structure used to compute approximate order statistics (such as median and
using sublinear space
rdigest_k = <integer>
* rdigest compression factor
* lower k = more compression
* after compression, number of nodes guaranteed to be <= 11*k
* default = 100, must be >=2

rdigest_maxnodes = <integer>
```

```
* maximum rdigest nodes before automatic compression is triggered
* default = 1, meaning automatically configure based on k value

[thruput]

maxKBps = <integer>
* If specified and not zero, this limits the speed through the thruput processor to the specifi
kilobytes per second.

[top]

maxresultrows = <integer>
* Maximum number of result rows to create.
* If not specified, defaults to searchresults::maxresultrows (which is by default 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.

[transactions]

maxopentxn    = <integer>
* maximum number of open transaction or events in open
* Defaults to 5000

maxopenevents = <integer>
* transaction before transaction eviction happens
* Defaults to 100000

[inputproc]

max_fd = <integer>
* Maximum number of file descriptors that Splunk will keep open, to capture any trailing data f
files that are written to very slowly.
* Defaults to 100

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 f
all monitor inputs.

tailing_proc_speed = <integer>
* REMOVED.  This setting is no longer used.

[scheduler]
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 25

max_action_results = <integer>
* the maximum number of results to load when triggering an alert action
* Defaults to 10000
```

```
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 20


[show_source]
max_count = <integer>
* maximum number of events accessible by show_source.  show source will fail when more than thi
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)


[typeahead]
maxcount = <integer>
* Maximum number of typeahead results to find
* Defaults to 1000

use_cache = <bool>
* Specifies whether the typeahead cache will be used if use_cache is not specified in the comma
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>
* The period, in seconds, for how long the typeahead cached results are valid
* Defaults to 300


min_prefix_length = <integer>
* The minimum string prefix for which to provide typeahead
* Defaults to 1


[typer]
maxlen = <int>
* in eventtyping, pay attention to first N characters of any attribute (such as _raw), includin
tokens. Can be overridden by supplying the typer operator with the argument maxlen (for example
* defaults to 10000
```

limits.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
# 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
# maximum time to wait for an already running subsearch
timeout = 30
# 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/r
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 return 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 largest "_raw" volume that should be read in memory
max_results_raw_size = 100000000

# the length of time to persist search cache entries (in seconds)
cache_ttl = 300

[slc]
# maximum number of clusters to create
maxclusters = 10000

[stats]
maxresultrows = 50000
maxvalues = 10000
maxvaluesize = 1000

[top]
maxresultrows = 50000
# maximum distinct value vectors to keep track of
maxvalues = 100000
maxvaluesize = 1000
```

literals.conf

literals.conf

The following are the spec and example files for literals.conf.

literals.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains attribute/value pairs for configuring externalized strings in literals.con
#
# 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://www.splunk.com/base/Documentation/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

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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 tempora
SERVER_RESTART_MESSAGE = This Splunk Server's configuration has been changed.  The server needs
UNABLE_TO_CONNECT_MESSAGE = Could not connect to splunkd at %s.
```

macros.conf

macros.conf

The following are the spec and example files for macros.conf.

macros.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[$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 append with "(<numargs>)", where <numargs> is the n
        * macros may be overloaded so there can be [foobar] and [foobar(1)] and [foobar(2)] etc
        * macros can be used in the search language by enclosing the macro name and any argumen
        * no macro expansion are attempted inside of quoted values, e.g. "foo`bar`baz"

args = <string>
        * A comma delimited string of argument names.
        * Argument names may only contain the characters alphanumerics and underscore '_' and d
        * If the stanza name indicates this macro takes no arguments, this key will be ignored
        * It is an error if this list contains any repeated elements

definition = <string>
        * The string that the macro will expand to, with the arguments filled in. (exception if
        * Arguments to be filled in must be wrapped by dollar signs $ e.g. "the last part of th
        * If a $ is not followed by the name of an argument (specified in the args list) follow
        * the $...$ pattern will be replace globally in the string, even inside of quotes

validation = <string>
        * A validation string that is an 'eval' expression.  This expression must statically ei
        * This validation is for verifying that the argument value used to invoke this macro ar
        * If the validation expression is a boolean expression, validation succeeds when it ret
        * If the validation expression is not a boolean expression, it is expected to return a

errormsg = <string>
        * The error message to be displayed if validation is a boolean expression and it does n

iseval = <true/false>
        * If true, 'definition' is expected to be an eval expression that returns a string that
        * Defaults to false
```

macros.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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
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
# `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 argumen
# 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 displaye
[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 rep
[fooeval(2)]
args = foo, bar
definition = if (bar > 0, "$foo$ + $bar$", "$foo$ - $bar$")
iseval = true
```

multikv.conf

multikv.conf

The following are the spec and example files for multikv.conf.

multikv.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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
# 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) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles
#
# NOTE:  Configure multikv.conf only if you are unhappy with Splunk's automatic multikv
# behavior.  If you use the multikv search command with successful outcome, there is no reason
```

```
# create this file.

# A table-like event includes a table, which in turn consists of four parts or sections:
#
#-------------------------------------------------------------------------------------
# Section Name | Description
#-------------------------------------------------------------------------------------
# pre          | optional: info/description (eg the system summary output in top)
# header       | optional: if not defined, fields are named Column_N
# body         | required: this is the body of the table from which child events are constructe
# post         | optional: info/description
#-------------------------------------------------------------------------------------

# NOTE: Each section up to and including the section for processing must have both a section
# definition (below) and processing (also below) set.

[multikv_config_name]
        * Name your stanza to use with the mulitkv search command:
                ex: '.... | multikv conf=$STANZA_NAME rmorig=f | ....'
        * Follow this stanza name with any number of the following attribute/value pairs.

######################
# 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 event-start or end of previous section where this section starts (ir
        * Use this if you cannot define a regex for the start of the section.

section_$NAME.member = <regex>
        * A line membership test.
        * Member iff 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
######################
# Set processing for each section.
#

section_$NAME.ignore = <string-matcher>
        * Member lines matching this string matcher will be ignored and thus NOT processed furt
        * <string-matcher> = _all_ | _none_ | _regex_ <regex-list>
```

```
section_$NAMEe.replace = <quoted-str> = <quoted-str>, <quoted-str> = <quoted-str>....
        * List of the form toReplace = replaceWith.
        * Can have any number of toReplace = replaceWith.
        * Example: "%" = "_", "#" = "_"

section_$NAME.tokens     = <chopper> | <tokenizer> | <aligner> | <token-list>
        * See below for definitions of each possible $VAL.

<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>
        * <delims> = comma-separated list of delimiting chars.
        * Tokenize the string using the delim characters.
        * This generates at most max_tokens tokens
        * Set max_tokens to:
                * -1 for complete tokenization
                * 0 to inherit from previous section (usually header)
                * Or to 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
        * Note: consecutive delimiters treated as an empty field.


<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
                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 in the output of 'ls -lah'
        which misses a header altogether.
```

multikv.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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://www.splunk.com/base/Documentation/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]
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

body.end     = "^\s*$"
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

outputs.conf

The following are the spec and example files for outputs.conf.

outputs.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains possible attributes and values for configuring outputs.conf. Configure
# Splunk's data forwarding actions by creating your own outputs.conf.
#
# There is NO DEFAULT outputs.conf. To set custom configurations, place an outputs.conf
# $SPLUNK_HOME/etc/system/local/.  For examples, 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles
#
# NOTE: Place outputs.conf on the forwarding side of any distributed Splunk deployment.
# To learn more about distributed configurations, see the documentation at
# TODO: the article below does not exist in the latest documentation:
# http://www.splunk.com/doc/latest/admin/ForwardingReceiving


############
#----TCP Output-----
############
# These configurations will be used if they are not overwritten in specific target groups.
# All events that do not have target group metadata will be sent to this group.
# If there is more than one group specified, the events will be cloned to all listed.

[tcpout]
defaultGroup= Group1, Group2, ...
attribute1 = val1
attribute2 = val2
...

# TODO : Deprecate the feature below. Use syslog out.
#NOTE: This is not for typical use:
#This configuration item looks in the event for <key>.  If the event contains this
#this key, the value is prepended to the raw data that is sent out to the destination
#server.  Note that this ONLY works if 'sendCookedData = false'. The key/value pair
#and how it is derived is set in props.conf and transforms.conf.
#Use case: appending <priority> to a syslog event which has been obtained by monitoring
#a syslog file and sending it out to a syslog server.
prependKeyToRaw = key

#This parameter is available only via the top level [tcpout] stanza, it can not be over
#-ridden in a target group:
indexAndForward = true | false
* In addition to other actions, index all this data locally as well as forwarding it.
* This is known as an index and forward configuration.
* Defaults to false.

#----TARGET GROUP CONFIGURATION-----

# You can have as many target groups as you wish.
# If more than one is specified, the forwarder will clone every event into each target group.
```

```
[tcpout:$TARGET_GROUP]
server=$IP:$PORT, $IP2:$PORT2...
attribute1 = val1
attribute2 = val2
...

#----SINGLE SERVER CONFIGURATION-----

# You can define specific configurations for individual indexers here on a server by server
# basis.  However, you must include any single server in a target group or default group
# to send data to it.

[tcpout-server://$IP:$PORT]
attribute1 = val1
attribute2 = val2
...

#----OPTIONAL SETTINGS----

# These attributes are optional.

sendCookedData = true | false
* If true, events are cooked (have been processed by Splunk and are not raw).
* 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' is true.
* Defaults to 30 seconds.

blockOnCloning = true | false
* If true, TcpOutputProcessor blocks till at least one of the cloned group gets events. This wi
  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
  the each cloned groups are full. When at least one of the cloned groups are up and Queues are
  the event is not dropped.
* Defaults to true.


compressed = true | false
* Send compressed data? y/n?
* Defaults to false.
* If this is set to true, the receiver port should also have compression turned on.

#----QUEUE SETTINGS----

maxQueueSize = <integer>
* The maximum number of queued events (queue size) on the forwarding server.
* Defaults to 1000.

dropEventsOnQueueFull = <integer>
* If set to a positive number N, wait N * 5 seconds before throwing out all new events until th
* Setting this to -1 or 0 will set the queue to block when it gets full causing blocking up the
* When any target group's queue is blocked, no more data will reach any other target group.
* Using load balanced groups is the best way to alleviate this condition because multiple
    receivers must be down (or jammed up) before queue blocking occurs.
```

```
* Defaults to -1 (do not drop events).
* DO NOT SET THIS VALUE TO A POSITIVE INTEGER (true) IF YOU ARE MONITORING FILES!

#----BACKOFF SETTINGS when connecting to indexer----

# The settings in this section determine how forwarders should retry when an indexer
# becomes unavailable.

backoffAtStartup = <integer>
* Set how long (in seconds) to wait until retrying the first time a retry is needed.
* Defaults to 5.

initialBackoff = <integer>
* Set how long (in seconds) to wait until retrying every time after the first retry.
* Defaults to 2.

maxNumberOfRetriesAtHighestBackoff = <integer>
* Specifies the number of times the system should retry after reaching the highest back-off
period before stopping completely.
* -1 means to try forever.
* Splunk recommends that you not change this from the default, or the forwarder will completely
stop forwarding to a downed URI at some point.
* Defaults to -1 (forever).

maxBackoff = <integer>
* Specifies the number of seconds before reaching the maximum backoff frequency.
* Defaults to 20.

#----BACKOFF SETTINGS when failure to send event to indexer----
# The settings in this section determine how forwarder should slow down when there
# is repeated failures in sending event to indexer.

maxFailuresPerInterval = <integer>
* Specifies the maximum number failures that is allowed per interval before backoff
  takes place. The interval is defined below.
* Defaults to 5.

secsInFailureInterval = <integer>
* Number of seconds in an interval. If number of write failure exceed maxFailuresPerInterval
  in the specified secsInFailureInterval seconds, forwarder sleeps for backoffOnFailure seconds
* Defaults to 1.

backoffOnFailure = <integer>
* When failures exceed maxNumberOfRetriesAtHighestBackoff in secsInFailureInterval seconds,
  forwarder sleeps for backoffOnFailure seconds before attempting to send data.
* Defaults to 5.

disableBackoffOnFailure = true | false
* If set to true, it turns off backoff policy on sending failures. This does not affect
  backoff settings for connecting to indexer
* Defaults to false - Backoff policy active by default

#----Configuring which events are forwarded by index----
forwardedindex.<n>.whitelist = <regex>
forwardedindex.<n>.blacklist = <regex>
* This should not be changed normally. This controls which events are forwarded based
  on the indices they belong to.
* Ordered list of whilelist and blacklist which decide if an index should be forwarded.
  The order is determined by <n> and should start from 0 to any positive number.
```

```
  There should not be any gap in the numbers between lowest and largest number. These
  filters can start from either whitelist or blacklist and are tested from
  forwardedindex.0 to forwardedindex.<max>
* Please see $SPLUNK_HOME/system/default/outputs.conf for default setting.

forwardedindex.filter.disable = true | false
* Disables index filter on true. If disabled, all the indices are forwarded.
* Defaults to false.

#----Automatic Load-balancing of forwarders

autoLB = true | false
* If set to true, forwarder switches to automatic load balancing mode. In this mode, the
* forwarder selects a new indexer every autoLBFrequency randomly. If connection to indexer
* is lost at any point, it selects a new live indexer and forwards data to it.
* If this flag is not present, forwarder uses load balancing with round robin strategy.
* Defaults to false.

autoLBFrequency = <seconds>
* This is used in automatic load balancing mode. Every autoLBFrequency a new indexer is
* selected randomly from list of indexers provided in server parameter
* 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.

sslCertPath = <path>
* If specified, this connection will use SSL.
* This is the path to the client certificate.
* There is no default value.

sslRootCAPath = <path>
* The path to the root certificate authority file (optional).
* TODO: default = ?

sslVerifyServerCert = true | false
* If true, 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.

sslCommonNameToCheck = <string>
* 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.
* You must specify this setting if 'sslVerifyServerCert' is true.

altCommonNameToCheck = <string>
* Check the alternate name of the server's certificate against this name.
* If there is no match, assume that Splunk is not authenticated against this server.
* You must specify this setting if 'sslVerifyServerCert' is true.

############
```

```
#----Syslog output----
############
# The following configuration is used to send output using syslog

[syslog]
defaultGroup = Group1, Group2, ...

[syslog:$TARGET_GROUP]
attribute1 = val1
attribute2 = val2
...

#----REQUIRED SETTINGS----
# Required settings for syslog output:

server = ip/servername:<port>
* IP or servername where syslog server is running
* Port on which syslog server is listening.
* There is no default value. You must specify a port. Syslog, by default, uses 514.

#----OPTIONAL SETTINGS----

# Optional settings for syslog output:

type = tcp | udp
* Protocol used. If type is not specified, default is udp.

priority = <ddd>
* ddd is value that will appear as <ddd> in the syslog header
* Users should compute ddd as (<facility> * 8) + <severity>
* If facility is 4(security/authorization messages) and severity is 2 (Critical: critical
conditions), priority value will be 34 = (4 * 8) + 2.
* TODO: default = ?

syslogSourceType = <string>
* string representing sourceType for syslog.
* In absense of this atrribute, "sourcetype::syslog" is assumed as the source type for syslog m

timestampformat = <%b %e %H:%M:%S>
* If specified, the format is used when adding timestamp into header
* TODO: default = ?

#---- Routing data to syslog server -----
To route data to syslog server:
1) First, decide which events to route to which servers.
2) Then, edit the props.conf, transforms.conf, and outputs.conf files on the forwarding servers

Edit $SPLUNK_HOME/etc/system/local/props.conf and set a TRANSFORMS-routing attribute as shown b
[<spec>]
TRANSFORMS-routing=$UNIQUE_STANZA_NAME

<spec> can be:
<sourcetype>, the sourcetype 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 stanz
```

```
[$UNIQUE_STANZA_NAME]
REGEX=$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

#############
#----HTTP output----
#############
# The following configuration is used to send output via http:

[httpoutput]
defaultGroup = Group1, Group2, ...

[httpoutput:$TARGET_GROUP]
attribute1 = val1
attribute2 = val2
...

#----REQUIRED SETTINGS----

# Required settings for HTTP output:

username = <username>
* username used to authenticate against splunk indexer

password = <password>
* password used to authenticate against splunk indexer

server = ip/servername:port
* ip/servername of splunk receiver
* port that splunk receiver is listening on

#----OPTIONAL SETTINGS----

# Optional settings for HTTP output:

ssl = true | false
* Set SSL for HTTP output.
* Defaults to true.

#---- Routing data to splunk instance with http -----
To route data to splunk server:
1) First, decide which events to route to which servers.
2) Then, edit the props.conf, transforms.conf, and outputs.conf files on the forwarding servers

Edit $SPLUNK_HOME/etc/system/local/props.conf and set a TRANSFORMS-routing attribute as shown b
[<spec>]
TRANSFORMS-routing=$UNIQUE_STANZA_NAME

<spec> can be:
<sourcetype>, the sourcetype 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 $SPLUNK_HOME/etc/system/local/transforms.conf and set rules to match your props.conf stanz
[$UNIQUE_STANZA_NAME]
REGEX=$YOUR_REGEX
DEST_KEY=_HTTP_ROUTING
FORMAT=$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 _HTTP_ROUTING to send events via HTTP
Set FORMAT to $UNIQUE_GROUP_NAME. This should match the syslog group name you create in outputs

#############
#----Index And Forward-----
#############
# IndexAndForward processor is used to determine the default behavior for indexing data
# When forwarders(tcpout, httpoutput) are configured, it turns 'index' to 'false'.
# When forwarders are not configured, 'index' is set to 'true'.
#
# If tcpout stanza is configured with 'indexAndForward', then value of 'index' is
# set to value of 'indexAndForward'.
#
# The setting of 'index' can be overridden in [indexAndForward] stanza which supercedes
# the values determined earlier.

[indexAndForward]
index = true | false
* If set to true, data is indexed.
* If set to false, data is not indexed.
```

outputs.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains an example outputs.conf.  Use this file to configure forwarding in a distr
# 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://www.splunk.com/base/Documentation/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

[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
# balanced (round-robin) 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 and send heartbeats every
# 45 seconds with a maximum queue size of 100,500 events.

[tcpout:group4]
server=foo.Splunk.com:9997
heartbeatFrequency=45
maxQueueSize=100500


# Set the hearbeat frequency to 15 for each group and clone the events to
# groups indexer1 and indexer2.  Also, index all this data locally as well.

[tcpout]
heartbeatFrequency=15
indexAndForward=true

[tcpout:indexer1]
server=Y.Y.Y.Y:9997

[tcpout:indexer2]
server=X.X.X.X:6666



# Round-Robin data balance between Y.Y.Y.Y and X.X.X.X.

[tcpout:indexerGroup]
server=Y.Y.Y.Y:9997, 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:

[syslog:syslog-out1]
disabled = false
server = X.X.X.X:9099
type = tcp
```

```
priority = 34
timestampformat = %b %e %H:%M:%S

# HTTP Output configuration

[httpoutput:httpout1]
server=indexer1:8089
ssl = true
username=admin
password=changeme


# 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 achive 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/cacert.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 ru
[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 serv
[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

#
# The following example shows how to route events to splunk instance using http
# 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=httpRouting

2. Edit $SPLUNK_HOME/etc/system/local/transforms.conf and set errorRouting and httpRouting rule
[errorRouting]
REGEX=error
DEST_KEY=_HTTP_ROUTING
FORMAT=errorGroup

[httpRouting]
REGEX=.
DEST_KEY=_HTTP_ROUTING
FORMAT=httpGroup
```

```
3. Edit $SPLUNK_HOME/etc/system/local/outputs.conf and set which http outputs go to with server
[httpoutput]
defaultGroup=everythingElseGroup

[httpoutput:httpGroup]
server=10.1.1.197:8089
ssl = true
username=admin
password=changeme

[httpoutput:errorGroup]
server=10.1.1.200:8089
ssl = true
username=admin
password=changeme

[httpoutput:everythingElseGroup]
server=10.1.1.250:8089
ssl = true
username=admin
password=changeme
```

pdf_server.conf

pdf_server.conf

The following are the spec and example files for pdf_server.conf.

pdf_server.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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 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://www.splunk.com/base/Documentation/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
```

pdf_server.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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 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://www.splunk.com/base/Documentation/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

procmon-filters.conf

The following are the spec and example files for procmon-filters.conf.

procmon-filters.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains potential attribute/value pairs to use when configuring Windows registry
# monitoring. The procmon-filters.conf file is used in conjunction with sysmon.conf, and
# contains the specific 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[<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.
        This must be a subset of those defined for the event_types attribute in regmon-filters.

hive = <string>
        * Not used in this contexted, but should always have value ".*"
```

procmon-filters.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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 cor
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[default]
hive = .*

[not-splunk-optimize]
proc = (?<!splunk-optimize.exe)$
type = create|exit|image
```

props.conf

props.conf

The following are the spec and example files for props.conf.

props.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains possible attribute/value pairs for configuring Splunk's processing propert
# via props.conf.
#
# 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 strin
# in Splunk Web:
#
# | extract reload=T
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[<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.
* 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 for an event.
3. source::<source>, where <source> is the source 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 classifi
These are only considered as a last resort before generating a new source type based on the sou
```

Precedence:

For settings that are specified in multiple categories of matching stanzas,
[host::<hostpattern>] spec settings override [<sourcetype>] spec settings.
Additionally, [source::<sourcepattern] spec settings override both
[host::<hostpattern>] and [<sourcetype>] settings.

Patterns:

When setting a <spec>, use the following regex-type syntax:

... = recurses through directories until the match is met.

* = matches anything but / 0 or more times.

| = or

( ) = used to limit scope of |.

Example: [source::....(?<!tar.)(gz|tgz)]

Considerations for Windows file paths:

When specifying Windows-based file paths as part of a source spec, you must escape any backslas

Example: [source::c:\\path_to\\file.txt]

Match language:

These match expressions must match the entire key value, not just a substring.

For those familiar with regular expressions, these are a full implementation
(PCRE) with the translation of ..., * and .
Thus . matches a period, * non-directory seperators, and ... any number of any characters.
For more information see the wildcards section at:

http://www.splunk.com/base/Documentation/latest/Admin/Specifyinputpathswithwildcards

Pattern collisions:

Suppose the source of a given input matches multiple source patterns. If the
stanzas for these patterns each supply distinct settings, all of these settings
are applied.

However, suppose two stanzas supply the same setting. In this case, we choose
the value to apply based on the ASCII order of the patterns in question.
For example, suppose we have a 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 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".

Building the final stanza:

The final stanza is built by layering together:

   (1) 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 for the priority field for:
     - pattern matching stanzas is 0
     - literal matching stanzas is 100

NOTE: setting priority to a value greater than 100 will result in the stanzas
      which are pattern matched to override the values of the literal matching
      stanza.

The priority key may also be used to resolve collisions between sourcetype
patterns and between host patterns. Note, however, that the priority key will
not affect precendence across spec types. For example, source patterns take
priority over host and sourcetype patterns, regardless of priority key values.

#*******************************************************************************
# The possible attributes/value pairs for props.conf, and their
# default values, are:
#*******************************************************************************

# International characters

CHARSET = <string>
* When set, Splunk assumes the input from the given <spec> is in the specified encoding.
* 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 fur
* If the source encoding is valid, but some characters from the <spec> are not valid in the spe
* When set to "AUTO", Splunk attempts to automatically determine the character encoding and con
* For a complete list of the character sets Splunk automatically detects, see the online docume
* Can only be used on the basis of sourcetype, or source::<spec>, not host::<spec>.
* Defaults to ASCII.

#*******************************************************************************
# Line breaking
```

```
#*******************************************************************************

# Use the following attributes to define the length of a line.

TRUNCATE = <non-negative integer>
* Change the default maximum line length.
* Set to 0 if you do not want truncation ever (very long lines are, however, often a sign of ga
* Defaults to 10000.

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 SHOULD_LINEMERGE)
* Defaults to ([\r\n]+), meaning data is broken into an event for each line, delimited by \r or
* The regex must contain a matching group.
* Wherever the regex matches, the start of the first matching group is considered the end of th
  previous event, and the end of the first matching group is considered the start of the next e
* The contents of the first matching group is ignored as event text.
* NOTE: There is a significant speed boost by using the LINE_BREAKER to delimit multiline event
  rather than using line merging to reassemble individual lines into events.

LINE_BREAKER_LOOKBEHIND = <integer>
* Change the default lookbehind for the regex based linebreaker.
* When there is leftover data from a previous raw chunk, this is how far before the end the raw
* Defaults to 100.

# Use the following attribute to define multi-line events with
# additional attributes and values.

SHOULD_LINEMERGE = true | false
* When set to true, Splunk combines several lines of data into a single event, based on the fol
* Defaults to true.

# When SHOULD_LINEMERGE = True, use the following attributes to define the multi-line events.

BREAK_ONLY_BEFORE_DATE = true | false
* When set to true, Splunk creates a new event if and only if it encounters a new line with a d
* Defaults to true.

BREAK_ONLY_BEFORE = <regular expression>
* When set, Splunk creates a new event if and only if it encounters a new line that matches the
* Defaults to empty.

MUST_BREAK_AFTER = <regular expression>
* When set, and the regular expression matches the current line, Splunk creates a new event for
* 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 su
* 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
* 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.
```

```
#*******************************************************************************
# Timestamp extraction configuration
#*******************************************************************************

DATETIME_CONFIG = <filename relative to $SPLUNK_HOME>
* Specifies which file configures the timestamp extractor.
* This configuration may also be set to "NONE" to prevent the timestamp extractor from running
* Defaults to /etc/datetime.xml (eg $SPLUNK_HOME/etc/datetime.xml).

MAX_TIMESTAMP_LOOKAHEAD = <integer>
* Specifies how far (in characters) into an event Splunk should look for a timestamp.
* Defaults to 150.

TIME_PREFIX = <regular expression>
* Specifies the necessary condition for timestamp extraction.
* The timestamping algorithm only looks for a timestamp after the first regex match.
* Defaults to empty.

TIME_FORMAT = <strptime-style format>
* Specifies a strptime format string to extract the date.
* For more information on strptime see `man strptime` or "Configure timestamp recognition" in t
* This method of date extraction does not support in-event timezones.
* TIME_FORMAT starts reading after the TIME_PREFIX.
* For good results, the <strptime-style format> should describe the day of the year and the tim
* 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 (e.g., UTC, -08:00), use that.
* If TZ is set to a valid timezone string, use that.
* Otherwise, use the timezone of the system that is running splunkd.
* Defaults to empty.

MAX_DAYS_AGO = <integer>
* Specifies the maximum number of days past, from the current date, for an extracted date to be
* If set to 10, for example, Splunk ignores dates that are older than 10 days ago.
* Defaults to 2000.
* IMPORTANT: If your data is older than 2000 days, change this setting.

MAX_DAYS_HENCE = <integer>
* Specifies the maximum number of days in the future, from the current date, for an extracted d
* If set to 3, for example, dates that are more than 3 days in the future are ignored.
* False positives are less likely with a tighter window.
* 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
* Defaults to 2.

MAX_DIFF_SECS_AGO = <integer>
* If the event's timestamp is more than <integer> seconds BEFORE the previous timestamp, only a
* IMPORTANT: If your timestamps are wildly out of order, consider increasing this value.
* Defaults to 3600 (one hour).

MAX_DIFF_SECS_HENCE = <integer>
* If the event's timestamp is more than <integer> seconds AFTER the previous timestamp only acc
* IMPORTANT: If your timestamps are wildly out of order, or you have logs that are written less
* Defaults to 604800 (one week).
```

```
#*******************************************************************************
# Transform configuration
#*******************************************************************************

# Use the TRANSFORMS class to create indexed fields.  Use the REPORT class to create extracted
# Please note that extracted fields are recommended as best practice.
# Note: Indexed fields have performance implications and are only recommended in specific circu
# You may want to use indexed fields if you search for expressions like foo!="bar" or NOT foo="
# Another common reason to use indexed fields is if the value of the field exists outside of th
# For example, if you commonly search for foo="1", but 1 occurs in many events that do not have
# For more information, see documentation at: http://www.splunk.com/doc/latest/admin/ExtractFie
# For examples, see props.conf.spec and transforms.conf.spec.

Precedence rules for classes:

* For each class, Splunk takes the configuration from the highest precedence configuration bloc
* If a particular class is specified for a source and a sourcetype, the class for source wins o
* Similarly, if a particular class is specified in ../local/ for a <spec>, it overrides that cl


TRANSFORMS-<value> = <unique_stanza_name>
*  <unique_stanza_name> is the name of your stanza from transforms.conf.
* <value> is any value you want to give to your stanza to identify its name-space.
* Transforms are applied in the specified order.
* If you need to change the order, control it by rearranging the list.

REPORT-<value> = <unique_stanza_name>
*  <unique_stanza_name> is the name of your stanza from transforms.conf.
* <value> is any value you want to give to your stanza to identify its name-space.
* Transforms are applied in the specified order.
* If you need to change the order, control it by rearranging the list.


EXTRACT-<class> = <regex> (in <src_field>)?
* Perform regex-based field extraction from the value of source field.
* 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.
* Note: this extraction is performed at search time *only*.
* Note: src_field can contain only the following chars: a-zA-Z0-9_ If your regex needs to end w
*       'in AlphaNumString' where AlphaNumString is not a field, to avoid confusion you should
*       to end with th following equivalent: '[i]n AlphaNumString'
* Where:
 * regex: a perl-compatible regex containing named capturing groups
 * src_field: name of the field to match the regex agains (defaults to _raw)


KV_MODE = none | auto | multi
* Specifies the key/value extraction mode for the data.
* Set KV_MODE to one of the following:
 * none: if you want no key/value extraction to take place.
 * auto: extracts key/value pairs separated by equal signs.
 * multi: invokes multikv to expand a tabular event into multiple events.
* Defaults to auto.


CHECK_FOR_HEADER = true | false
* 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),
* Can only be used on the basis of sourcetype, or source::<spec>, not host::<spec>.
```

```
* Defaults to false.

SEDCMD-<class> = <sed script>
* Specifie a sed script to apply to the _raw field at index time *only*.
 * 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
  * where regex is a perl regex (optionally containing capturing groups)
  * replacement is a string to replace the regex match, use \N for backreferences
  * 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]

LOOKUP-<class> = $TRANSFORM (<match_field> (AS <match_field_in_event>)?)+ (OUTPUT|OUTPUTNEW (<o
* Specifies a specifc lookup table and how to apply that lookup table to events
* <match_field> specifies a field in the lookup table to match on.
* By default will look for field with that same name in the event to match with (if <match_fiel
* multiple match fields may be provided, at least one is required
* <output_field> specifies a field in the lookup entry to copy into each matching event, where
* If that is not specified, <output_field> will be used.
* A list of output fields is not required.
* If not provided, all fields in the lookup table except for the match fields (and the timestam
* If the output field list starts with the keyword "OUTPUTNEW" instead of "OUTPUT", then the lo

FIELDALIAS-<class> = (<orig_field> AS <new_field>)+
* A list of fields to alias as new fields.
* Both fields will exist, i.e. the original field will not be removed.
* Field aliasing is performed after kv extraction but before lookups.
* Therefore, it is possible to specify a lookup based on a field alias.
* In addition, a field that is extracted at search time can be aliased.

#*******************************************************************************
# Binary file configuration
#*******************************************************************************

NO_BINARY_CHECK = true | false
* Can only be set for a [source::...] stanza.
* When set to true, Splunk processes binary files.
* By default, binary files are ignored.
* Can only be used on the basis of sourcetype, or source::<spec>, not host::<spec>.
* Defaults to false.

#*******************************************************************************
# Segmentation configuration
#*******************************************************************************

SEGMENTATION = <segmenter>
* Specifies the segmenter from segmenters.conf to use at index time.
* Set segmentation for any of the <spec> outlined at the top of this file.

SEGMENTATION-<segment selection> = <segmenter>
* Specifies that Splunk Web should use the specific segmenter (from segmenters.conf) for the gi
* Default <segment selection> choices are: all, inner, outer, raw.
* Do not change the set of default <segment selection> choices, unless you have some overriding


#*******************************************************************************
```

```
# File checksum configuration
#*******************************************************************************

CHECK_METHOD = endpoint_md5 | entire_md5 | modtime
* Set to 'endpoint_md5' to have Splunk checksum of the first and last 256 bytes of a file.  Whe
* Set this to "entire_md5" to use the checksum of the entire file.
* Alternatively, set this to "modtime" to check only the modification time of the file.
* Settings other than endpoint_md5 will cause splunk to index the entire file for each detected
* Defaults to endpoint_md5.


#*******************************************************************************
# Small file settings
#*******************************************************************************

PREFIX_SOURCETYPE = true | false
* NOTE: this attribute is only relevant to the "[too_small]" sourcetype.
* Determines the sourcetype given to files smaller than 100 lines, and therefore not classifiab
* False sets the sourcetype to "too_small."
* True sets the sourcetype to "<sourcename>-too_small", where "<sourcename>" is a cleaned up ve
* The advantage of a True value is that not all small files are classified as the same sourcety
* For example, a splunk search of "sourcetype=access*" will retrieve "access" files as well as
* Defaults to true.


#*******************************************************************************
# Sourcetype configuration
#*******************************************************************************

sourcetype = <string>
* Can only be set for a [<source>::...] stanza.
* Anything from that <source> is assigned the specified sourcetype.
* 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>
* With renaming, you can search for the sourcetype with sourcetype=<string>
* To search for the original sourcetype without renaming, use the field _sourcetype

invalid_cause = <string>
* Can only be set for a [<sourcetype>] stanza.
* Splunk does not index any data with invalid_cause set.
* Set <string> to "archive" to send the file to the archive processor (specified in unarchive_c
* Set to any other string to throw an error in the splunkd.log if running Splunklogger in debug
* Defaults to empty.

is_valid = true | false
* Automatically set by invalid_cause.
* 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::stanza
* <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)
```

```
* Defaults to empty.

unarchive_sourcetype = <string>
* the sourcetype of the contents of the matching archive file. Use this field instead of source
* to set the sourcetype 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 will strip off the
* archive file's extension (.gz, bz etc) and lookup another stanza to attempt to determine the
* sourcetype
* Defaults to empty.

LEARN_SOURCETYPE = <bool>
* whether learning of known or unknown sourcetypes is enabled. For known sourcetypes refer to I
* for unknown sourcetypes refer to rule:: and delayedrule:: configuration
* Setting this field to false disables CHECK_FOR_HEADER too
* Defauls to true

LEARN_MODEL = true | false
* For known sourcetypes, the fileclassifier adds a model file to the learned directory.
* To disable this behavior for diverse sourcetypes (such as sourcecode, where there is no good
* Defaults to empty.

maxDist = <integer>
* Determines how different a sourcetype model may be from the current file.
* The larger the value, the more forgiving.
* For example, if the value is very small (for example, 10), then files of the specified  sourc
* A larger value indicates that files of the given sourcetype vary quite a bit.
* Defaults to 300.
* If you're finding that a sourcetype model is matching too broadly, reduce its value of maxDis


# rule:: and delayedrule:: configuration

MORE_THAN<optional_unique_value>_<number> = <regular expression> (empty)
LESS_THAN<optional_unique_value>_<number> = <regular expression> (empty)

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 mat

#*******************************************************************************
# 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 sourcetypes.
* Defaults to empty.
```

props.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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 configuration
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles


########
# Line merging settings
########

# The following example linemerges source data into multi-line events for apache_error sourcety

[apache_error]
SHOULD_LINEMERGE = True



########
# Settings for tuning
########

# The following example limits the amount of characters indexed per event from host::small_ever

[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.

[source::.../mylogs/*.log]
DATETIME_CONFIG = NONE



########
# 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
# directory. This sets all events coming in from hosts starting with dharma to use this custom

[host::dharma*]
DATETIME_CONFIG = <etc/apps/custom_time/datetime.xml>
```

```
########
# Transform configuration
########

# The following example creates a search field for host::foo if tied to a stanza in transforms.

[host::foo]
TRANSFORMS-foo=foobar

# The following example creates an extracted field for sourcetype access_combined
# if tied to a stanza in transforms.conf.

[eventtype::my_custom_eventtype]
REPORT-baz = foobaz


# 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 *nix file web_access.log.

[source::.../web_access.log]
sourcetype = splunk_web_access

# The following example sets a sourcetype for the Windows file iis6.log.  Note: Backslashes wit

[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 examp
# with a smaller than default maxDist.

[custom_sourcetype]
LEARN_MODEL = true
maxDist = 30
```

```
# rule:: and delayedrule:: configuration
# The following examples create sourectype 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 = ####



# Extract a field at search-time based on the value of the source field
#
# Where sources look like:
#
# [monitor:///Users/logs/instance001/]
# [monitor:///Users/logs/instance003/]
# [monitor:///Users/logs/instance002/]
#
# This extracts and saves into myfield: instance001, instance002, and instance003.

[foo]
SHOULD_LINEMERGE = false
EXTRACT-rgb = (?<myfield>instance\w+) in source


########
# File configuration
########

# Binary file configuration
# The following example eats binary files from the host::sourcecode.

[host::sourcecode]
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

pubsub.conf

The following are the spec and example files for pubsub.conf.

pubsub.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains possible attributes and values for configuring a client of the PubSub syst
#
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

#*******************************************************************
# Configure the physical location where deploymentServer is running.
# This configuration is used by the clients of the pubsub system.
#*******************************************************************
[pubsub-server:deploymentServer]

disabled = <false or true>
    * defaults to 'false'

targetUri = <IP:Port or hostname:Port or "direct">
    * specify either the url of a remote server in case the broker is remote, or just the keywo
    * It is usually a good idea to co-locate the broker and the Deployment Server on the same S
    * 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]
disabled = false
targetUri = direct

[pubsub-server:<logicalName>]
    * It is possible for any Splunk to be a broker. If you have multiple brokers, assign a logi

disabled = <false or true>
    * defaults to 'false'

targetUri = <IP:Port or hostname:Port or "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 th
```

pubsub.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5

[pubsub-server:deploymentServer]
disabled=false
targetUri=somehost:8089

[pubsub-server:internalbroker]
disabled=false
targetUri=direct
```

regmon-filters.conf

regmon-filters.conf

The following are the spec and example files for regmon-filters.conf.

regmon-filters.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains potential attribute/value pairs to use when configuring Windows registry
# monitoring. The regmon-filters.conf file is used in conjunction with sysmon.conf, and
# contains the specific regular expressions you create to refine and filter the hive key paths
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[<stanza name>
        * Name of the filter being defined.

proc = <string>
        * Regex specifying process image that you want Splunk to monitor.

hive = <string>
        * Regex specifying the registry key path that you want Splunk to monitor.

type = <string>
        * Regex specifying the type(s) of registry event that you want Splunk to monitor.
        This must be a subset of those defined for the event_types attribute in regmon-filters.

baseline = <int 0|1>
        * Whether or not to establish a baseline value for the keys this filter defines.

baseline_interval = <int>
        * The threshold, in seconds, for how long Spunk has to have been down before re-taking
          snapshot.

disabled = <int 0|1>
        * Disables or enables a given filter.
```

regmon-filters.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains example registry monitor filters
#
# To use one or more of these configurations, copy the configuration block into
# regmon-filters.conf in $SPLUNK_HOME/etc/search/local/. You must restart Splunk to enable conf
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles
```

```
# The following are examples of registry monitor filters.  To create your own filter, modify
# the values by following the spec outlined in regmon-filters.conf.spec.

[default]
disabled = 1
baseline = 0
baseline_interval = 86400

[User keys]
proc = \\Device\\.*
hive = \\REGISTRY\\USER\\.*
type = set|create|delete|rename
```

report_server.conf

report_server.conf

The following are the spec and example files for report_server.conf.

report_server.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.0
#
# This file contains possible attributes and values you can use to configure Splunk's report se
#
# There is a report.conf in $SPLUNK_HOME/etc/system/default/.  To set custom configurations,
# place a report.conf in $SPLUNK_HOME/etc/system/local/.  For examples, see report.conf.example
# You must restart the report server to enable configurations.
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/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 the server.
        * 0 disables SplunkWeb, 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 '/'
   * Ex: if you want to proxy the splunk UI at http://splunk:8000/splunkui, then set root_endpo

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 webserver applies gzip compression to responses
   * Defaults to True



#
# cherrypy HTTP server config
#

server.thread_pool = <integer>
   * Specifies the numbers of threads the appserver 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 report_access.log

log.error_file = <filename>
   * Specifies the HTTP error log filename
   * Stored in default Splunk /var/log directory
   * Defaults to report_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 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
   * 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 mechahims
   * 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
   * WARNING: Disabling this will likely break the application, as all incoming strings are ass
   * 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 head
   * 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

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 reques
   * Defaults to 10
```

```
max_concurrent = <integer>
   * Specifies the maximum number of copies of Firefox that the report server will use concurre
   * 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 report server will only query Splunk appservers on IP addresses within the IP r
  * Networks can be specified as a prefix (10.1.0.0/16) or using a netmask (10.1.0.0/255.255.0.
  * 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 report server will only accept requests from hosts whose IP address falls withi
  * 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_ipadd
  * Defaults to False
```

report_server.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.0
#
# 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://www.splunk.com/base/Documentation/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 report server
appserver_ipaddr = 192.168.3.1,192.168.3.2
client_ipaddr = 192.168.3.1,192.168.3.2
```

restmap.conf

restmap.conf

The following are the spec and example files for restmap.conf.

restmap.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains possible attribute and value pairs for creating new 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://www.splunk.com/base/Documentation/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 paramater 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.

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.


############################
# 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 hanlder 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.
* Python is currently the only option for scripttype.

handler=<SCRIPT>.<CLASSNAME>
* The name and class name of the file to execute.
* The file *must* live in an application's ../rest/ subdirectory.
 * For example, $SPLUNK_HOME/etc/apps/<APPNAME>/default/rest/TestHandler.py
has a class called MyHandler (which, in the case of python must be derived from a base class ca
'splunk.rest.BaseRestHandler'). The tag/value pair for this is: "handler=TestHandler.MyHandler'

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 tag determines if this endpoint requires authentication or not.
* It defaults to 'true'.

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).

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.

script=<path to a script executable>
* Optional.
* Execute a script which is *not* derived from 'splunk.rest.BaseRestHandler'.
* Put the path to that script here.
* This is rarely used.
* Do not use this unless you know what you are doing.


############################
# '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.

########################
# 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> - an eval expression using the validate() function to
*                     evaluate arg correctness and return an error message.
*                     A boolean returning function can be used, and a generic
*                     message will be 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>/
*
* Example-1:
* action.email.sendresult = validate( isbool('action.email.sendresults'), "'action.email.sendre
*
* NOTE: use ' or $ to enclose field names that contain non alphanumeric characters
```

restmap.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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 configurati
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/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
# 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

savedsearches.conf

The following are the spec and example files for savedsearches.conf.

savedsearches.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains possible attribute/value pairs for saved search entries in savedsearches.c
# 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 configuration
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

#*******
# The possible attribute/value pairs for savedsearches.conf are:
#*******

[<stanza name>]
* Name your saved search.
* 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.
* Searchis not visible in Splunk Web if set to 1.
* 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 create a macro search, read the documentation: http://www.splunk.com/base/Documentation/la


#*******
# Scheduling options
#*******

enableSched = 0 | 1
* Set this to 1 to run your search on a schedule.
* Defaults to 0.


schedule = <cron-style string>
* This field is DEPRECATED as of 4.0 release use cron_schedule instead
* Cron-like schedule.
 * For example, */12 * * * *
* Note: Splunk's current cron implementation differs from standard POSIX cron.
 * Use the */n as "divide by n" (instead of standard POSIX cron's "every n").

cron_schedule = <cron string>
* The cron schedule used to execute this search.
* For example: */5 * * * *  will cause 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 s
  that they're staggered over time. This reduces system load. Running all of them (*/20) every
  minutes means they would all launch at hh:00 (20, 40) and might slow your system every 20 min


max_concurrent = <int>
* The maximum number of concurrent instances of this search the scheduler
* is allowed to run.
* Defaults to 1


realtime_schedule = 0 | 1
* The way the scheduler computes the next execution time of a scheduled search.
* If this value is set to 1, the scheduler would compute the next time based on the current tim
* otherwise, if it is set to 0, the next time is computed based on the last execution time (cor
*
*    If set to 1, some execution periods might be skipped to make sure that the scheduler is ex
*    the searches running over the most recent time range.
*
*    If set to 0, scheduled execution periods are guaranteed to never be skipped, however the e
*    of the savedsearch might fall behind depending on the scheduler's load. Use continous sche
*    you're enabling the summary index option
*
* The scheduler will try to execute searches that have the realtime_schedule set to 1 before ex
* searches that have a continous scheduling
*
* Defaults to 1

#*******
# Notification options
#*******
```

```
counttype = number of events | number of hosts | numbers 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).

relation = greater than | less than | equal to | drops by | rises by
* How to compare against counttype.

quantity = <integer>
   * Specify a value for relation and counttype.
   * For example, "number of events [is] greater than 10" sends an alert when the count of ever
   * For example, "number of events drops by 10%" sends an alert when the count of events drops

alert_condition = <string>
* A search that is evaluated on the artifacts of the saved search to determine whether to trigg
* Alerts are triggered if the search specified yields a non-empty search result list.


#*******
# generic action settings
# for a comprehensive list of actions and their arguments
# please refer to alert_actions.conf
#*******
action.<action_name> = 0 | 1
 * whether the action is enabled or disabled

action.<action_name>.<parameter> = <value>
 * override an action's parameter defined in alert_actions.conf

#******
# email action settings
#******
action.email.to = <email list>
 * a comma delimited list of recepient email addresses

action.email.from = <email address>
 * email address that would be used as the sender's address

action.email.subject = <string>
 * the subject of the email delivered to recepients

action.email.mailserver = <string>
 * address of the MTA server to be used to send emails


#******
# script action settings
#******
action.script = 0 | 1
* Toggle whether or not the script action is enabled.
* 1 to enable, 0 to disable.
* Defaults to 0

action.script.filename = <script filename>
* The filename of the shell script to execute. The script should live in $SPLUNK_HOME/bin/scrip


#*******
# Summary index settings
```

```
#*******

action.summary_index = 0 | 1
* Toggle whether or not the summary index is enabled.
* 1 to enable, 0 to disable.
* Defaults to 0.

action.summary_index._name = <index>
* The summary index where the results of the scheduled search are saved.
* Specify the summary index where the results of the scheduled search are saved.
* Defaults to summary.

action.summary_index.inline = <bool>
* whether to execute the summary indexing action as part of the scheduled search.
* NOTE: this option is considered if and only if the summary index action is enabled
*       and is always executed
* Defaults to true

action.summary_index.<KEY> = <string>
* Optional <KEY> = <string> to add to each event when saving it in the summary index.

#*******
# Lookup table population settings
#*******

action.populate_lookup = 0 | 1
* Toggle whether or not the lookup population action is enabled

action.populate_lookup.dest = <string>
* Path to a lookup csv file where to copy the search results to.
* NOTE:  This path must point to a .csv file in either of the following directories:
 * $SPLUNK_HOME/etc/system/lookups/
 * $SPLUNK_HOME/etc/apps/<app-name>/lookups
* NOTE: the destination directories of the above files must already exist

run_on_startup = true | false
* Toggle whether this search runs when Splunk starts.
* If it does not run on startup, it will run at the next scheduled time.
* It is recommended that you set this to true for scheduled searches that populate lookup table


#*******
# dispatch search options
#*******

* Read/write/connect timeout (in seconds) for the HTTP connection (to splunkd).
* Used to execute the scheduled search and any of its actions/alerts

dispatch.ttl = <integer>[p]
* Time to live (in seconds) for the artifacts of the scheduled search, if no actions are trigge
* If an action is triggered the ttl is changed to that actions's ttl, if multiple actions are t
* the maximum ttl is applied to the artifacts. For setting action's ttl refer to alert_actions.
* If the integer is followed by the letter 'p' the ttl is interpreted as a multiple of the sche
* Defaults to 2p.

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 10000.

dispatch.max_time = <integer>
* The maximum amount of time (in seconds) before finalizing the search.
* Defaults to 0.

dispatch.lookups = true | false
* Toggle whether lookups are enabled for this search.
* Defaults to true.

dispatch.earliest_time = <time-str>
* the earliest time for the search

dispatch.latest_time = <time-str>
* the latest time for the search

dispatch.time_format = <time format str>
* the time format used to specify the earliest and latest time

dispatch.spawn_process = <bool>
* whether to spawn a new search process when this saved search is executed (default true)


#*******
# UI-specific settings
#*******

displayview
* Defines the default UI view name (not label) in which to load the results
* Accessibility is subject to the user having sufficient permissions

vsid
* Defines the viewstate id associated with the UI view listed in 'displayview'
* Must match up to a stanza in viewstates.conf


is_visible = <bool>
* whether this saved search should be listed in the visible saved search list
* Defaults to true
```

savedsearches.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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 confi
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/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 NO
_Indexing_Volume_in_MBs = todaysBytesIndexed/1024/1024 | timechart avg(Daily_Indexing_Volume_in
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_ac
 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/kb
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

searchbnf.conf

The following are the spec and example files for searchbnf.conf.

searchbnf.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
#
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[<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]
      syntax = geocode <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 i
    * 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, appears-in
```

searchbnf.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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

segmenters.conf

The following are the spec and example files for segmenters.conf.

segmenters.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[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 br
        * By default, major breakers are set to most characters and blank spaces.
    * Typically, major breakers are single characters.
    * Default is [ ] < > ( ) { } | ! ; , ' " * \n \r \s \t & ? + %21 %26 %2526 %3B %7C %20 %2B
    * Please note: \s represents a space; \n, a newline; \r, a carriage return; and \t, a tab.


MINOR = <space separated list of strings>
        * Set minor breakers.
        * In addition to the segments specified by the major breakers, for each minor breaker f
        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,
        * 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

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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 configur
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/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

server.conf

The following are the spec and example files for server.conf.

server.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains possible attributes and values you can use to configure SSL and HTTP serve
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

# This file contains options for controlling the server configuration
# The only options currently available is controlling the SSL
# configuration of the server.

############################################################################################
# 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>.

sessionTimeout = <time range string>
    * 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
      SSO logins from trustedIP.
    * If 'always', all remote logins 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 the admin user that has not ch

############################################################################################
# SSL Configuration details
############################################################################################

[sslConfig]
        * Set SSL for communications on Splunk's 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 = <true | false>
        * Enables/disables SSL on the splunkd management port (8089).
    * Defaults to true.

useClientSSLCompression = <true | false>
    * 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 Splu
    * Defaults to true.

 useSplunkdClientSSLCompression = <true | false>
    * 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 sep
    * Defaults to false

supportSSLV3Only = <true|false>
        * If true, tells the HTTP server to only accept connections
        * from SSLv3 clients.
        * Default is false.

sslVerifyServerCert = <true|false>
        * 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 true, 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.
        * Default is false

sslCommonNameToCheck = <commonName>
        * The common name to check when 'sslVerifyServerCert' is set to true
        * Optional.  Defaults to no common name checking.

sslAltNameToCheck = <alternateName>
        * The alternate name to check when 'sslVerifyServerCert' is set to true
        * Optional.  Defaults to no alternate name checking

requireClientCert = <true|false>
        * Requires that any HTTPS client that connects to splunkds internal HTTPS server
        has a certificate that was signed by our certificate authority.
        * 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, 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.

sslKeysfile = <filename>
        * Server certificate file.
        * Certificates are auto-generated by splunkd upon starting Splunk.
        * You may replace the default cert with your own PEM format file.
        * Certs are stored in caPath (see below).
        * Default is server.pem.

sslKeysfilePassword = <password>
        * Server certificate password.
        * Default is password.

caCertFile = <filename>
        * Public key of the signing authority.
        * Default is cacert.pem.

caPath = <path>
        * path where all these certs are stored.
        * Default is $SPLUNK_HOME/etc/auth.

certCreateScript = <script name>
        * Creation script for generating certs on startup
          of Splunk.
        * Default is genSignedServerCert.sh

############################################################################################
# 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 = <int>
    * Set the maximum time (in seconds) to cache a static asset served off of the '/static' dir
    * 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)
        * Default value is 'false'.

############################################################################################
# Splunkd HTTPServer listner configuration
############################################################################################
```

```
[httpServerListener:<ip>:<port>]
        * Enable the splunkd http server to listen on a network interface (NIC) specified by
        <ip> and a port number specified by <port>.  If you leave <ip> blank (but still include
        splunkd will listen on the kernel picked NIC using port <port>.

ssl = <true|false>
        * Toggle whether this listening ip:port will use SSL or not.
        * Default value is 'true'.



############################################################################################
# Static file handler MIME-type map


[mimetype-extension-map]
        * Map filename extensions to MIME type for files served from the static file handler un
        this stanza name.

<file-extension> = <MIME-type>
    * Instructs the HTTP static file server to mark any files ending in 'file-exension'
         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

############################################################################################
# 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.

url = <URL>
        * Applications repository.
        * Defaults to http://www.splunkbase.com/api/apps

loginUrl = <URL>
        * Applications repository login.
        * Defaults to http://www.splunkbase.com/api/account:login/

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.

############################################################################################
# Misc. configuration
############################################################################################
```

```
[scripts]

initialNumberOfScriptProcesses = N
        * N is 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)
############################################################################################

[diskUsage]

minFreeSpace = <num>
        * Specifies a safe amount of space that must exist for splunkd to continue operating.
        * Used in two different ways, for indexing and for search.
        * For indexing:
            * The diskusage processor prevents Splunk from adding data to the indexes once
              the minimum diskspace allowed an index path is reached.
            * homePath and coldPath are checked for all indexes periodically.
        * 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.
        * Specified in megabytes.
        * The default setting is 2000.

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=<queueName>]

maxSize = <num>
        * The maximum number of pipeline data that queue can hold
        * The default is 1000

############################################################################################
# PubSub server settings for the http endpoint.
############################################################################################

[pubsubsvr-http]

disabled=<true or false>
```

```
    * If disabled, then http endpoint is not registered. Set this value to 'false' to
        expose PubSub server on http.
    * Defaults to 'true'

stateIntervalInSecs=300
    * How many seconds before a connection is flushed due to inactivity. Connectsions are not
        closed, only messages for that connection are flushed.
    * Defaults to 300 seconds/5 minutes.


############################################################################################
# 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 dignostic tool
############################################################################################

[diag]

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

server.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains an example server.conf.  Use this file to configure SSL and HTTP server op
#
# 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 configuratio
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

# Allow users 8 hours before they time out
[general]
sessionTimeout=8h

# Turn on SSL:

[sslConfig]
enableSplunkdSSL = true
useClientSSLCompression = true
sslKeysfile = server.pem
sslKeysfilePassword = password
caCertFile = cacert.pem
caPath = $$SPLUNK_HOME/etc/auth
certCreateScript = genSignedServerCert.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
```

serverclass.conf

serverclass.conf

The following are the spec and example files for serverclass.conf.

serverclass.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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 serve
# class member will receive from the deployment server.
#
# To define server classes for this deployment server to use when deploying content to deployme
# clients, place a serverclass.conf in $SPLUNK_HOME/etc/system/local/.
# For examples, see serverclass.conf.example. You must restart Splunk for changes to this file
# to take effect.
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/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 so-called clientName tag strings.
# 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:
# [serverClass] -> [serverClass:<name>] -> [serverClass:<scname>:app:<appname>]
#
# Some properties defined at a general level (say [serverClass]) can be
# overridden by the more specific stanzas as it applies to them. All inheritable
# properties are marked as such.
#*******************************************************************************

# Global stanza that defines properties for all server classes.
[global]

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 to install the apps and configuration content defin
    * If this value is unset, or set emptyr the repositoryLocation path is used.
    * Useful only for 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 that 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
    * Any custom URL can also be supplied here as long as it uses the specified variables.
    * This attribute does not need to be specified unless you have very specific need, for exam
    * Can be overridden at the serverClass level.
    * Defaults to $deploymentServerUri$/services/streams/deployment?name=$serverClassName$:$app

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 consid
        * Items that match any blacklist entry are not considered to match the stanza, regardle
    * 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 consid
        * 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>
blacklist.<n> = <clientName> | <ip address> | <hostname>
    * 'n' is a number starting at 0, and increasing by 1. Stop looking at the filter when 'n' h
    * 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
    * All of these can be used with wildcards.  * will match any sequence of characters.  For e
        * Match an network range: 10.1.1.*
        * Match a domain: *.splunk.com
    * These patterns are PCRE regular expressions with the additional mappings:
        * '.' is mapped to '\.'
        * '*' is mapped to '.*'
    * Can be overridden at the serverClass level, and the serverClass:app level.
    * There are no whitelist or blacklist entries by default.
```

```
# Note: Overriding one type of filter (whitelist/blacklist) causes the other to
# the overridden too. It is important to note that if you are overriding the
# whitelist, the blacklist will not be inherited from the parent - you must
# provide one in the stanza.

# Example of when filterType is 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

# Example of when filterType is 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 wi

# client machineTypes can also be used to match clients.
# This setting lets you use the hardware type of the deployment client as a filter.
# This filter will be used only if a client could not be matched using the whitelist/blacklist
# The value for machineTypes is a specific string that is designated by the hardware platform i
# The method for finding this string on the client itself will vary by platform, but if the dep
# is already connected to the deployment server, you can determine what this string is by using
# Splunk CLI command on the deployment server:
# <code>./splunk list deploy-clients</code>
# This will return a value for <code>utsname</code> that you can use to
# specify <code>machineTypes</code>.
machineTypes = <comma separated list>
    * Not used unless specified.
    * Match any of the machine types in the comma-delimited list.
    * Commonly used machine types: linux-x86_64, windows-intel, linux-i686, freebsd-i386, darwi
    * This filter will be used only if a client could not be matched using the whitelist/blackl
    * Note: be sure to include the 's' at the end of "machineTypes"
    * Can be overridden at the serverClass level, and the serverClass:app level.
    * This value is unset by default.

restartSplunkWeb = True | False
    * If True, restart SplunkWeb on the client when a member app or directly configured app is
    * Can be overridden at the serverClass level, and the serverClass:app level.
    * Defaults to False

restartSplunkd = True | False
    * If True, restart splunkd on the client when a member app or directly configured app is up
    * Can be overridden at the serverClass level, and the serverClass:app level.
    * Defaults to False

stateOnClient = enabled | disabled | noop
    * For enabled, set the application state to enabled on the client, regardless of state on t
    * For disable, set the application state to disabled on the client, regardless of state on
    * For 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.

[serverClass:<serverClassName>]
    * This stanza defines a server class. A serverClass is a collection of applications.
    * serverClassName is a unique name that is assigned to this serverClass.
    * A serverClass can override all inheritable properties from the [serverClass] stanza.
```

```
# Properties that you may want to typically override at this level include:
# repositoryLocation
# continueMatching
# filtering using whitelist/blacklist, startBuild, endBuild, machineType
# requiresRestart
# stateOnClient

[serverClass:<server class name>:app:<app name>]
    * This stanza adds an application that exists in repositoryLocation to the 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 serv
        * Other values 'someAppName' explicitly adds the app/configuration content to a server
    * Important note on matching: A server class must be matched before content belonging to th

appFile=<file name>
    * In cases where an appName is different from the file or directory name, you can use this
```

serverclass.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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: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
# at the app level. An app is matched _only_ if the server class it is contained in was succesf
whitelist.0=*

[serverClass:AppsByMachineType:app:SplunkDesktop]
# Deploy this app only to Windows boxes.
machineTypes=Windows-Intel

[serverClass:AppsByMachineType:app:unix]
# Deploy this app only to unix boxes - 32/64 bit.
machineTypes=linux-i686, linux-x86_64
```

serverclass.seed.xml.conf

serverclass.seed.xml.conf

The following are the spec and example files for serverclass.seed.xml.conf.

serverclass.seed.xml.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5

<!--
# This configuration is used by deploymentClient to seed a Splunk installation with applicatior
# 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 overriden by servi
    # In addition, deploymentclient.conf can control how this property is used by deploymentCli
    -->
    <endpoint>$deploymentServerUri$/services/streams/deployment?name=$serviceClassName$:$appNam

    <!--
    # The location on the deploymentClient where all applications will be installed. This value
    # app declarations below.
    # In addition, deploymentclient.conf can control how this property is used by deploymentCli
    -->
    <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
        -->
        <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

```
No example
```

source-classifier.conf

source-classifier.conf

The following are the spec and example files for source-classifier.conf.

source-classifier.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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 configura
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/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 p
        classifying a source.
```

source-classifier.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains an example source-classifier.conf.  Use this file to configure classificat
# 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://www.splunk.com/base/Documentation/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 thu

# 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
```

sourcetypes.conf

sourcetypes.conf

The following are the spec and example files for sourcetypes.conf.

sourcetypes.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc. All Rights Reserved. Version 4.1.5
#
# NOTE: sourcetypes.conf is a machine-generated file that stores the document models used by th
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles


_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

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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 th
# 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 configu
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

#
# This is an example of a machine-generated sourcetype models for a fictitious sourcetype cadca
#

[/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
```

sysmon.conf

sysmon.conf

The following are the spec and example files for sysmon.conf.

sysmon.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains possible attribute/value pairs for configuring registry monitoring
# on a Windows system, including global settings for which event types (adds, deletes, renames,
# and so on) to monitor, which regular expression filters from the regmon-filters.conf file to
# and whether or not Windows registry events are monitored at all.
# This file is used in conjunction with regmon-filters.conf.
# You must restart Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[<stanza name>]
        * Defaults to [RegistryMonitor]
        * Follow this stanza name with the following attribute/value pairs

filter_file_name = <string>
        * String representing the name of the file where filters for this monitor are stored

event_types = <string>
        * Regex string specifying the type of events to monitor. Can be delete, set, create, re
```

```
inclusive = <1 or 0>
        * 1 to specify that filter rules specified in active_filters field are inclusive(white
        0 the filter rules are exclusive(black list)

disabled = <1 or 0>
        * 1 to disable, 0 to enable.
```

sysmon.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains an example configuration for monitoring changes
# to the Windows registry. Refer to sysmon.conf.spec for details.
# The following is an example of a registry monitor filter and process monitor filter.
# To create your own filters, modify the values using the information in
# regmon-filters.conf.spec.
#
# To use one or more of these configurations, copy the configuration block into
# sysmon-filters.conf in $SPLUNK_HOME/etc/system/local/. You must restart Splunk to enable conf
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[RegistryMonitor]
filter_file_name = regmon-filters
event_types = set.*|create.*|delete.*|rename.*
disabled = 0

[ProcessMonitor]
filter_file_name = procmon-filters
event_types = create.*|exit.*|image.*
inclusive = 0
disabled = 1
```

tags.conf

tags.conf

The following are the spec and example files for tags.conf.

tags.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains possible attribute/value pairs for configuring tags.  Set any number of ta
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles
```

```
[<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 en
    * config file parsing errors especially if the field value contains the following character
    * 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.
    * Only one tag is allowed per stanza line.
```

tags.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This is an example tags.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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles
#
# This first example presents a situation where the field is "host" and the three hostnames for
# are "hostswitch," "emailbox," and "devmachine." Each hostname has two tags applied to it, one
# the "building1" tag has been applied to two hostname values (emailbox and devmachine).

[host=hostwitch]
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
```

tenants.conf

tenants.conf

The following are the spec and example files for tenants.conf.

tenants.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# Use tenants.conf to redirect incoming requests from deployment clients to another deployment
# server or servers. This is typically used for offloading load on your splunkd's HTTP server.
# This is not a typical configuration for deployment server. There is no default tenants.conf
#
# *There is no need to create/edit tenants.conf* unless you have worked with Splunk Professiona
# Services to design a custom deployment that includes explicit involvement of tenants.conf.
#
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

#*******************************************************************************
# Configure tenants (DeploymentServer instances) within the same Splunk server.
#
# Multiple instances of deployment servers can be configured withing the same Splunk instance
# using this configuration file.
# If this file is missing, a default DeploymentServer for tenant='default'is configured
# by the system, if there exists serverclass.conf or default-serverclass.conf.
#
# It is possible to redirect deployment clients to the appropriate instance of deployment serve
# by using a whitelist/blacklist mechanism, similar to the one in serverclass.conf.
#
# How does it all work?
# A DeploymentClient does a handshake with TenantService to determine which DeploymentServer
# it should be talking to. The TenantService will use this configuration to redirect a
# client to the appropriate deployment server (represented by phoneHomeTopic).
#
# How is multi-tenant configuration stored?
# Server class configuration for each tenant should be made available in:
# <tenantName>-serverclass.conf
#
# Configuration for the 'default' tenant can also be in 'serverclass.conf' - note the missing
# tenantName prefix.
#*******************************************************************************

[tenant:<tenantName>]

filterType = <whitelist or blacklist>
    * defaults to whitelist

whitelist.<n> = <ipAddress or hostname or clientName>
blacklist.<n> = <ipAddress of hostname of clientName>

    * 'n' is a number starting at 0, and increasing by 1. Stop looking at the filter when 'n' h
    * ipAddress of deployment client. Can also use wildcards as 10.1.1.*
    * hostname of deployment client. Can also use wildcards as *.splunk.com.
    * clientName- a logical or 'tag' name that can be assigned to each deployment client in dep
```

```
# Internal.
phoneHomeTopic=deploymentServer/phoneHome/$tenantName$
    * some unique suffix. Default is to use the tenant name. Make sure this value is unique.
    * Override this value only when you wish to script and roll your own deployment server.
```

tenants.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# Define two tenants - dept1 and dept2.
# DS Configuration for dept1 will be in a matching dept1-serverclass.conf
# DS Configuration for dept2 will be in a matching dept2-serverclass.conf

[tenant:dept1]
whitelist.0=*.dept1.splunk.com

[tenant:dept2]
whilelist.0=*.dept2.splunk.com
```

times.conf

times.conf

The following are the spec and example files for times.conf.

times.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles


[<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

earliest_time = <relative_time_identifier>
    * The relative time identifier string that represents the earliest event to
      to return, inclusive.
    * Optional.  If omitted, no earliest time bound is used.

latest_time = <relative_time_identifier>
```

```
    * The relative time identifier string that represents the latest event to
      to return, exclusive.
    * 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

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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://www.splunk.com/base/Documentation/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


#
# 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

transactiontypes.conf

The following are the spec and example files for transactiontypes.conf.

transactiontypes.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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 configur
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles


[<TRANSACTIONTYPE>]
* Create any number of transaction types, each represented by a stanza name and any number of t
* 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

maxspan = [<integer> s|m|h|d]
* Set the maximum time span for the transaction.
* Can be in seconds, minutes, hours or days, -1 for unlimited
 * For example:  5s, 6m, 12h or 30d.
* Defaults to: maxspan=-1

maxpause = [<integer> s|m|h|d]
* Set the maximum pause between the events in a transaction.
* Can be in seconds, minutes, hours or days, -1 for unlimited.
 * For example:  5s, 6m, 12h or 30d.
* Defaults to: maxpause=-1


maxevents = <integer>
* The maximum number of events in a transaction. If the value is a negative integer then this c
* 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: ""


startswith=<transam-filter-string>
* A search or eval filtering expression which if satisfied by an event marks the beginning of a
 * 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 trans
 * 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> is defined as
* syntax:   "<search-expression>" | (<quoted-search-expression>) | eval(<eval-expression>)
* description = Where:
*             <search-expression>       is a valid search expression that does not contain quo
*             <quoted-search-expression> is a valid search expression that contains quotes
*             <eval-expression>         is a valid eval expression that evaluates to a boolean
* Examples
 * search expression:     (name="foo bar")
```

```
 * search expression:     "user=mildred"
 * search expression:     ("search literal")
 * eval bool expression:  eval(distance/time < max_speed)
*

connected=<bool>
* Relevant iff fields is not empty. Controls whether an event that is not inconsistent and not
* with the fields of a transaction, opens a new transaction (connected=t) or is added to the tr
* An event can be not inconsistent and not consistent if it contains fields required by the tra
* but none of these fields has been instantiated in the transaction (by a previous event additi
* Defaults to: connected=t


### memory constraint options ###

maxopentxn=<int>
* Specifies the maximum number of not yet closed transactions to keep in the open pool before s
* to evict transactions, using LRU policy.
* Defautls to: the default value of this field is read from the transactions stanza in limits.c

maxopenevents=<int>
* Specifies the maximum number of events (which are) part of open transactions before transacti
* eviction starts happening, using LRU policy.
* Defaults to: the default value of this field is read from the transactions stanza in limits.c

keepevicted=<bool>
* Whether to output evicted transactions. Evicted transactions can be distinguished from non-ev
* transactions by checking the value of the 'evicted' field, which is set to '1' for evicted tr
* 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 ori
* events ordered in arrival order or (2) a set of unique field values ordered lexigraphically.
* 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=" "
```

transactiontypes.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This is an example transactiontypes.conf.  Use this file as a template to configure transacti
#
# To use one or more of these configurations, copy the configuration block into transactiontype
# in $SPLUNK_HOME/etc/system/local/.
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[default]
maxspan = 5m
```

```
maxpause = 2s
match = closest

[purchase]
maxspan  = 10m
maxpause = 5m
fields  = userid
```

transforms.conf

transforms.conf

The following are the spec and example files for transforms.conf.

transforms.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains possible attributes and values you can use to configure transform
# and event signing in transforms.conf.
#
# There is a transforms.conf in $SPLUNK_HOME/etc/system/default/.  To set custom configurations
# place a transforms.conf $SPLUNK_HOME/etc/system/local/.  For examples, see transforms.conf.ex
# You can enable configurations changes made to transforms.conf by typing the following search
# in Splunk Web:
#
# | extract reload=t
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles


[<unique_stanza_name>]
* Name your stanza. Use this name when configuring props.conf.
  For example, in a props.conf stanza, enter TRANSFORMS-<value> = <unique_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.

REGEX = <regular expression>
* Enter a regular expression to operate on your data.
* For transformations that occur at search time:
 * Name capturing groups in the REGEX are extracted directly to fields, meaning there is no nee
 * If both field name and field value are extracted using the REGEX, the following special capt
   groups can be used 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]+)
* Defaults to empty.
* NOTE: this option is valid for both index/search time KV extraction.

LOOKAHEAD = <integer>
* Specify how many characters to search into an event.
* Defaults to 256.
```

```
* NOTE: this option is valid only for index time KV extraction.

DEST_KEY = <KEY>
* Specify where to store the results of the REGEX.
* Use the KEYs listed below.
* NOTE: this option is valid only for index time KV extraction.

FORMAT = <string>
* Specify the format of the event, including any fields names or values you want to add.
* NOTE: this option is valid for both index/search time KV extraction.
** Index time:
* Use $n (for example $1, $2, etc) to specify the output of each REGEX match.
* If the regex does not have n groups, the matching fails.
* The special identifier $0 represents what was in the DEST_KEY before this regex was performed
* Defaults to $1.
** Search time:
* The format of this field as used during search time extractions is as follows:
* FORMAT = <field-name>::<field-value>( <field-name>::<field-value>)*
* where:
* field-name  = <string>|$<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  $4::$3
*
* Defaults to empty string




WRITE_META = <true | false>
* Automatically writes REGEX to metadata.
* Use instead of DEST_KEY = meta.
* Defaults to false.
* NOTE: this option is valid only for index time KV extraction.

DEFAULT_VALUE = <string>
* If set, and REGEX (above) fails, write this value to DEST_KEY.
* Defaults to empty.
* NOTE: this option is valid only for index time KV extraction.

SOURCE_KEY = <string>
* Set which KEY to perform the regex on.
* Defaults to _raw (the raw event).
* For index-time transformations use the KEYs listed below.
* For search-time extractions use any field that is available at the time of the
  execution of this field extraction
* NOTE: this option is valid for both index/search time KV extraction.


REPEAT_MATCH = <true | false>
* Specify whether to run REGEX several times on the SOURCE_KEY.
* REPEAT_MATCH starts wherever the last match stopped, and continues until no more matches are
* Defaults to false.
* NOTE: this option is valid only for index time KV extraction.
```

```
DELIMS = <quoted string list>
* Set delimiter characters to separate data into key-value pairs, and then to separate key from
* NOTE: Delimiters must be quoted with " " (to escape, use \).
* Usually, two sets of delimiter characters must be specified:
 * The first to extract key/value pairs.
 * The second to separate the key from the value.
* If you enter only one set of delimiter characters, then the extracted tokens:
 * Are named with names from FIELDS, if FIELDS are entered (below).
 * OR even tokens are used as field names while odd tokens become field values.
* Consecutive delimiter characters are consumed except when a list of field names is specified.
* NOTE: this option is valid only for search time KV extraction.

FIELDS = <quoted string list>
* List the names of the field values extracted using DELIMS.
* NOTE: If field names contain spaces or commas they must be quoted with " " (to escape, use \)
* Defaults to "".
* NOTE: this option is valid only for search time KV extraction.

MV_ADD = <bool>
* Option controlling 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
* NOTE: this option is valid only for search time KV extraction.

CLEAN_KEYS = <bool>
* Option controlling whether the keys extracted at search time are cleaned. Key cleaning
  is defined as the replacement of non-alphanumeric characters with underscores. Leading
  underscores and numbers are stripped.
* Defaults to true

CAN_OPTIMIZE = <bool>
* Option controlling whether Splunk can optimize this extraction out. An extraction is disabled
  if Splunk can determine that none of the fields extracted (by an extraction) will ever be
  needed for the successful evaluation of a search. This option should be rarely set to false
* Defaults to true


#*******
# 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_HOM
* If file is in multiple 'lookups' directories, no layering is done.
* Standard conf file precedence is used to disambiguate.

max_matches = <integer>
* Maximum number of matching for each input lookup value
* Default = 100 if non-temporal, default = 1 if temporal (i.e. time_field is specified).
* If non-temporal, the first (in file order) <integer> entries are used.
* If temporal, the first in descending time order <integer> are used.

min_matches = <integer>
* Minimum number of matches for each input lookup value
* Default = 0 for both temporal and non-temporal, meaning that nothing is output if no match is
* However, if min_matches > 0, and we get less than min_matches, then we emit the default_match
```

```
default_match = <string>
* If min_matches > 0 and we have less than min_matches for any given input, we write out this d
  one or more times such that the min_matches threshold is reached

case_sensitive_match = <bool>
* If set to false, case insensitive matching will be performed for all fields in a lookup table
* Default to true (case sensitive matching)

external_cmd = <string>
* Command and arguments to invoke to perform lookups.
* This string is parsed like a shell command.
* The first argument is expected to be a python script located in $SPLUNK_HOME/etc/<app_name>/b
* Presence of this field indicates that lookup is external command based.

fields_list = <string>
* A comma and space delimited list of all fields that are supported by the external command.

external_type = python
* Type of external command.
* Currently, only python is supported.

time_field = <string>
* For temporal (i.e. time bounded) lookups, specifies the field in the lookup table that repres
* Default = <empty string>, meaning that lookup is not temporal.

time_format = <string>
* For temporal lookups, specifies the 'strptime' format of the timestamp field.
* You can include subseconds but they will be ignored
* Default format is pure UTC time.

max_offset_secs = <integer>
* For temporal lookups, the maximum time in seconds that the event time may be
  ahead of lookup entry time for a match to occur
* Default is 2000000000 (no maximum).

min_offset_secs = <integer>
* For temporal lookups, the minimum time (in seconds) that the event time must be ahead of look
* Defaults to 0.


#*******
# 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 parsingQueue, nullQueue, indexQueue).
_raw  : The raw text of the event.
_done : If set to any string, this is the last event in a stream.
_meta : A space separated list of metadata for an event.
_time : The timestamp of the event, in seconds since 1/1/1970 UTC.
MetaData:FinalType  : The event type of the event.

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].

* NOTE: Any KEY prefixed by '_' is not indexed by Splunk, in general.
```

transforms.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

# Note: These are examples.  Replace the values with your own customizations.


# Indexed field:

[netscreen-error]
REGEX =  device_id=[^ ]+\s+\[w+\](.*)(?
FORMAT = err_code::$1
WRITE_META = true

# Extracted field:

[netscreen-error]
REGEX = device_id=[^ ]+\s+\[w+\](.*)(?
FORMAT = err_code::$1

# Override host:

[hostoverride]
DEST_KEY = MetaData:Host
REGEX = \s(\w*)$
FORMAT = host::$1


# Extracted fields:

[netscreen-error]
REGEX = device_id=[^ ]+\s+\[w+\](.*)(?
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

# 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.

# Extract key-value pairs
# This example extracts key-value pairs which are separated by '|'
# while the key is delimited from value by '='.

[pipe_eq]
DELIMS = "|", "="

# This example extracts key-value pairs which are separated by '|'
# while the key is delimited from value by '='.
```

user-seed.conf

user-seed.conf

The following are the spec and example files for user-seed.conf.

user-seed.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# Specification for user-seed.conf.  Allows configuration of Splunk's initial username and pass
# 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/default. You must restart Splunk to enable configurations.
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

[user_info]
USERNAME = <string>
          * Username you want to associate with a password.
          * Default is Admin.
PASSWORD = <string>
          * Password you wish to set for that user.
          * Default is changeme.
```

user-seed.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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 in
# $SPLUNK_HOME/etc/system/default/ prior to starting Splunk for the first time.
#
# 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 the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles


[user_info]
USERNAME = admin
PASSWORD = myowndefaultPass
```

web.conf

web.conf

The following are the spec and example files for web.conf.

web.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains possible attributes and values you can use to configure Splunk's web inter
#
# 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://www.splunk.com/base/Documentation/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.

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.

serviceFormPostURL = http://headlamp.Splunk.com/event/add
userRegistrationURL = https://www.Splunk.com/index.php/pre_reg?destination=prod_reg
updateCheckerBaseURL = http://quickdraw.Splunk.com/js/
   * 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

login_content = <content_string>
   * Add custom content to the login page
   * Supports any text including html

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 '/'
   * Ex: if you want to proxy the splunk UI at http://splunk:8000/splunkui, then set root_endpo

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

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
```

```
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
   * This should usually be disabled for security reasons
   * Defaults to False

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 v
   * Mode Server: Log to a defined server endpoint
   * See js/logger.js Splunk.Logger.Mode for mode implementation details and if you would like
   * Defaults to Firebug

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 buff
   * 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 in
   * Notifies client side pollers to stop, resulting in sessions expiring at the tools.sessions
   * If less than 1, results in no timeout notification ever being triggered (Sessions will sta
   * Defaults to 60 minutes

js_no_cache = [True | False]
   * Toggle js cache control
   * Defaults to False

enable_autocomplete_login = [True | False]
   * Indictes 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 True

#
# cherrypy HTTP server config
#

server.thread_pool = <integer>
   * Specifies the numbers of threads the appserver 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

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 re
    * Warning: setting this to very high numbers (eg. 10000) may impact performance during log
    * 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 byte
    * 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 r
    * Warning: setting this to very high numbers (eg. 10000) may impact performance during log
    * 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

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 mechahims
   * 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
   * WARNING: Disabling this will likely break the application, as all incoming strings are ass
   * 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 head
   * 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

pid_path = <filepath>
   * Specifies the path to the PID file
   * Defaults to var/run/splunk/splunkweb.pid

disabled_decomposers = <intention> [, <intention>]...
   * This is an experimental flag and should not be used.
   * Added in Splunk 4.1.5 as a short term workaround measure for certain problems with search
   * Comma separated list of intentions to be disabled.
   * 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 intentions are disabled.
   * Search decomposition in general is deprecated functionality.  Do not rely on being able to
   * The current possible values are: addcommand, stats, addterm, addtermgt, addtermlt, setfiel
   * Default is not present, in which case this setting leaves all intentions enabled.
```

web.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles



# This stanza heading must precede any changes.
[settings]

# Change the default port number:
httpport = 12800

# Turn on SSL:
# NOTE: paths are relative to $SPLUNK_HOME.
enableSplunkWebSSL = true
privKeyPath = /certs/myprivatekey.pem
caCertPath = /certs/mycacert.pem
```

wmi.conf

wmi.conf

The following are the spec and example files for wmi.conf.

wmi.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This file contains possible attribute/value pairs for configuring 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://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

############################################################################################
#----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 to wait (in seconds) before retrying the connection to the WMI provider after the fi
* If connection errors continue, the wait time doubles until it reaches max_backoff.
* Defaults to 5.

max_backoff = <integer>
* Maximum time (in seconds) to attempt reconnect.
* Defaults to 20.

max_retries_at_max_backoff = <integer>
```

```
* Try to reconnect this many times once max_backoff is reached.
* If reconnection fails after max_retries, give up forever (until restart).
* Defaults to 2.

checkpoint_sync_interval = <integer>
* Minimum wait time (in seconds) for state data (event log checkpoint) to be written to disk.
* Defaults to 2.

############################################################################################
#----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 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 list>
* A comma-separated list of servers from which to get data.
* If missing, defaults to local machine.

interval = <integer>
* How often to poll for new data.
* Not optional.
* No default.

disabled = 0 | 1
* 1 to disable, 0 to enable.
* If missing, defautls to 0.

hostname = <host>
* All results generated by this stanza will appear to have arrived from this host.
* Optional.  If missing, will detect the host automatically.

current_only = 0 | 1
* Default value,  current_only = 0
* When current_only is 1 -
*    For event log stanzas, this will only capture events that occur while Splunk is running.
*    For WQL stanzas, event notification query is expected.  The queried class must support ser
*    An example event notification query that watches for process creation:
*    SELECT * FROM __InstanceCreationEvent WITHIN 1 WHERE TargetInstance ISA 'Win32_Process'.
* When current_only is 0
*    For event log stanzas, all the events from the checkpoint are gathered. If there is no che
*    starting from the oldest events are retrieved.
*    For WQL stanzas, the query is executed and results are retrieved. The query is a non-notif
*    For example - Select * Win32_Process where caption = "explorer.exe"

* Event log-specific attributes:

event_log_file = <Application, System, etc>
* Use this instead of WQL to specify sources.
* Specify a comma-separated list of log files to poll.
* No default.

* WQL-specific attributes:

wql = <string>
* Use this if you are not using event_log_file.
```

```
* Specify wql to extract data from WMI provider.
* For example, select * from Win32_PerfFormattedData_PerfProc_Process where Name = "splunkd".

namespace = <string>
* The namespace where the WMI provider resides.
* Defaults to root\cimv2.
* 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.
```

wmi.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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 fi
#
# 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://www.splunk.com/base/Documentation/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

# These stanzas pull event logs from the local system.

[WMI:LocalApplication]
interval = 10
event_log_file = Application
disabled = 0

[WMI:LocalSystem]
interval = 10
event_log_file = System
disabled = 0

[WMI:LocalSecurity]
interval = 10
event_log_file = Security
disabled = 0

# These stanzas gather performance data from the local system.

[WMI:LocalPhysicalDisk]
interval = 1
wql = select Name, DiskBytesPerSec, PercentDiskReadTime, PercentDiskWriteTime, PercentDiskTime
disabled = 0

[WMI:LocalMainMemory]
interval = 10
wql = select CommittedBytes, AvailableBytes, PercentCommittedBytesInUse, Caption from Win32_Per
```

```
disabled = 0

[WMI:LocalSplunkdProcess]
interval = 1
wql = select * from Win32_PerfFormattedData_PerfProc_Process where Name = "splunkd"
disabled = 0

# Listen from three event log channels, capturing log events that occur only
# while Splunk is running.  Gather data from three servers.

[WMI:TailApplicationLogs]
interval = 10
event_log_file = Application, Security, System
server = srv1, srv2, srv3
disabled = 0
current_only = 1

# Listen for process-creation events on a remote machine

[WMI:ProcessCreation]
interval = 1
server = remote-machine
wql = select * from __InstanceCreationEvent within 1 where TargetInstance isa 'Win32_Process'
disabled = 0
current_only = 1

# Receive events whenever someone plugs/unplugs a USB device to/from the computer

[WMI:USBChanges]
interval = 1
wql = select * from __InstanceOperationEvent within 1 where TargetInstance ISA 'Win32_PnPEntity
disabled = 0
current_only = 1
```

workflow_actions.conf

workflow_actions.conf

The following are the spec and example files for workflow_actions.conf.

workflow_actions.conf.spec

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# 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
# 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 edit
# them through the Splunk manager.
#
# To learn more about configuration files (including precedence) please see the documentation
# located at http://www.splunk.com/base/Documentation/latest/Admin/Aboutconfigurationfiles

################################################################################
```

```
# 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
* 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
* Acceptable values are any valid field name, any field name including the * character, or * (e
* 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 app
* Acceptable values are any valid eventtype name, or any eventtype name plus the * character (e

display_location = <string>
* Dictates whether to display the workflow action in the event menu, the field menus or in both
* Accepts field_menu, event_menu, or both.
* Defaults to both.

###############################################################################################
# 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 o
# valid field to be substituted into the final string.
#
# For example, you might construct a Google search using an error message field called error_ms
# 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 and


#################################################################################
# Field action types
#################################################################################

#################################################################################
# Link type:
# Allows for the construction of GET and POST requests via links to external resources.
#################################################################################

[REQUIRED]
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.

[OPTIONAL]
link.target = <string>
* Determines if clicking the link opens a new window, or redirects the current window to the re
* 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 resour
* Accepts: "get" or "post".
* Defaults to "get".

link.postargs = 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.
#################################################################################

[REQUIRED]
```

```
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 brake quoting or other search

[OPTIONAL]
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 use
* Defaults to false.
```

workflow_actions.conf.example

```
# Copyright (C) 2005-2010 Splunk Inc.  All Rights Reserved.  Version 4.1.5
#
# This is an example workflow_actions.conf.  These settings are used to create workflow actions
# Refer to workflow_actions.conf.spec and the documentation at splunk.com for more information
#
# To use one or more of these configurations, copy the configuration block into workflow_actior
# 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://www.splunk.com/base/Documentation/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 requ
# 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
```

Troubleshooting

Splunk log files

Splunk log files

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

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

Search these files in Splunk Web by typing:

```
index::_internal
```

Internal logs

Here is a list with descriptions of the internal logs in `$SPLUNK_HOME/var/log/splunk`. Splunk's internal logs are useful for troubleshooting or metric analysis.

audit.log

Log of audit events.

crawl.log

Log of crawl activities.

inputs.log

license_audit.log

Continuous audit of license violations.

metrics.log

Contains information about CPU usage and Splunk's data processing. The metrics.log file is a sampling of the top ten items in each category for in 30-second intervals, based on the size of _raw. It can be used for limited analysis of volume trends for data inputs. For more information about what's in metrics.log, refer to Work with metrics.log as well as this developer blog post about Splunk forwarder and indexer metrics.

migration.log

A log of events during install and migration. Specifies which files were altered during upgrade.

python.log

A log of python events within Splunk. Useful for debugging REST endpoints and communication with splunkd.

searches.log

A log of all searches performed on the server since installation or the most recent `splunk clean` command.

splunkd_stdout.log

The Unix standard output device for the server.

splunkd_stderr.log

The Unix standard error device for the server.

splunklogger.log

A subset of the Splunk server's own log events since installation or the most recent `splunk clean` command. This file is sent to `index::splunklogger` and can be searched through Splunk Web.

splunkd.log

A record of actions made by the Splunk server. May be requested by Splunk Support for troubleshooting purposes.

splunkmon.log

Log of splunk's watchdog process. **Note:** The watchdog process and command are deprecated and should not be used.

web_access.log

A record of actions made by Splunk Web, in an Apache access_log format.

web_service.log

A record of actions made by Splunk Web.

wmi.log

Only relevant on Windows machines. Logs attempts from Splunk to connect to WMI.

debug

Splunk has a debugging parameter (`--debug`) you can add when starting Splunk from the CLI (with `./splunk start`).

```
./splunk start --debug
```

Note: Navigate to Splunk's CLI `$SPLUNK_HOME/bin` and use the `./splunk` command. This command outputs logs to `$SPLUNK_HOME/var/log/splunk/splunkd.log`. To turn off debugging, stop or restart Splunk.

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

log.cfg

For more granular debugging messages, you can change log levels by editing `$SPLUNK_HOME/etc/log.cfg`. This affects Splunk's internal logs.

You can change the following categories in `log.cfg`. Set the category you wish to debug from `WARN` or `INFO` to `DEBUG`.

The message levels, in order from least to most urgent are:

- DEBUG
- INFO
- WARN
- ERROR
- FATAL
- CRIT

```
rootCategory=WARN,A1
category.LicenseManager=INFO
category.TcpOutputProc=INFO
category.TcpInputProc=INFO
category.UDPInputProcessor=INFO
category.SavedSplunker=INFO
category.DistributedMgr=INFO
category.DistributedExecutionContext=INFO
category.DistributedDeploymentProcessor=INFO
category.DistributedDeploymentClientProcessor=INFO
category.DistributedDeploymentClientMgr=INFO
category.DistributedDeploymentMgr=INFO
category.ThruputProcessor=WARN
category.ShutdownHandler=WARN
# leave loader at INFO!  this is what gives us our build + system info...
category.loader=INFO
category.ulimit=INFO
category.SearchPerformance=INFO
category.SearchPipelinePerformance=WARN
```

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

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

If you change this file, restart Splunk.

log-local.cfg

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

Work with metrics.log

Work with metrics.log

Splunk's internal `metrics.log` file is a sampling of the top ten items in each category for in 30-second intervals, based on the size of `_raw`. It does not give you an exact accounting of all your inputs, just the top 10 hot data sources. You can examine its contents for limited analysis of volume trends for data inputs. It is different from the numbers reported by LicenseManager, which include the indexed fields. Also, the default configuration only maintains the metrics data in the internal index a few days, but by going to the files you can see trends over a period of months if your rolled files go that far back.

Note: You can change the number of series that `metrics.log` tracks from the default of 10 by editing the value of `maxseries` in the `[metrics]` stanza in `limits.conf`.

You can find more information about `metrics.log` in this developer blog posting about forwarder and indexer metrics.

Use metrics log to troubleshoot issues with data inputs

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

A typical metrics.log has stuff like this:

```
03-13-2008 10:48:55.620 INFO Metrics - group=pipeline, name=tail, processor=tail, cpu_seconds=0
03-13-2008 10:48:55.620 INFO Metrics - group=pipeline, name=typing, processor=annotator, cpu_se
03-13-2008 10:48:55.620 INFO Metrics - group=pipeline, name=typing, processor=clusterer, cpu_se
03-13-2008 10:48:55.620 INFO Metrics - group=pipeline, name=typing, processor=readerin, cpu_sec
03-13-2008 10:48:55.620 INFO Metrics - group=pipeline, name=typing, processor=sendout, cpu_seco
03-13-2008 10:48:55.620 INFO Metrics - group=thruput, name=index_thruput, instantaneous_kbps=0.
03-13-2008 10:48:55.620 INFO Metrics - group=per_host_thruput, series="fthost", kbps=0.019563,
03-13-2008 10:48:55.620 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.283203,
03-13-2008 10:48:55.620 INFO Metrics - group=per_index_thruput, series="_internal", kbps=0.2753
03-13-2008 10:48:55.620 INFO Metrics - group=per_index_thruput, series="_thefishbucket", kbps=0
03-13-2008 10:48:55.620 INFO Metrics - group=per_index_thruput, series="default", kbps=0.007876
03-13-2008 10:48:55.620 INFO Metrics - group=per_source_thruput, series="/applications/splunk3.
03-13-2008 10:48:55.620 INFO Metrics - group=per_source_thruput, series="/applications/splunk3.
03-13-2008 10:48:55.620 INFO Metrics - group=per_source_thruput, series="/var/log/apache2/somec
03-13-2008 10:48:55.620 INFO Metrics - group=per_source_thruput, series="filetracker", kbps=0.0
03-13-2008 10:48:55.620 INFO Metrics - group=per_sourcetype_thruput, series="access_common", kb
03-13-2008 10:48:55.620 INFO Metrics - group=per_sourcetype_thruput, series="filetrackercrclog"
03-13-2008 10:48:55.620 INFO Metrics - group=per_sourcetype_thruput, series="splunkd", kbps=0.2
03-13-2008 10:48:55.620 INFO Metrics - group=queue, name=aeq, max_size=10, filled_count=0, empt
```

```
03-13-2008 10:48:55.620 INFO Metrics - group=queue, name=aq, max_size=10, filled_count=0, empty
03-13-2008 10:48:55.620 INFO Metrics - group=queue, name=tailingq, current_size=0, largest_size
03-13-2008 10:48:55.620 INFO Metrics - group=queue, name=udp_queue, max_size=1000, filled_count
```

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

`group` identifies what type of thing is being reported on and `series` gives the particular item.

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

```
03-13-2008 10:49:57.634 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.245401,
03-13-2008 10:50:28.642 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.237053,
03-13-2008 10:50:59.648 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.217584,
03-13-2008 10:51:30.656 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.245621,
03-13-2008 10:52:01.661 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.311051,
03-13-2008 10:52:32.669 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.296938,
03-13-2008 10:53:03.677 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.261593,
03-13-2008 10:53:34.686 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.263136,
03-13-2008 10:54:05.692 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.261530,
03-13-2008 10:54:36.699 INFO Metrics - group=per_host_thruput, series="grumpy", kbps=0.313855,
```

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

```
03-13-2008 10:51:30.656 INFO Metrics - group=per_sourcetype_thruput, series="access_common", kb
03-13-2008 10:52:01.661 INFO Metrics - group=per_sourcetype_thruput, series="access_common", kb
03-13-2008 10:52:32.670 INFO Metrics - group=per_sourcetype_thruput, series="access_common", kb
03-13-2008 10:53:34.686 INFO Metrics - group=per_sourcetype_thruput, series="access_common", kb
03-13-2008 10:54:36.700 INFO Metrics - group=per_sourcetype_thruput, series="access_common", kb
03-13-2008 10:56:09.722 INFO Metrics - group=per_sourcetype_thruput, series="access_common", kb
03-13-2008 10:56:40.730 INFO Metrics - group=per_sourcetype_thruput, series="access_common", kb
03-13-2008 10:57:11.736 INFO Metrics - group=per_sourcetype_thruput, series="access_common", kb
03-13-2008 10:58:13.748 INFO Metrics - group=per_sourcetype_thruput, series="access_common", kb
```

But you have probably got more than 10 sourcetypes, so at any particular time some other one could spike and `access_common` wouldn't be reported. `per_index_thruput` and `per_source_thruput` work similarly.

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

```
index::_internal metrics group=per_index_thruput NOT debug NOT sourcetype::splunk_web_access |
```

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

```
sum of kb vs. time for results in the past day
_time totalKB
1 03/12/2008 11:00:00 922.466802
2 03/12/2008 12:00:00 1144.674811
```

```
3 03/12/2008 13:00:00 1074.541995
4 03/12/2008 14:00:00 2695.178730
5 03/12/2008 15:00:00 1032.747082
6 03/12/2008 16:00:00 898.662123
```

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

```
index::_internal metrics grumpy group=per_host_thruput | timechart fixedrange=t span=1h sum(kb)
sum of kb vs. time for results in the past day
_time totalKB
1 03/12/2008 11:00:00 746.471681
2 03/12/2008 12:00:00 988.568358
3 03/12/2008 13:00:00 936.092772
4 03/12/2008 14:00:00 2529.226566
5 03/12/2008 15:00:00 914.945313
6 03/12/2008 16:00:00 825.353518
index::_internal metrics access_common group=per_sourcetype_thruput | timechart fixedrange=t sp
sum of kb vs. time for results in the past day
_time totalKB
1 03/12/2008 11:00:00 65.696285
2 03/12/2008 12:00:00 112.035162
3 03/12/2008 13:00:00 59.775395
4 03/12/2008 14:00:00 35.008788
5 03/12/2008 15:00:00 62.478514
6 03/12/2008 16:00:00 14.173828
```

Contact Support

Contact Support

For contact information, see the main Support contact page.

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

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

diag

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

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

From `$SPLUNK_HOME/bin` run

UNIX:

```
./splunk diag
```

Windows:

```
splunk diag
```

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

```
./splunk cmd python ../lib/python2.6/site-packages/splunk/clilib/info_gather.py
```

This produces `diag-<server name>-<date>.tar.gz` (or .zip) that you can send to Splunk Support for troubleshooting.

Note: Before you upload, please make sure the user who uploads the file has read permissions to the diag*.tar.gz file.
Upload your diag output to your Support case here -

- Enterprise Support: http://www.splunk.com/index.php/track_issues
- Community Members: http://www.splunk.com/index.php/send_to_splunk

Diag options

- File exclusion

Diag can be told to leave some files out of the diag with the switch --exclude, for example:

```
splunk diag --exclude "*/etc/auth/splunk.secret"
```

Defaults can also be controlled in server.conf. Refer to server.conf.spec for more information.

Log levels and starting in debug mode

Splunk logging levels can be changed to provide more detail for different features either from within Splunk Manager, or by editing `$SPLUNK_HOME/var/log/splunk/splunkd.log`.

Splunk's logging levels are `DEBUG INFO NOTICE WARN ERROR CRIT ALERT FATAL EMERG` (most to least verbose). If a default level is not specified for a category the logging level defaults to your rootCategory setting. The easiest way to increase the verbosity of the log is to enable all messages with the --debug option. This does impact performance and should not be used routinely.

Change logging levels for individual subsystems from within Manager

- Log into Splunk Web as a user with **admin** role privileges.
- Click **Manager > System logging**.
- Select the log channel or channels you are interested in and adjust the logging level as desired.
- Click **Save**.

Enable debug logging for all logs by restarting Splunk in debug mode

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

Specific areas can be enabled to collect debugging details over a longer period with minimal performance impact. See the category settings in the file `$SPLUNK_HOME/etc/log.cfg` to set specific log levels without enabling a large number of categories as with --debug. Restart Splunk after changing this file. **Note:** Not all messages marked WARN or ERROR indicate actual problems with Splunk; some indicate that a feature is not being used.

Enable debug messages in splunkd.log dynamically with a search (4.1.3 and earlier versions)

To enable debugging, execute the following search

```
| debug cmd=logchange param1=root param2=DEBUG
```

To return to the default log level, execute the following search:

```
| debug cmd=logchange param1=root param2=WARN
```

To set a particular category of messages, replace "root" with the desired category. This does not change any settings in `log.cfg`. On restart, the log level reverts to what is defined in `log.cfg`.

Note This search will return a "Error in 'DebugCommand': setting root priority" message. This is not an error and is normal. It is posted as an error message to ensure it is logged at any debug level.

Enable debug messages from the CLI (4.1.4 and later versions)

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

Note This search will return a "HTTP Status: 200" message. This is not an error and is normal.

For investigating problems monitoring files, use the FileInputTracker and selectProcessor categories. These are not enabled with the normal "--debug" option because they are very verbose.

Debug Splunk Web

Change the logging level for Splunkweb by editing the file:

```
$SPLUNK_HOME/etc/log.cfg
```

or if you have created your own

```
$SPLUNK_HOME/etc/log-local.cfg
```

Locate the `[python]` stanza and change the contents to:

```
[python]
splunk = DEBUG
# other lines should be removed
```

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

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

Core Files

To collect a core file, use *ulimit* to remove any maximum file size setting before starting Splunk.

```
# ulimit -c unlimited

# splunk restart
```

This setting only affects the processes you start in a particular shell, so you may wish to do it in a new session. For Linux, start Splunk with the --nodaemon option (`splunk start --nodaemon`). In another shell, start the web interface manually with `splunk start splunkweb`.

Depending on your system, the core may be named something like core.1234, where the number indicates the process id and be the same location as the splunkd executable.

LDAP configurations

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

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

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

LDAP configurations

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

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

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

Recover metadata for a corrupt Splunk index directory

Important: You must contact Splunk support for direction before using this command.

The `recover-metadata` command recovers missing or corrupt metadata associated with any Splunk index directory, sometimes also referred to as a 'bucket'. If your Splunk instance will not start up, one possible diagnosis is that one or more of your index buckets is corrupt in some way. Contact support; they will help you determine if this is indeed the case and if so, which bucket(s) are affected. Then, run this command:

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

Splunk will return a success or failure message.

Anonymize data samples to send to support

Anonymize data samples to send to support

Splunk contains an anonymize function. The anonymizer combs through sample log files or event files to replace identifying data - usernames, IP addresses, domain names, etc. - with fictional values that maintain the same word length, and event type. For example, it may turn the string `user=carol@adalberto.com` into `user=plums@wonderful.com`. This lets Splunk users share log data without revealing confidential or personal information from their networks.

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

You can anonymize files from Splunk's CLI. To use Splunk's CLI, navigate to the `$SPLUNK_HOME/bin/` directory and use the `./splunk` command.

Simple method

The easiest way to anonymize a file is with the anonymizer tool's defaults, as shown in the session below. Note that you currently need to have `$SPLUNK_HOME/bin` as your current working directory; this will be fixed in an incremental release.

From the CLI, type the following:

```
# ./splunk anonymize file -source /path/to/[filename]
```

```
# cp -p /var/log/messages /tmp
# cd $SPLUNK_HOME/bin
# splunk anonymize file -source /tmp/messages
Getting timestamp from: /opt/paul207/splunk/lib/python2.4/site-packages/splunk/timestamp.config
Processing files: ['/tmp/messages']
Getting named entities
        Processing /tmp/messages
Adding named entities to list of public terms: Set(['secErrStr', 'MD_SB_DISKS', 'TTY', 'target'
        Processing /tmp/messages for terms.
        Calculating replacements for 4672 terms.
==================================================
```

```
Wrote dictionary scrubbed terms with replacements to "/tmp/INFO-mapping.txt"
Wrote suggestions for dictionary to "/tmp/INFO-suggestions.txt"
================================================
Writing out /tmp/ANON-messages
Done.
```

Advanced method

You can customize the anonymizer by telling it what terms to anonymize, what terms to leave alone, and what terms to use as replacements. The advanced form of the command is shown below.

```
# ./splunk anonymize file -source <filename> [-public_terms <file>] [-private_terms <file>] [-r
```

- `filename`
  - Default: `None`
  - Path and name of the file to anonymize.
- `public_terms`
  - Default: `$SPLUNK_HOME/etc/anonymizer/public-terms.txt`
  - A list of locally-used words that will not be anonymized if they are in the file. It serves as an appendix to the `dictionary` file.
  - Here is a sample entry:

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

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

```
481-51-6234
passw0rd
```

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

```
charlie
claire
desmond
jack
```

- `dictionary`
- Default: `$SPLUNK_HOME/etc/anonymizer/dictionary.txt`

- A global list of common words that will not be anonymized, unless overridden by entries in the `private_terms` file.
- Here is a sample entry:

```
algol
ansi
arco
arpa
arpanet
ascii
```

- `timestamp_config`
  - Default: `$SPLUNK_HOME/etc/anonymizer/anonymizer-time.ini`
  - Splunk's built-in file that determines how timestamps are parsed.

Output Files

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

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

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

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

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

Not finding the events you're looking for?

Not finding the events you're looking for?

If you're searching for events and not finding them or looking at a dashboard and seeing "No result data", there could be a couple of reasons why:

Are you running Splunk Free?

Splunk Free does not support scheduled saved searches or summary indexing. If you're in an app that uses search artifacts created by scheduled searches (for example by including them with the HiddenSavedSearch module), those searches will be run on-demand when you view the dashboard. That may result in longer load times than you experienced in the Enterprise or Enterprise Trial versions.

If an app uses summary indexes, however, the summary index(es) will not be updated in a Free version of Splunk because the job scheduler is unavailable. If an app does not already have an alternative Free view defined, you may see "No Results" in dashboards that were relying on summary indexes.

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

Review this topic about object ownership and this topic about configuration file precedence for information about where Splunk writes knowledge objects such as saved searches.

Learn more about Splunk Free

Was the data added under a different app?

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 that ship with Splunk, write input data to a specific index (in the case of *Nix and Windows, that is the 'os' index). If you're not finding data that you're certain is in Splunk, be sure that you're looking at the right index. You may want to add the 'os' index to the list of default indexes for the role you're using. For more information about roles, refer to the topic about roles in this manual.

SuSE Linux: unable to get a properly formatted response from the server

SuSE Linux: unable to get a properly formatted response from the server

Users running Splunk on a SuSE server may see the error message `Unable to get a properly formatted response from the server; canceling the current search` when executing a search. Alteratively, the dashboard just won't display properly. To resolve this issue, edit `/etc/mime.types`. Delete (or comment out) these 2 lines:

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

Also change this line:

```
text/xml xml
```

to:

```
text/xml xml xsl
```

With these changes in place, restart Splunk and clear your browser cache.

Note: If you are using a proxy, you will need to flush that as well.

Command line tools for use with Support's direction

Command line tools for use with Support's direction

Caution: DO NOT use these commands without consulting Splunk Support first.

cmd

btool

Cmd line modification and listing of bundles.

Syntax

Add

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

Delete

./splunk cmd btool --app=*app_name* --user=*user_name conf_file_prefix* **delete** *stanza_name* [*attribute_name*]

List

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

btprobe

Queries the fishbucket for file records stored by tailing.

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

There are 2 possible ways to invoke this tool:

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

This method will query the specified BTree for the specified record. You can specify a crc directly or just input a file and crc will be computed from it.

- If you specify --validate, it will run a btree validation to look for errors.
- If you specify --salt, it will salt the crc in the case that --file param is specified

2. <code>btprobe [-h or --help] --compute-crc <filename> [--salt <salt>]

This method will compute a crc from the specified file, and salt it if --salt specified

- Example: btprobe -d /opt/splunk/var/lib/splunk/fishbucket/splunk_private_db -k 0xe8d117ddba85e714 --validate
- Example: btprobe -d /opt/splunk/var/lib/splunk/fishbucket/splunk_private_db --file /var/log/inputfile --salt SOME_SALT
- Example: btprobe --compute-crc /var/log/inputfile --salt SOME_SALT

classify

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

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

gzdumper

locktest

locktool

./splunk cmd locktool

Usage :

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

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

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

parsetest

pcregextest

regextest

searchtest

signtool

Sign

./splunk cmd signtool [*-s* | *--sign*] [*<dir to sign>*]

Verify

./splunk cmd signtool [*-v* | *--verify*] [*<dir to verify>*]

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

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

tsidxprobe

This will take a look at your index files (.tsidx) and verify that they meet the necessary format requirements. It should also identify any files that are potentially causing a problem

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

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

1. for i in `find $SPLUNK_DB | grep tsidx`; do tsidxprobe $i >> tsidxprobeout.txt; done

(If you've changed the default datastore path, then this should be in the new location.)

The file tsidxprobeout.txt will contain the results from your index files. You should be able to gzip this and attach it to an email and send it to Splunk Support.

Troubleshooting configurations

Troubleshooting configurations

Splunk's configuration file system supports many overlapping configuration files in many different locations. How these configuration files interact with and take precedence over one another is described in "Configuration file precedence" in this manual. Sometimes the price of this level of flexibility is that figuring out which value for which configuration option is being used in your Splunk installation.

Splunk provides a command line tool you can use to help troubleshoot issues with your configuration files, or just see what values are being used by your Splunk installation.

The command line tool is called btool. You can run btool for a given app in your Splunk installation, and it will list out all the configuration options that are currently in use by that app for a given configuration file.

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

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

where *app_name* is the name of the app you want to see the configurations for and *prefix* is the name of the config file you're interested in without the .conf extension. `list` indicates that you want to list the options. btool supports changing configuration files as well, but Splunk does not recommend you use btool this way without discussing it with the Splunk Support team.

So for example, if you want to know what configuration options are being used in props.conf by the Search app, you'd specify the Search app and props.conf in your btool string like so:

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

and would see a list of the props.conf settings currently being used for the Search app.

You can then save this list to a file and examine it.

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

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

Note: btool is not tested by Splunk and is not officially supported or guaranteed. That said, this is what our Support team uses when trying to troubleshoot your issues.