# Table of Contents

**Introduction**
- What data can I index? ................................................................. 1
- Get started with getting data in .................................................. 4
- Is my data local or remote? ............................................................ 5
- Use forwarders to get data in ...................................................... 7
- Use apps to get data in ............................................................... 9
- Configure your inputs ................................................................ 9
- How Splunk Enterprise handles your data ................................. 12

**How to get data into your Splunk deployment**
- How do you want to add data? .................................................... 16
- Upload data .................................................................................. 17
- Monitor data ................................................................................ 18
- Forward data ................................................................................ 19
- The Set Source Type page ............................................................ 22
- Prepare your data for previewing ............................................... 28
- Modify event processing .............................................................. 29
- Modify input settings .................................................................. 34
- Distribute source type configurations in Splunk Enterprise ......... 36

**Get data from files and directories**
- Monitor files and directories ....................................................... 39
- Monitor files and directories with Splunk Web ........................... 42
- Monitor files and directories with the CLI ................................. 44
- Monitor files and directories with inputs.conf .......................... 48
- Specify input paths with wildcards ............................................ 57
- Whitelist or blacklist specific incoming data ............................. 61
- How Splunk Enterprise handles log file rotation ....................... 64

**Get data from network sources**
- Get data from TCP and UDP ports ............................................ 66
- How Splunk Enterprise handles syslog data over UDP .............. 76
- Send SNMP events to your Splunk deployment ......................... 81

**Get Windows data**
- Monitoring Windows data with Splunk Enterprise .................. 84
- How to get Windows data into your Splunk deployment ............ 85
- Considerations for deciding how to monitor remote Windows data 87
- Monitor Active Directory ............................................................ 93
# Table of Contents

## Get Windows data
- Monitor Windows event log data ............................................................ 105
- Monitor file system changes ................................................................. 132
- Monitor data through Windows Management Instrumentation (WMI) .... 138
- Monitor Windows Registry data ............................................................ 150
- Monitor Windows performance ............................................................ 158
- Monitor Windows data with PowerShell scripts ...................................... 179
- Monitor Windows host information ........................................................ 184
- Monitor Windows printer information ..................................................... 190
- Monitor Windows network information .................................................. 195

## Get data with HTTP Event Collector ...................................................... 204
- Set up and use HTTP Event Collector in Splunk Web ............................. 204
- Set up and use HTTP Event Collector with configuration files ................. 219
- Set up and use HTTP Event Collector from the CLI ............................... 224
- Use cURL to manage HTTP Event Collector tokens, events, and services ..................................................................................... 227
- About HTTP Event Collector Indexer Acknowledgment ......................... 233
- Scale HTTP Event Collector with distributed deployments .................... 240
- Format events for HTTP Event Collector ............................................... 243
- Automate indexed field extractions with HTTP Event Collector ............... 250
- Send metrics to a metrics index ............................................................ 252
- HTTP Event Collector REST API endpoints ....................................... 254
- HTTP Event Collector Examples .......................................................... 256
- Troubleshoot HTTP Event Collector ..................................................... 258

## Get other kinds of data in .................................................................... 270
- Monitor First In, First Out (FIFO) queues ............................................. 270
- Monitor changes to your file system ..................................................... 272
- Get data from APIs and other remote data interfaces through scripted inputs ..................................................................................... 280

## Configure event processing ................................................................. 289
- Overview of event processing ............................................................. 289
- Configure character set encoding ......................................................... 289
- Configure event line breaking ............................................................. 295
- Configure event timestamps .............................................................. 300
- Configure indexed field extraction ..................................................... 301
- Anonymize data .................................................................................. 301
Table of Contents

Configure timestamps

- How timestamp assignment works.........................................................310
- Configure timestamp recognition..........................................................310
- Configure timestamp assignment for events with multiple timestamps...326
- Configure advanced timestamp recognition with datetime.xml............328
- Specify time zones for timestamps.........................................................335
- Tune timestamp recognition for better indexing performance..............338

Configure indexed field extraction

- About indexed field extraction..............................................................340
- About default fields (host, source, sourcetype, and more).....................341
- Assign default fields dynamically.........................................................344
- Create custom fields at index time........................................................345
- Extract fields from files with structured data......................................356
- Process events with ingest-time eval....................................................367

Configure host values

- About hosts...........................................................................................371
- Set a default host for a Splunk instance...............................................373
- Set a default host for a file or directory input.......................................375
- Set host values based on event data......................................................381
- Change host values after indexing.......................................................383

Configure source types

- Why source types matter......................................................................384
- Override automatic source type assignment........................................388
- Configure rule-based source type recognition.....................................392
- List of pretrained source types...............................................................394
- Override source types on a per-event basis.........................................399
- Create source types...............................................................................402
- Manage source types............................................................................404
- Rename source types at search time....................................................410

Manage event segmentation

- About event segmentation.....................................................................412
- Set the segmentation for event data.....................................................414
- Set search-time event segmentation in Splunk Web.........................417
# Table of Contents

*Improve the data input process* ..............................................................................................................418  
  - Use a test index to test your inputs.................................................................................................418  
  - Use persistent queues to help prevent data loss...........................................................................420  
  - Troubleshoot the input process.......................................................................................................422  
  - Resolve data quality issues..............................................................................................................423
Introduction

What data can I index?

Splunk Enterprise can index any kind of data. In particular, any and all IT streaming, machine, and historical data, such as Windows event logs, web server logs, live application logs, network feeds, **metrics**, change monitoring, message queues, archive files, and so on.

How do I get data in?

To get data into your Splunk deployment, point it at a data source. Tell it a bit about the source. That source then becomes a data input. Splunk Enterprise indexes the data stream and transforms it into a series of events. You can view and search those events right away. If the results aren't exactly what you want, you can tweak the indexing process until they are.

If you have Splunk Enterprise, the data can be on the same machine as an **indexer (local data)** or on another machine (**remote data**). If you have Splunk Cloud, the data resides in your corporate network and you send it to your Splunk Cloud deployment. You can get remote data into your Splunk deployment using network feeds or by installing Splunk **forwarders** on the hosts where the data originates. For more information on local vs. remote data, see Where is my data?

Splunk offers **apps** and **add-ons**, with pre-configured inputs for things like Windows- or Linux-specific data sources, Cisco security data, Symantec Blue Coat data, and so on. Look on Splunkbase for an app or add-on that fits your needs. Splunk Enterprise also comes with dozens of recipes for data sources like web server logs, Java 2 Platform, Enterprise Edition (J2EE) logs, or Windows performance metrics. You can get to these from the **Add data** page in Splunk Web. If the recipes and apps don't cover your needs, then you can use the general input configuration capabilities to specify your particular data source.

For more information on how to configure data inputs, see Configure your inputs.

**Guided Data Onboarding**

The Guided Data Onboarding (GDO) feature also provides end-to-end guidance for getting select data sources into specific Splunk platform deployments.
From your home page in Splunk Web, find the data onboarding guides by clicking Add Data. From there you can select a data source and configuration type. Then view diagrams, high-level steps, and documentation links that help you set up and configure your data source.

You can find all the Guided Data Onboarding manuals by clicking the Add data tab on the Splunk Enterprise Documentation site.

Types of data sources

Splunk provides tools to configure many kinds of data inputs, including those that are specific to particular application needs. Splunk also provides the tools to configure any arbitrary data input types. In general, you can categorize Splunk inputs as follows:

- Files and directories
- Network events
- Windows sources
- Other sources

Files and directories

A lot of data comes directly from files and directories. You can use the files and directories monitor input processor to get data from files and directories.

To monitor files and directories, see Get data from files and directories.

Network events

Splunk Enterprise can index data from any network port, for example, remote data from syslog-ng or any other application that transmits over the TCP protocol. It can also index UDP data, but you should use TCP instead whenever possible for enhanced reliability.

Splunk Enterprise can also receive and index SNMP events, alerts fired off by remote devices.

To get data from network ports, see Get data from TCP and UDP ports in this manual.

To get SNMP data, see Send SNMP events to your Splunk deployment in this manual.
Windows sources

Splunk Cloud and the Windows version of Splunk Enterprise accept a wide range of Windows-specific inputs. Splunk Web lets you configure the following Windows-specific input types:

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

To index and search Windows data on a non-Windows instance of Splunk Enterprise, you must first use a Windows instance to gather the data. See Considerations for deciding how to monitor remote Windows data.

For a more detailed introduction to using Windows data in Splunk Enterprise, see Monitoring Windows data in this manual.

Other data sources

Splunk software also supports other kinds of data sources. For example:

- Metrics
  Get metrics data from your technology infrastructure, security systems, and business applications.

- First-in, first-out (FIFO) queues

- Scripted inputs
  Get data from APIs and other remote data interfaces and message queues.

- Modular inputs
  Define a custom input capability to extend the Splunk Enterprise framework.

- The HTTP Event Collector endpoint
  Use the HTTP Event Collector to get data directly from a source with the HTTP or HTTPS protocols.
Get started with getting data in

To get started with getting data into your Splunk deployment, point it at some data by configuring an input. There are several ways to do this. The easiest way is to use Splunk Web.

Alternatively, you can download and enable an app, such as the Splunk App for Microsoft Exchange or Splunk IT Service Intelligence.

After you configure the inputs or enable an app, your Splunk deployment stores and processes the specified data. You can go to either the Search app or the main app page and begin exploring the data that you collected.

- To learn how to configure an input, see Configure your inputs.
- To learn how to add data to your Splunk deployment, see How do you want to add data?.
- To learn how to experiment with adding a test index, see Use a test index.
- To learn about how to add source types, see "The Set Sourcetype page."
- To learn what event processing is and how to configure it, see How Splunk software handles your data.
- To learn how to delete data from your Splunk deployment, see Delete indexed data and start over.
- To learn about how to configure your inputs with a default index, see Point your inputs at the default index.

Add new inputs

Here is a high-level procedure for adding data.

1. Understand your needs. Ask the following questions.
   ♦ What kind of data do I want to index? See What data can I index?.
   ♦ Is there an app for that? See Use apps to get data in.
   ♦ Where does the data reside? Is it local or remote? See Where is my data?
   ♦ Should I use forwarders to access remote data? See Use forwarders to get data in.
   ♦ What do I want to do with the indexed data? See What is Splunk knowledge? in the Knowledge Manager Manual.

2. Create a test index and add a few inputs. Any data you add to your test index counts against your maximum daily indexing volume for licensing purposes.
3. Preview and modify how your data will be indexed before committing the data to the test index.
4. Review the test data that you have added with the Search app:
   ♦ Do you see the sort of data you were expecting?
   ♦ Did the default configurations work well for your events?
   ♦ Is data missing or mangled?
   ♦ Are the results optimal?
5. If necessary, tweak your input and event processing configurations further until events look the way you want them to.
6. Delete the data from your test index and start over, if necessary.
7. When you are ready to index the data permanently, configure then inputs to use the default main index.

You can repeat this task to add other inputs as you familiarize yourself with the getting data in process.

**Index custom data**

Splunk software can index any time-series data, usually without additional configuration. If you have logs from a custom application or device, process it with the default configuration first. If you do not get the results you want, you can tweak things to make sure the software indexes your events correctly.

See [Overview of event processing](#) and [How indexing works](#) so that you can make decisions about how to make Splunk software work with your data. Consider the following scenarios for collecting data.

- Are the events in your data more than one line? See [Configure event line breaking](#).
- Is your data in an unusual character set? See [Configure character set encoding](#).
- Is the Splunk software unable to determine the timestamps correctly? See [How timestamp assignment works](#).

**Is my data local or remote?**

If you have Splunk Cloud or run Splunk Enterprise in the cloud, all indexed data is remote. If you have an on-premises Splunk Enterprise deployment, the answer to this question depends on a number of things, which include:

- The operating system on which your Splunk Enterprise instance resides.
Where the data is physically.
• The types of data storage that are connected to the Splunk Enterprise instance.
• Whether or not you need to perform any authentication or other intermediate to access the data store that contains the data you want to index.

**Local Data**

A local resource is a fixed resource that your Splunk Enterprise instance has direct access to. You are able to access a local resource, and whatever it contains, without having to attach, connect, or perform any other intermediate action (such as authentication or mapping a network drive). If your data is on such a resource, the data is considered local.

Some examples of local data include:

• Data on a hard disk or solid state drive installed in a desktop, laptop, or server host.
• Data on a resource that has been permanently mounted over a high-bandwidth physical connection that the host can access at boot time.
• Data on a RAM disk.

**Remote Data**

A remote resource is any resource that does not meet the definition of a "local" resource. Data that exists on such a resource is remote data. Some examples of remote resources are:

• Network drives on Windows hosts.
• Active Directory schemas.
• NFS or other network-based mounts on *nix hosts.
• Most cloud-based resources.

**Exceptions**

Some cases where resources might be considered remote are actually not remote. Here are some examples.

• A host has a volume that has been permanently mounted over a high-bandwidth physical connection such as USB or FireWire. Because the computer can mount the resource at boot time, Splunk Enterprise
treats it as a local resource, even though the resource can theoretically be disconnected at a later time.

- A host has a resource that has been permanently mounted over a high-bandwidth network standard such as iSCSI, or to a Storage Area Network over fiber. As the standard treats such volumes as local block devices, such a resource would be considered local.

Use forwarders to get data in

Splunk **forwarders** consume data and send it to an indexer. Forwarders require minimal resources and have little impact on performance, so they can usually reside on the machines where the data originates.

For example, if you have a number of Apache Web servers that generate data that you want to search centrally, you can set up forwarders on the Apache hosts. The forwarders take the Apache data and send it to your Splunk deployment for indexing, which consolidates, stores, and makes the data available for searching. Because of their reduced resource footprint, forwarders have minimal performance impact on the Apache servers.

Similarly, you can install forwarders on your employees' Windows desktops. These can send logs and other data to your Splunk deployment, where you can view the data as a whole to track malware or other issues. The Splunk App for Windows Infrastructure relies on this kind of deployment.

What forwarders do

Forwarders get data from remote machines. They represent a more robust solution than raw network feeds, with their capabilities for the following actions:

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

Forwarders usually do not index the data, but rather forward the data to a Splunk deployment that does the indexing and searching. A Splunk deployment can process data that comes from many forwarders. For detailed information on forwarders, see the *Forwarding Data* or *Universal Forwarder* manuals.
In most Splunk deployments, forwarders serve as the primary consumers of data. In a large Splunk deployment, you might have hundreds or even thousands of forwarders that consume data and forward for consolidation.

**How to configure forwarder inputs**

The following procedure is a general procedure. See the *Forwarding Data Manual* or the *Universal Forwarder Manual* for details on how to configure forwarding and receiving.

1. Configure a Splunk Enterprise host to receive the data.
2. Determine the kind of forwarder you want to put on the host with the data.
   - You can use a heavy forwarder, which is a full Splunk Enterprise instance with forwarding turned on, or a universal forwarder, which is its own installation package.
   - The type of forwarder you use depends on the performance requirements you have on the host and whether or not you need to transform the data in any way as it comes into Splunk.
3. Download Splunk Enterprise or the universal forwarder for the platform and architecture of the host with the data.
4. Install the forwarder onto the host.
5. Enable forwarding on the host and specify a destination.
6. Configure inputs for the data that you want to collect from the host. You can use Splunk Web if the forwarder is a full Splunk Enterprise instance.
7. Confirm that data from the forwarder arrives at the receiving indexer.

Here are the main ways that you can configure data inputs on a forwarder:

- Specify inputs during initial deployment.
- For Windows forwarders, specify common inputs during the installation process.
- For *nix forwarders, specify inputs directly after installation.
- Use the CLI.
- Edit inputs.conf.
- Install an app that contains the inputs you want.
- Use Splunk Web to configure the inputs and a deployment server to copy the resulting inputs.conf file to forwarders.

**Forwarder Topologies and Deployments**

- For information on forwarders, including use cases, typical topologies, and configurations, see About forwarding and receiving in the *Forwarding Data Manual.*
For details on how to deploy the universal forwarder, including how to use the deployment server to simplify distribution of configuration files and apps to multiple forwarders, see Example forwarder deployment topologies in the *Universal Forwarder* manual.

**Use apps to get data in**

Splunk **apps** and **add-ons** extend the capability and simplify the process of getting data into your Splunk platform deployment. Download apps from Splunkbase.

Apps typically target specific data types and handle everything from configuring the inputs to generating useful views of the data. For example, the Splunk App for Windows Infrastructure provides data inputs, searches, reports, alerts, and dashboards for Windows host management. The Splunk App for Unix and Linux offers the same for Unix and Linux environments. There is a wide range of apps to handle specific types of application data, including the following:

- Splunk DB Connect
- Splunk Stream
- Splunk Add-on for Amazon Web Services

**Further reading for getting and installing apps**

Go to Splunkbase to browse through the large set of apps available for download. Check Splunkbase frequently, because new apps get added all the time.

For more information on apps, see What are apps and add-ons? in the *Admin Manual*. In particular, Where to get more apps and add-ons tells you how to download and install apps.

For information on how to create your own apps, see the *Developing Views and Apps for Splunk Web* manual.

**Configure your inputs**

To add a new type of data to your Splunk deployment, configure a data input. There are a number of ways to configure data inputs:
• **Apps.** Splunk has a variety of apps that offer preconfigured inputs for various data types. For more information, see Use apps to get data in.

• **Splunk Web.** You can configure most inputs using the Splunk Web data input pages. You can access the Add Data landing page from Splunk Home. In addition, when you upload or monitor a file, you can preview and make adjustments to how the file is to be indexed.

• **The Splunk Command Line Interface (CLI).** If you have Splunk Enterprise, you can use the CLI to configure most types of inputs.

• **The inputs.conf configuration file.** When you specify your inputs with Splunk Web or the CLI, the details are saved in a configuration file, inputs.conf. If you have Splunk Enterprise, you can edit that file directly. Some advanced data input needs might require you to edit it.

In addition, if you configure forwarders to send data from outlying machines to a central indexer, you can specify some inputs at installation time. See Use forwarders to get data in.

**Use Splunk Web**

You can add data inputs from Splunk Home or the Settings > Data Inputs menu

- From Splunk Home, select Add Data
- Select Settings > Add data
- Select Settings > Data inputs from the Data section of the Settings pop-up menu.

The Add Data page has options to get data in. Click an icon to go to a page to define the data you want to upload, monitor, or forward.

- Upload
- Monitor
- Forward

For more help on how to use the "Add Data" page, see How do you want to add data?

*How app context determines where Splunk Enterprise writes configuration files*
When you add an input through Splunk Web, Splunk Enterprise adds that input to a copy of `inputs.conf`. The app context, that is, the Splunk app you are currently in when you configure the input, determines where Splunk Enterprise writes the `inputs.conf` file.

For example, if you navigated to the Settings page directly from the Search page and then added an input, Splunk Enterprise adds the input to `$SPLUNK_HOME/etc/apps/search/local/inputs.conf`.

When you add inputs, confirm that you are in the app context that you want to be in. For background on how configuration files work, read About configuration files in the Admin manual.

**Guided Data Onboarding**

The Guided Data Onboarding (GDO) feature also provides end-to-end guidance for getting select data sources into specific Splunk platform deployments.

From your home page in Splunk Web, find the data onboarding guides by clicking Add Data. From there you can select a data source and configuration type. Then view diagrams, high-level steps, and documentation links that help you set up and configure your data source.

You can find all the Guided Data Onboarding manuals by clicking the Add data tab on the Splunk Enterprise Documentation site.

**Use the CLI**

If you have Splunk Enterprise, you can use the Splunk CLI to configure many inputs. From a shell or command prompt, navigate to the `$SPLUNK_HOME/bin/` directory and use the `./splunk` command. For example, the following command adds `/var/log/` as a data input:

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

For more information on the CLI, including how to get command line help, see About the CLI in the Admin manual.

**Edit inputs.conf**

You can edit `inputs.conf` to configure your inputs. You use a text editor to create or modify the file, where you can add a stanza for each input. You can add the stanza to the `inputs.conf` file in `$SPLUNK_HOME/etc/system/local/`, or in your
custom application directory (in $SPLUNK_HOME/etc/apps/<app name>/local).

You can configure the data input by adding key/value pairs to its stanza. You can set multiple settings in an input stanza. If you do not specify a value for a setting, Splunk Enterprise uses the default setting value. Default values for all inputs.conf attributes are in $SPLUNK_HOME/etc/system/default/inputs.conf.

If you have not worked with configuration files, see About configuration files. before starting to add inputs.

**Example inputs.conf stanza**

The following example configuration directs Splunk Enterprise to listen on TCP port 9995 for raw data from any remote host. Splunk Enterprise uses the DNS name of the remote host to set the host of the data. It assigns the source type "log4j" and the source "tcp:9995" to the data.

```
[tcp://:9995]
connection_host = dns
sourcetype = log4j
source = tcp:9995
```

For information on how to configure a specific input, see the topic in this manual for that input. For example, to learn how to configure file inputs, see Monitor files and directories with inputs.conf.

The topic for each data input describes the main attributes available for that input. See the inputs.conf spec file for the complete list of available attributes, including descriptions of the attributes and several examples.

**How Splunk Enterprise handles your data**

Splunk Enterprise consumes data and indexes it, transforming it into searchable knowledge in the form of events. The data pipeline shows the main processes that act on the data during indexing. These processes constitute event processing. After the data is processed into events, you can associate the events with knowledge objects to enhance their usefulness.

**The data pipeline**

Incoming data moves through the data pipeline, which is described in How data moves through Splunk deployments: The data pipeline in the Distributed...
This diagram shows the main steps in the data pipeline.

Deployment Manual.
Event processing

Event processing occurs in two stages, parsing and indexing. All data enters through the parsing pipeline as large chunks. During parsing, Splunk software breaks these chunks into events. It then hands off the events to the indexing pipeline, where final processing occurs.

During both parsing and indexing, Splunk software transforms the data. You can configure most of these processes to adapt them to your needs.

In the parsing pipeline, Splunk software performs a number of actions, including:

- Extracting a set of default fields for each event, including host, source, and sourcetype.
- Configuring character set encoding.
- Identifying line termination using line breaking rules. You can also modify line termination settings interactively, using the "Set Sourcetype" page in Splunk Web.
- Identifying or creating timestamps. At the same time that it processes timestamps, Splunk software identifies event boundaries. You can modify timestamp settings interactively, using the "Set Sourcetype" page.
- Anonymizing data, based on your configuration. You can mask sensitive event data (such as credit card or social security numbers) at this stage.
- Applying custom metadata to incoming events, based on your configuration.

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

- Breaking all events into segments that can then be searched. 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 distinction between parsing and indexing pipelines matters mainly for forwarders. Heavy forwarders can parse data locally and then forward the parsed data on to receiving indexers, where the final indexing occurs. Universal forwarders offer minimal parsing in specific cases such as handling structured data files. Additional parsing occurs on the receiving indexer.
For information about events and what happens to them during the indexing process, see *Overview of event processing* in this manual.

**Enhance and refine events**

After the data has been transformed into events, you can make the events more useful by associating them with knowledge objects, such as event types, field extractions, and reports. For information about managing Splunk knowledge, see the *Knowledge Manager* manual, starting with "What is Splunk knowledge?".
How to get data into your Splunk deployment

How do you want to add data?

The fastest way to add data to your Splunk Enterprise deployment is to use Splunk Web.

The Add Data page

After you log into your Splunk deployment, the Home page appears.

To add data, click Add Data. The Add Data page appears. If your Splunk deployment is a self-service Splunk Cloud deployment, from the system bar, click Settings > Add Data.

There are some conditions where the Add Data page does not appear:

- This instance is part of a search head cluster. See About search head clustering in the Distributed Search manual.
- This instance is a managed Splunk Cloud instance.

There are three options for getting data into your Splunk deployment with Splunk Web: Upload, Monitor, and Forward.

Guided Data Onboarding

The Guided Data Onboarding (GDO) feature also provides end-to-end guidance for getting select data sources into specific Splunk platform deployments.

From your home page in Splunk Web, find the data onboarding guides by clicking Add Data. From there you can select a data source and configuration type. Then view diagrams, high-level steps, and documentation links that help you set up and configure your data source.

You can find all the Guided Data Onboarding manuals by clicking the Add data tab on the Splunk Enterprise Documentation site.
Upload

The Upload option lets you upload a file or archive of files for indexing. When you click Upload, Splunk Web goes to a page that starts the upload process. See Upload data.

Monitor

The Monitor option lets you monitor one or more files, directories, network streams, scripts, Event Logs (on Windows hosts only), performance metrics, or any other type of machine data that the Splunk Enterprise instance has access to. When you click Monitor, Splunk Web loads a page that starts the monitoring process. See Monitor data.

Forward

The Forward option lets you receive data from forwarders into your Splunk deployment. When you click on the "Forward" button, Splunk Web takes you to a page that starts the data collection process from forwarders. See Forward data.

The Forward option requires additional configuration. Use it only in a single-instance Splunk environment.

Upload data

The Upload page lets you specify a file to upload directly to your Splunk Enterprise instance from your computer.

Note: Windows Event Log (.evt) and Windows Event Log XML (.evtx) files that have been exported from another host do not work with the upload feature. This is because those files contain information that is specific to the host that generated them. Other hosts won't be able to process the files in their unaltered form. See Index exported event log (.evt or .evtx) files for more information about the constraints for working with these kinds of files.

The "Upload" page
You can upload data through one of the following methods:

- Drag the file you want to index from your desktop to the "Drop your data file here" area on the page.

or

- In the upper left of the screen, click Select File and select the file that you want to index.

Splunk software then loads the file and processes it, depending on what type of file it is. After it has completed loading, you can then click the green Next button on the upper right to proceed to the next in the "Add data" process.

Next steps

Set sourcetype

Monitor data

You can use the "Monitor data" page in Splunk Web to monitor files and network ports on the host that runs the Splunk Enterprise instance. You access this page from the "Add Data" page, which itself is accessible from the Settings menu in the Splunk Web system bar.

The Monitor page
When you access the "Monitor" page, choose the type of data that you want Splunk Enterprise to monitor. Default inputs are listed first, followed by forwarded inputs, and then any modular inputs that are installed on the instance.

The "Monitor" page shows only the types of data sources that you can monitor, which depends on the type of Splunk deployment you have (Splunk Enterprise or Splunk Cloud) as well as the platform that the Splunk Enterprise instance runs on. See Types of data sources for more information.

**Add a data input**

Some data sources are available only on certain operating systems. For example, Windows data sources only available on hosts that run Windows.

If you experience problems with this procedure, the logged-in Splunk user account might not have permissions to add data or see the data source you want to add.

1. Select a source from the left pane by clicking it once. The page updates based on the source you selected. For example, if you select "Files & Directories", the page updates with a field to enter a file or directory name and specify how Splunk software should monitor the file or directory.
2. Follow the on-screen prompts to complete the selection of the source object that you want to monitor.
3. Click **Next** to proceed to the next step in the Add data process.

**Next Steps**

Set sourcetype

**Forward data**
The "Forward data" page lets you select forwarders that have connected to the Splunk Enterprise instance to configure and send data to the instance. Splunk Web loads this page when you click the Forward button on the Add data page.

This page is available in the following cases:

- You have a single instance of Splunk Enterprise that acts as an indexer and deployment server.
- You have a self-service Splunk Cloud deployment and have configured the universal forwarder as a deployment client.

If you have multiple machines in your Splunk deployment that perform indexing, then this page is not useful. See About deployment server and forwarder management in Updating Splunk Enterprise Instances to learn about the deployment server and how to use it to manage forwarder configurations to send to multiple indexers.

If you have a managed Splunk Cloud deployment, then this page is not available. Instead, you can install a deployment server on-premises to synchronize forwarder configurations so that you do not have to configure forwarders manually.

To determine what type of Splunk Cloud deployment you have, follow the procedures in Types of Splunk Cloud Deployment.

**Prerequisites**

To use the Forward Data page to configure data inputs, you must configure at least one forwarder as a deployment client. If you have not configured a forwarder as a deployment client, the page notifies you that no deployment clients have been found.

To configure a light or heavy forwarder as a deployment client, see Configure deployment clients in the Updating Splunk Enterprise Instances manual.

To configure a universal forwarder as a deployment client, see Configure the universal forwarder as a deployment client in the Universal Forwarder manual. On Windows hosts, you can configure the forwarder as a deployment client during installation.
The Select Forwarders page

When you select "Forward Data" from the "Add Data" page, the following page appears.

You can define server classes and add forwarders to those classes. Server classes are logical groupings of hosts based on things such as architecture or host name.

This page only displays forwarders that you configured to forward data and act as deployment clients to this instance. If you have not configured any forwarders, the page warns you of this.

1. In Select Server Class, click one of the options.
   - New to create a new server class, or if an existing server class does not match the group of forwarders that you want to configure an input for.
   - Existing to use an existing server class.
2. In the Available host(s) pane, choose the forwarders that you want this instance to receive data from. The forwarders move from the Available host(s) pane to the Selected host(s) pane.
   Note: A server class must contain hosts of a certain platform. You cannot, for example, put Windows and *nix hosts in the same server class.
3. (Optional) You can add all of the hosts by clicking the add all link, or remove all hosts by selecting the remove all link.
4. If you chose New in "Select server class", enter a unique name for the server class that you will remember. Otherwise, select the server class you want from the drop-down list.
5. Click Next. The "Select Source" page shows source types that are valid for the forwarders that you selected.
6. Select the data sources that you want the forwarders to send data to this instance.
7. Click Next to proceed to the Set Sourcetype page.

**Next step**

**Modify input settings**

**The Set Source Type page**

The Set Source Type page lets you improve event processing by previewing how your data will be indexed. Use this page to confirm that Splunk Enterprise indexes your data as you want it to appear.

**Preview data prior to indexing**

The Set Source Type page appears after you use the Upload or Monitor pages to specify a single file as a source of data.

On the Set Source Type page, you can make adjustments to how the software indexes your data. You can adjust and improve the indexing process interactively so that when Splunk software indexes and stores the incoming data, the event data ends up in the format that you want.

**Assign the correct source type to your data**

The Set Source Type page helps you apply the correct source type to your incoming data. The source type is one of the default fields that is assigned to all incoming data, and determines how Splunk software formats the data during indexing. By assigning the correct source type to your data, the indexed version of the data (the event data) will look the way you want it to, with correct timestamps and event breaks.

Splunk Enterprise comes with many predefined source types and attempts to assign the correct source type to your data based on its format. In some cases, you might need to manually select a different predefined source type to the data. In other cases, you might need to create a new source type with customized event processing settings.

The page displays how Splunk Enterprise will index the data based on the application of a predefined source type. You can modify the settings interactively...
and save those modifications as a new source type.

Use the Set Source Type page to:

- See what your data will look like without any changes, using the default event-processing configuration.
- Apply a different source type to see whether it offers results more to your liking.
- Modify settings for timestamps and event breaks to improve the quality of the indexed data and save the modifications as a new source type.
- Create a new source type from scratch.

The page saves any new source types to a `props.conf` file that you can later distribute across the indexers in your deployment so that the source types are available globally. See Distribute source type configurations.

For information on source types, see Why source types matter in this manual.

**About the Log to Metrics source type category**

Source types in the Log to Metrics category are used for the ingest-time conversion of log events to metric data points. If you select a source type from this category, a set of Metrics controls will appear on the left side of the Set Source type page. For more information about log-to-metrics conversion and the Metrics settings, see Set up ingest-time log-to-metrics conversion in Splunk Web in Metrics.

When you apply a Log to Metrics source type to an input, the data preview feature is disabled.

**Use the Set Source Type page**

When the Set Source Type page loads, Splunk Enterprise chooses a source type based on the data you specified. You can accept that recommendation or change it.

Here is an example of the Set Source Type page.
1. Look at the preview pane on the right side of the page to see how Splunk Enterprise will index the data. Review event breaks and timestamps.

2. (Optional) View the event summary by clicking on the View event summary link on the right. Splunk Web displays the event summary in a new window. See View event summary.

3. If the data appears the way that you want, then proceed to Step 5. Otherwise, choose from one of the following options:
   - **Choose an existing source type** to change the data formatting. See Choose an existing source type.
   - **Adjust time stamps, delimiters, and line breaking manually**, then save the changes as a new source type. See Adjust time stamps and event line breaks.

4. After making the changes, return to Step 1 to preview the data again.

5. After you are satisfied with the results, click "Next" to proceed to the Input Settings page.

**Choose an existing source type**

If the data does not appear the way that you want, see whether or not an existing source type fixes the problem.

**Note:** If Splunk Enterprise can detect a source type, the source type is displayed in the "Source type <sourcetype>" button. If a source type cannot be determined, "Sourcetype: System Defaults" is displayed. Input Settings
1. Click the **Source type: <sourcetype>** button to see a list of source type categories. Under each category is a list of source types within that category.
2. Hover over the category that best represents your data. As you do, the source types under that category appear in a pop-up menu to the right.
3. Select the source type that best represents your data. Splunk Web updates the data preview pane to show how the data looks under the new source type. You might need to scroll to see all source types in a category.
4. Review your data again.

**View event summary**
You can see a summary of the events within the data sample by clicking the "View Event Summary" link on the right side of the page. This summary shows the following information:

- The size of the sample data, in bytes.
- The number of events that were present in the sample.
- The chart that represents the distribution of the events over time. Splunk software uses date stamps within the file to determine how to display this chart.
- A breakdown of the number of lines each event in the sample took up.

**Adjust time stamps and event breaks**

If you choose an existing source type without success, then you can manually adjust how Splunk Enterprise processes timestamps and event line breaks for the incoming data.

To manually adjust time stamp and event line breaking parameters, use the [Event Breaks](#), [Timestamp](#), [Delimited Settings](#), and [Advanced](#) drop-down tabs on the left pane of the Set Sourcetypes page. The preview pane updates as you make changes to the settings.

**Note:** Some tabs appear only if Splunk Enterprise detects that a file is a certain type, or if you select a specific source type for a file.

- The Event breaks tab appears when Splunk Enterprise cannot determine how to line-break the file, or if you select a source type that does not have line breaking defined.
- The Delimited settings tab appears only when Splunk Enterprise detects that you want to import a structured data file, or you select a source type for structured data (such as `csv`).

For more information about how to adjust time stamps and event breaks, see Modify event processing.

1. Click the Event breaks tab. The tab displays the Break type buttons, which control how Splunk software line-breaks the file into events.
   - **Auto:** Detect event breaks based on the location of the time stamp.
   - **By Line:** Breaks every line into a single event.
   - **Regex?:** Uses the specified regular expression to determine line breaking.
2. Click the Timestamps tab. The tab expands to show options for extraction. Select from one of the following options.
• **Auto**: Extract timestamps automatically by looking in the file for timestamp events.
• **Current time**: Apply the current time to all events detected.
• **Advanced**: Specify the time zone, timestamp format (in a specific format known as strftime()), and any fields that comprise the timestamp.

3. Click the **Delimited settings** tab to display delimiting options.

   - **Field delimiter**: The delimiting character for structured data files, such as comma-separated value (CSV) files.
   - **Quote character**: The character that Splunk Enterprise uses to determine when something is in quotes.
   - **File preamble**: A regular expression that tells Splunk Enterprise to ignore one or more preamble lines (lines that don't contain any actual data) in the structured data file.
   - **Field Names**: How to determine field names: Automatically, based on line number, based on a comma-separated list, or through a regular expression.

After the results look the way you want, save your changes as a new **source type**, which you can then apply to the data when it is indexed.

4. Click the **Advanced** tab to display fields that let you enter attribute/value pairs that get committed directly to the **props.conf** configuration file.

   **Caution**: The "Advanced" tab requires advanced knowledge of Splunk features, and changes made here might negatively affect the indexing of your data. Consider consulting a member of Splunk Professional Services for help in configuring these options.

**Make configuration changes in the Advanced tab**

1. Click a field to edit props.conf entries that Splunk Enterprise generates based on your previous choices.
2. Click on the X to the right of an attribute/value field pair to delete that pair.
3. Click **New setting** to create a new attribute/value field pair and specify a valid attribute and value for **props.conf**.
4. Click **Apply settings** to commit the changes to the **props.conf** file.
**Next step**

Modify Input Settings

**Prepare your data for previewing**

The "Set Sourcetype" page works on single files only, and can only access files that reside on the Splunk deployment or have been uploaded there. Although it does not directly process network data or directories of files, you can work around those limitations.

**Preview network data**

You can direct some sample network data into a file, which you can then either upload or add as a file monitoring input. Several external tools can do this. On *nix, the most popular tool is `netcat`.

For example, if you listen for network traffic on UDP port 514, you can use `netcat` to direct some of that network data into a file.

```
nc -lu 514 > sample_network_data
```

For best results, run the command inside a shell script that has logic to kill `netcat` after the file reaches a size of 2MB. By default, Splunk software reads only the first 2MB of data from a file when you preview it.

After you have created the "sample_network_data" file, you can add it as an input, preview the data, and assign any new source types to the file.

**Preview directories of files**

If all the files in a directory are similar in content, then you can preview a single file and be confident that the results will be valid for all files in the directory. However, if you have directories with files of heterogeneous data, preview a set of files that represents the full range of data in the directory. Preview each type of file separately, because specifying any wildcard causes Splunk Web to disable the "Set Sourcetype" page.)
File size limit

Splunk Web displays the first 2MB of data from a file in the "Set Sourcetypes" page. In most cases, this amount provides a sufficient sampling of your data. If you have Splunk Enterprise, you can sample a larger quantity of data by changing the `max_preview_bytes` attribute in `limits.conf`. Alternatively, you can edit the file to reduce large amounts of similar data, so that the remaining 2MB of data contains a representation of all the types of data in the original file.

Modify event processing

You can change the event processing settings and save the improved settings as a new source type.

1. View the event data, as described in the "Set Sourcetype" page.
2. Modify the event processing settings.
3. Review the effect of your changes and iterate until you are satisfied.
4. Save the modified settings as a new source type.
5. Apply the new source type to any of your inputs.

Modify the event processing settings

To create the new source type, use the event-breaking and timestamp parameters, then save the source type.
On the left side of the "Set Sourcetypes" page, there are collapsible tabs and links for the three types of adjustments that you can perform:

- **Event Breaks.** Adjust the way that Splunk Enterprise breaks the data into events.
- **Timestamps.** Adjust the way Splunk Enterprise determines event timestamps.
- **Advanced mode.** If you have Splunk Enterprise, edit `props.conf`.

### Event breaks

To modify event break parameters, click **Event Breaks**. The bar opens to display the following buttons:

- **Auto.** Break events based on the location of timestamps in the data.
- **Every line.** Consider every line a single event.
- **Regex...** Use specified regular expression to break data into events.

For information on line breaking, See [Configure event linebreaking](#). For a primer on regular expression syntax and usage, see Regular-Expressions.info. You can test your regular expression by using it in a search with the rex search command. The Splunk software also maintains a list of useful third-party tools for writing and testing regular expressions.

### Timestamps
To modify timestamp recognition parameters click the Timestamps tab to expand it. The tab opens to reveal these options:

For Extraction, you can choose one of these options:

- **Auto.** Locate the timestamp automatically.
- **Current Time.** Uses the current system time.
- **Advanced.** Specify additional advanced parameters to adjust the timestamp.

The "Advanced" parameters are:

- **Timezone.** The time zone that you want to use for the events.
- **Timestamp format.** A string that represents the timestamp format for Splunk Enterprise to use when searching for timestamps in the data.
- **Timestamp prefix.** A regular expression that represents the characters that appear before a timestamp.
- **Lookahead.** The number of characters that Splunk Enterprise should look into the event (or, for the regular expression that you specified in "Timestamp prefix") for the timestamp.

**Note:** If you specify a timestamp format in the "Timestamp format" field and the timestamp is not located at the very start of each event, you must also specify a prefix in the Timestamp prefix field. Otherwise, Splunk Enterprise cannot process the formatting instructions, and every event will contain a warning about the inability to use strftime. (It's possible that you still end up with a valid timestamp, based on how Splunk Enterprise attempts to recover from the problem.)

For information on configuring timestamps, see "Configure timestamps".
**Advanced**

To modify advanced parameters, click the **Advanced** tab. The tab shows options that let you specify source type properties by editing the underlying `props.conf` file.

You can add or change source type properties by specifying attribute/value pairs. See `props.conf` for details on how to set these properties.

The "Advanced" box shows the current, complete set of properties for the selected source type:

- Settings generated by changes made in the **Event Breaks** or **Timestamps** tabs (after you click **Apply**).
- Pre-existing settings for a source type that was either auto-detected or manually selected when you first previewed the file.
- Settings you apply from the **Additional settings** text box (after you click **Apply settings**).

For information on how to set source type properties, see "props.conf" in the Configuration file reference. See also "How timestamp assignment works" and "event linebreaking."

**How Splunk Enterprise combines settings**

The settings changes you make in **Advanced mode** take precedence. For example, if you alter a timestamp setting using the **Timestamps** tab and also make a conflicting timestamp change in **Advanced mode**, the **Advanced mode** change takes precedence over the modification that you made in the "Timestamps" tab.

Starting with highest precedence, here is how Splunk Enterprise combines any adjustments with the underlying default settings:
- Advanced mode changes
- Event Breaks/Timestamps changes
- Settings for the underlying source type, if any
- Default system settings for all source types

Also, if you return to the Event Breaks or Timestamps tabs after making changes in Advanced mode, the changes will not be visible from those tabs.

**Review your changes**

When you are ready to view the effect of your changes, select **Apply settings**. Splunk Web refreshes the screen, so you can review the effect of your changes on the data.

To make further changes using any of the three adjustment methods available again, select **Apply changes** to view the effect of the changes on your data.

**Save modifications as a new source type**

1. Click "Save As" next to the "Sourcetype" button. Splunk Web displays a dialog box where you can name your new source type, choose the category in which it should be shown in the "Sourcetype" button dialog, and the application context it should use.
2. Enter the **Name** of the new source type.
3. Enter the **Description** of the source type.
4. Select the **Category** in which the source type should appear when you click "Sourcetype".
5. Select the **App** for which the new source type should be used.
6. Click **Save** to save the source type and return to the Set Sourcetypes page.

**Next steps**

You have several options after you save the source type:

1. (Optional) Click **Next** to apply the source type to your data and proceed to the **Input settings** page.
2. (Optional) Click "<" to go back and choose a new file to upload or monitor.
3. (Optional) Click **Add data** to return to the beginning of the Add Data wizard.

**Modify input settings**

After you select the source (or set your source type when uploading or monitoring a single file), the **Modify input settings** page appears.

You can specify additional parameters for your data input, such as its source type, its application context, its host value, and the index where data from the input should be stored.

The following input settings are available:

**Configure source type**

You can specify the source type to be applied to your data with the "Source type" setting. This setting appears when:

- You specify a directory as a data source.
- You specify a network input as a data source.
- You specify a data source that has been forwarded from another Splunk instance.

If your data source does not meet these criteria, then the "Source type" setting does not appear.
Specify a source type

1. Click one of the buttons.
   - **Select**: Applies the source type you specify to the data. When you click "Select", a drop-down appears.
   - **New**: Adds a new source type. When you click "New", two text fields and a drop-down appear.

Choose an existing source type

1. From the "Select Source Type" drop-down, choose the category that best represents the source type you want.
2. Choose the source type from the pop-up list that appears.

Add a new source type

1. In the "Source Type" text field, enter the name of the new source type.
2. Choose a category for the source type in the "Source Type Category" drop-down.
3. In the "Source Type Description" field, enter the description for the source type.

Configure app context

The **Application Context** setting determines the context in which the input should collect data. When you set the application context, you determine which Splunk app that the input configuration gets stored into. Splunk apps run on the Splunk platform and typically address use cases. Application contexts improve manageability of input and source type definitions. Application contexts are loaded based on precedence rules. See Configuration file precedence in the *Admin* manual.

- Select the application context you want this input to operate within by clicking the drop-down list and selecting the application context you want.

Configure host value

Splunk Enterprise tags events with a host. You can configure how the software determines the host value.

- **IP**: Uses the IP address of the host from which the event originates.
- **DNS**: Use Domain Name Services (DNS). Events are tagged with the host name that Splunk software determines using DNS name resolution.
• Custom: Uses the host value you assign in the "Host field value" text box that appears when you select this option.

Index

The "Index" setting determines the index where the events for this input should be stored.

1. To use the default index, leave the drop-down list set to "Default". Otherwise, click the drop-down list and select the index you want the data to go to by clicking the selection in the list.
2. (Optional) If the index you want to send the data to is not in the list, and you have permissions to create indexes, you can create a new index by clicking the Create a new index button.

After you make your selections, click Next to proceed to the final step of the "Add Data" process.

Distribute source type configurations in Splunk Enterprise

If you create source types in Splunk Cloud using Splunk Web, Splunk Cloud manages the source type configurations automatically. However, if you have Splunk Enterprise and manage a distributed configuration, you must distribute new source type as described in this topic.

You can use either the "Set source type" or source type management pages in Splunk Web to create new source types, which you can then assign to inputs from specific files or directories, or for network inputs. Either of these pages saves a new source type to a props.conf configuration file on the local Splunk Enterprise instance. You can then distribute this file to other Splunk Enterprise instances so that they recognize the new source type.

You can use a new source type in a distributed environment where you have forwarders consuming data and then sending the data to indexers.

To install this new source type, follow these high-level steps:

1. Distribute the props.conf file that contains the source type definition to the $SPLUNK_HOME/etc/system/local directory on indexers that you want to index data with the source type you created.
2. Use the new source type when you define an input on forwarders that send data to those indexers.

When a forwarder sends data that has been tagged with the new source type to an indexer, the indexer can correctly process it into events.

Data preview props.conf file

When you create a source type in the "Set Sourcetype" page, the software saves the source type definition as a stanza in a props.conf file in the app that you selected when you saved the source type. If you later create additional source types, they are saved to the same props.conf file.

For example, if you selected the "Search and Reporting" app, the file resides in $SPLUNK_HOME/etc/apps/search/local/props.conf. The only exception is the "System" app: If you choose that app when saving the source type, the file resides in $SPLUNK_HOME/etc/system/local..

Note: A Splunk Enterprise instance might have multiple versions of some configuration files, in several directories. At run-time, Splunk Enterprise combines the contents of configuration files according to a set of precedence rules. For background on how configuration files work, see About configuration files and Configuration file precedence.

Distribute props.conf to other indexers

After you create source types, you can distribute props.conf to another Splunk Enterprise instance. That instance can then index any incoming data that you tag with the new source type.

A Splunk best practice is to place the configuration file in its own app directory on the target Splunk Enterprise instance; for example, $SPLUNK_HOME/etc/apps/custom_sourcetype/local/.

To distribute configuration files to other Splunk instances, you can use a deployment server or another distribution tool. See the Updating Splunk Instances manual.

Note: Splunk software uses the source type definitions in props.conf to parse incoming data into events. For this reason, you can only distribute the file to a Splunk Enterprise instance that performs parsing (either an indexer or a heavy forwarder.)
Specify the new source type in forwarder inputs

Forwarders (with the exception of the heavy forwarder) do not have Splunk Web. This means that you must configure their inputs through the CLI or the `inputs.conf` configuration file. When you specify an input in that file, you can also specify its source type. For information on `inputs.conf`, read the section on `inputs.conf` in the Configuration file reference.

1. To tag a forwarder input with a new source type, add the source type to the input stanza in `inputs.conf`. For example:

   ```
   [tcp://:9995]
   sourcetype = new_network_type
   ```

2. Confirm that all of the indexers that the forwarder sends data to have copies of the `props.conf` file that contains the source type definition for `new_network_type`. When the forwarder sends data to the indexers, they can identify the new source type and correctly format the data.
Get data from files and directories

Monitor files and directories

Splunk Enterprise has three file input processors: monitor, MonitorNoHandle, and upload.

You can use monitor to add nearly all your data sources from files and directories. However, you might want to use upload to add one-time inputs, such as an archive of historical data.

On hosts that run Windows Vista or Windows Server 2008 and later, you can use MonitorNoHandle to monitor files which the system rotates automatically. The MonitorNoHandle input works only on Windows hosts.

Add inputs to monitor or upload using any of these methods:

- Splunk Web
- The CLI
- inputs.conf

You can add inputs to MonitorNoHandle using either the CLI or inputs.conf.

Use the "Set Sourcetype" page to see how the data from a file will be indexed. See The "Set Sourcetype" page for details.

How the monitor processor works

Specify a path to a file or directory and the monitor processor consumes any new data written to that file or directory. This is how you can monitor live application logs such as those coming from Web access logs, Java 2 Platform Enterprise Edition (J2EE) or .NET applications, and so on.

Splunk Enterprise monitors and indexes the file or directory as new data appears. You can also specify a mounted or shared directory, including network file systems, as long as Splunk Enterprise can read from the directory. If the specified directory contains subdirectories, the monitor process recursively examines them for new files, as long as the directories can be read.

You can include or exclude files or directories from being read by using whitelists and blacklists.
If you disable or delete a monitor input, Splunk Enterprise does not stop indexing the files that the input references. It only stops checking those files again. To stop all in-process data indexing, the Splunk server must be stopped and restarted.

**How Splunk Enterprise handles monitoring of files during restarts**

When the Splunk server is restarted, it continues processing files where it left off. It first checks for the file or directory specified in a monitor configuration. If the file or directory is not present on start, Splunk Enterprise checks for it every 24 hours from the time of the last restart. The monitor process scans subdirectories of monitored directories continuously.

Monitor inputs may overlap. So long as the stanza names are different, Splunk Enterprise treats them as independent stanzas and files matching the most specific stanza will be treated in accordance with its settings.

**How Splunk Enterprise monitors archive files**

Archive files (such as a `.tar` or `.zip` file, are decompressed before being indexed. The following types of archive files are supported:

- `.tar`
- `.gz`
- `.bz2`
- `.tar.gz` and `.tgz`
- `.tbz` and `.tbz2`
- `.zip`
- `.z`

If you add new data to an existing archive file, the entire file is reindexed, not just the new data. This can result in event duplication.

**How Splunk Enterprise monitors files that the operating system rotates on a schedule**

The monitoring process detects log file rotation and does not process renamed files that it has already indexed (with the exception of `.tar` and `.gz` archives). See How Splunk Enterprise handles log file rotation.
How Splunk Enterprise monitors nonwritable Windows files

Windows can prevent Splunk Enterprise from reading open files. If you need to read files while they are being written to, you can use the `monitorNoHandle` input.

Restrictions on file monitoring

Splunk Enterprise cannot monitor a file whose path exceeds 1024 characters.

Files with a `.splunk` filename extension are also not monitored, because files with that extension contain Splunk metadata. If you need to index files with a `.splunk` extension, use the `add oneshot` CLI command.

Why use upload or batch?

To index a static file once, select **Upload** in Splunk Web.

You can also use the CLI `add oneshot` or `spool` commands for the same purpose. See Use the CLI for details.

If you have Splunk Enterprise, you can use the `batch` input type in `inputs.conf` to load files once and destructively. By default, the Splunk batch processor is located in `$SPLUNK_HOME/var/spool/splunk`. If you move a file into this directory, the file is indexed and then deleted.

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

Why use MonitorNoHandle?

This Windows-only input lets you read files on Windows systems as Windows writes to them. It does this by using a kernel-mode filter driver to capture raw data as it gets written to the file. Use this input stanza on files which get locked open for writing. You can use this input stanza on a file which the system locks open for writing, such as the Windows DNS server log file.

Caveats for using MonitorNoHandle

The `MonitorNoHandle` input has the following caveats:

- `MonitorNoHandle` only works on Windows Vista or Windows Server 2008 and later operating systems. It does not work with earlier version of Windows, nor does it work on operating systems that are not Windows.
• You can only monitor single files with MonitorNoHandle. To monitor more than one file, you must create a MonitorNoHandle input stanza for each file.
• You cannot monitor directories with MonitorNoHandle.
• If a file you choose to monitor with MonitorNoHandle already exists, Splunk Enterprise does not index its current contents, only new information that comes into the file as processes write to it.
• When you monitor a file with MonitorNoHandle, the source field for the file is MonitorNoHandle, not the name of the file. If you want to have the source field be the name of the file, you must set it explicitly in inputs.conf. See Monitor files and directories with inputs.conf.

Monitor files and directories with Splunk Web

If you have Splunk Enterprise, you can use Splunk Web to add inputs from files and directories.

Go to the Add New page

You add an input from the Add Data page in Splunk Web.

You can get there by two routes:

• Splunk Home
• Splunk Settings

Splunk Settings:

1. Click Settings in the upper right corner of Splunk Web.
2. In the Data section of the Settings pop-up, click Data Inputs.
3. Click Files & Directories.
4. Click New to add an input.

Splunk Home:

1. Click Add Data in Splunk Home.
2. Click Upload to upload a file, Monitor to monitor a file, or Forward to forward a file.

Note: Forwarding a file requires additional setup. See the following topics:
• Configure the universal forwarder if you work with universal forwarders.
• Enable forwarding on a Splunk Enterprise instance if you work with heavy and light forwarders.

Select the input source

1. To add a file or directory input, click Files & Directories.
2. In the File or Directory field, 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). Confirm that Splunk Enterprise
   has read access to the mounted drive, as well as to the files you want to
   monitor.
3. Choose how you want Splunk Enterprise to monitor the file.
   - Continuously Monitor. Sets up an ongoing input. Splunk
     Enterprise monitors the file continuously for new data.
   - Index Once. Copies a file on the server into Splunk Enterprise.
4. Click Next. If you specified a directory in the "File or Directory" field,
   Splunk Enterprise refreshes the screen to show fields for "whitelist" and
   "blacklist". These fields let you specify regular expressions that Splunk
   Enterprise then uses to match files for inclusion or exclusion. Otherwise,
   Splunk Enterprise proceeds to the "Set Sourcetype" page where you can
   preview how Splunk Enterprise proposes to index the events.

   For more information on how to whitelist and blacklist data, see Whitelist or
   blacklist specific incoming data.

Preview your data and set its source type

When you add a new file input, Splunk Enterprise lets you set the source type of
your data and preview how it will look once it has been indexed. This lets you
ensure that the data has been formatted properly and make any necessary
adjustments.

   For information about this page, see The Set Sourcetype page.

   If you skip previewing the data, the Input Settings page appears.

   Note: You cannot preview directories or archived files. You also cannot preview
   inputs with the Log to Metrics source type.
Specify input settings

You can specify application context, default host value, and index in the Input Settings page. All parameters are optional.

1. Select the appropriate Application context for this input.
2. Set the Host name value.
   Note: Host only sets the host field in the resulting events. It does not direct Splunk Enterprise to look on a specific host on your network.
3. Set the Index that Splunk Enterprise should send data to for this input.
   Leave the value as "default", unless you have defined multiple indexes and want to use one of those instead.
4. Click Review to review all of the choices you have made.

Review your choices

After you specifying all input settings, review your selections. Splunk Web lists the options you selected, including but not limited to the type of monitor, the source, the source type, the application context, and the index.

1. Review the settings.
2. If they do not match what you want, click < to go back to the previous step in the wizard. Otherwise, click Submit. The "Success" page appears and Splunk Enterprise begins indexing the specified file or directory.

Monitor files and directories with the CLI

Monitor files and directories via the Splunk Enterprise Command Line Interface (CLI). To use the CLI, navigate to the $SPLUNK_HOME/bin/ directory from a command prompt or shell, and use the splunk command in that directory.

The CLI has built-in help. Access the main CLI help by typing splunk help. Individual commands have their own help pages as well. Access that help by typing splunk help <command>.

CLI commands for input configuration

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

<table>
<thead>
<tr>
<th>Command</th>
<th>Command syntax</th>
<th>Action</th>
</tr>
</thead>
</table>

44
<table>
<thead>
<tr>
<th>Command</th>
<th>Command syntax</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>add monitor</td>
<td><code>add monitor [-source] &lt;source&gt; [-parameter value] ...</code></td>
<td>Monitor inputs from <code>&lt;source&gt;</code>.</td>
</tr>
<tr>
<td>edit monitor</td>
<td><code>edit monitor [-source] &lt;source&gt; [-parameter value] ...</code></td>
<td>Edit a previously added monitor input for <code>&lt;source&gt;</code>.</td>
</tr>
<tr>
<td>remove monitor</td>
<td><code>remove monitor [-source] &lt;source&gt; ...</code></td>
<td>Remove a previously added monitor input for <code>&lt;source&gt;</code>.</td>
</tr>
<tr>
<td>list monitor</td>
<td><code>list monitor</code></td>
<td>List the currently configured monitor inputs.</td>
</tr>
<tr>
<td>add oneshot</td>
<td><code>add oneshot &lt;source&gt; [-parameter value] ...</code></td>
<td>Copy the file <code>&lt;source&gt;</code> directly into Splunk. This uploads the file once, but Splunk Enterprise does not continue to monitor it. You cannot use the <code>oneshot</code> command against a remote Splunk Enterprise instance. You also cannot use the command with either recursive folders or wildcards as a source. Specify the exact source path of the file you want to monitor.</td>
</tr>
<tr>
<td>spool</td>
<td><code>spool &lt;source&gt;</code></td>
<td>Copy the file <code>&lt;source&gt;</code> into Splunk Enterprise using the sinkhole directory. Similar to <code>add oneshot</code>, except that the file spools from the sinkhole directory, rather than being added immediately. You cannot use the <code>spool</code> command against a remote Splunk Enterprise instance. You also cannot use the command with either recursive folders or wildcards as a source. Specify the exact source path of the file you want to monitor.</td>
</tr>
</tbody>
</table>

**CLI parameters for input configuration**

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

45
Note: You can only set one `-hostname`, `-hostregex` or `-hostsegmentnum` per command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;source&gt;</code></td>
<td>Yes</td>
<td>Path to the file or directory to monitor/upload for new input. Unlike the other parameters, the syntax for this parameter can be the value itself. It does not have to follow a parameter flag. You can use either of &quot;./splunk monitor &lt;source&gt;&quot; or &quot;./splunk monitor -source &lt;source&gt;&quot;.</td>
</tr>
<tr>
<td>sourcetype</td>
<td>No</td>
<td>Specify a sourcetype field value for events from the input source.</td>
</tr>
<tr>
<td>index</td>
<td>No</td>
<td>Specify the destination index for events from the input source.</td>
</tr>
<tr>
<td>hostname or host</td>
<td>No</td>
<td>Specify a host name to set as the host field value for events from the input source. These parameters are functionally equivalent.</td>
</tr>
<tr>
<td>hostregex or host_regex</td>
<td>No</td>
<td>Specify a regular expression to use to extract the host field value from the source key. These parameters are functionally equivalent.</td>
</tr>
<tr>
<td>hostsegmentnum or host_segment</td>
<td>No</td>
<td>An integer, which determines what &quot;/&quot; separated segment of the path to set as the host field value. If set to 3, for example, the third segment of the path is used. These parameters are functionally equivalent.</td>
</tr>
<tr>
<td>rename-source</td>
<td>No</td>
<td>Specify a value for the &quot;source&quot; field to be applied to data from this file.</td>
</tr>
<tr>
<td>follow-only</td>
<td>No</td>
<td>Set to true or false. Default is false.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When set to true, Splunk Enterprise reads from the end of the source (like the &quot;tail -f&quot; Unix command).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This parameter is not available for <code>add oneshot</code>.</td>
</tr>
</tbody>
</table>
## Parameter Required? Description

<table>
<thead>
<tr>
<th>Example 1: Monitor files in a directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>The following example shows how to monitor files in <code>/var/log/</code>.</td>
</tr>
<tr>
<td>Add <code>/var/log/</code> as a data input:</td>
</tr>
<tr>
<td><code>.splunk add monitor /var/log/</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example 2: Monitor windowsupdate.log</th>
</tr>
</thead>
<tbody>
<tr>
<td>The following example shows how to monitor the Windows Update log file where Windows logs automatic updates, sending the data to an index called &quot;newindex&quot;.</td>
</tr>
<tr>
<td>Add <code>C:\Windows\windowsupdate.log</code> as a data input:</td>
</tr>
<tr>
<td><code>splunk add monitor c:\Windows\windowsupdate.log -index newindex</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example 3: Monitor Internet Information Server (IIS) logging</th>
</tr>
</thead>
<tbody>
<tr>
<td>This example shows how to monitor the default location for Windows IIS logging.</td>
</tr>
<tr>
<td>Add <code>C:\windows\system32\LogFiles\W3SVC</code> as a data input:</td>
</tr>
<tr>
<td><code>.splunk add monitor c:\windows\system32\LogFiles\W3SVC</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example 4: Upload a file</th>
</tr>
</thead>
<tbody>
<tr>
<td>This example shows how to upload a file into Splunk. Splunk Enterprise consumes the file only once. It does not monitor it continuously.</td>
</tr>
<tr>
<td>Upload <code>/var/log/applog</code> (C:\Program Files\AppLog\log.txt on Windows) directly into Splunk Enterprise with the <code>add oneshot</code> command:</td>
</tr>
<tr>
<td>Unix</td>
</tr>
<tr>
<td>-------------------------------------------</td>
</tr>
<tr>
<td><code>.splunk add oneshot /var/log/applog</code></td>
</tr>
</tbody>
</table>

You can also upload a file through the sinkhole directory with the `spool` command:
Monitor files and directories with inputs.conf

You can use inputs.conf to monitor files and directories with Splunk Enterprise. Inputs.conf provides the most configuration options for setting up a file monitor input. For Splunk Cloud, use Splunk Web to configure file monitoring inputs instead.

To configure an input to Splunk Enterprise, add a stanza to inputs.conf in $SPLUNK_HOME/etc/system/local/ or your own custom application directory in $SPLUNK_HOME/etc/apps/. These locations are on the machine that runs Splunk Enterprise.

You can configure multiple settings in an input stanza. If you do not specify a value for a setting, Splunk Enterprise uses the default for that setting. You can find the defaults for settings in $SPLUNK_HOME/etc/system/default/inputs.conf.

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

Configure monitoring of files with inputs.conf

1. On the machine that runs Splunk Enterprise, open a shell or command prompt.
2. Change to the $SPLUNK_HOME/etc/system/local directory.
3. (Optional) If inputs.conf does not exist, create the file.
4. Open inputs.conf for editing with a text editor.
5. Add a stanza that references the files or directories that you want to monitor. For example, to monitor the /var/log/messages file on a *nix system, specify:

```
[monitor:///var/log/messages]
disabled = 0
```

To monitor C:\Windows\System32\WindowsUpdate.log on a Windows system, specify:

```
[monitor://C:\Windows\System32\WindowsUpdate.log]
disabled = 0
```
6. (Optional) Add settings that further configure the input, depending on what you want the input to do. See Configuration settings later in this topic, or review the inputs.conf configuration specification file in the Admin Manual for additional settings.

[monitor://path/to/file]
disabled = 0
setting1 = value
setting2 = value
...

7. Save the inputs.conf file and close it.
8. Either restart Splunk Enterprise or reload the configuration by running the following command. Splunk Enterprise prompts you for credentials if you reload the configuration.

./splunk _internal call /services/data/inputs/monitor/_reload -auth

Configuration settings

You can use the following settings in both monitor and batch input stanzas.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>host = &lt;string&gt;</td>
<td>Sets the host key to a static initial value for this stanza. The input processor uses the key during parsing and indexing to set the host field and uses the field during searching. Splunk Enterprise prepends the &lt;string&gt; with host:.</td>
<td>the IP address or fully-qualified domain name of the host where the data originated.</td>
</tr>
<tr>
<td>index = &lt;string&gt;</td>
<td>Sets the index where events from this input will be stored. Splunk Enterprise prepends the &lt;string&gt; with index:. For more information about the index field, see How indexing works in the Managing Indexers and Clusters manual.</td>
<td>main or whatever you set the default index to</td>
</tr>
<tr>
<td>sourcetype = &lt;string&gt;</td>
<td>Sets the sourcetype key/field for events from this input. Explicitly declares the source Splunk Enterprise picks a source</td>
<td></td>
</tr>
<tr>
<td>Setting</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>queue = parsingQueue</td>
<td>Specifies where the input processor should deposit the events that it reads. Set to &quot;parsingQueue&quot; to apply props.conf and other parsing rules to your data. Set to &quot;indexQueue&quot; to send your data directly into the index.</td>
<td>parsingQueue</td>
</tr>
<tr>
<td>_TCP_ROUTING = &lt;tcpout_group_name&gt;,&lt;tcpout_group_name&gt;,...</td>
<td>Specifies a comma-separated list of tcpout group names. Use this setting to selectively forward your data to specific indexers by specifying the tcpout groups that the forwarder should use when forwarding the data.</td>
<td>the groups present in 'defaultGroup' in [tcpout] stanza in outputs.conf</td>
</tr>
</tbody>
</table>

Sets the sourcetype key initial value. Splunk Enterprise uses the key during parsing and indexing to set the source type field and uses the source type field during searching. Splunk Enterprise prepends the `<string>` with sourcetype::.

For more information about source types, see Why source types matter.

Sets the sourcetype key initial value. Splunk Enterprise uses the key during parsing and indexing to set the source type field and uses the source type field during searching. Splunk Enterprise prepends the `<string>` with sourcetype::.

For more information about source types, see Why source types matter.
Monitor syntax

Monitor input stanzas configure Splunk Enterprise to watch all files in the `<path>` or the `<path>` itself if it represents a single file. You must specify the input type before the path, so add three slashes in the path if the path includes the root directory on *nix machines.

You can use wildcards for the path. See Specify input paths with wildcards.

[monitor://<path>]
<setting1> = <val1>
<setting2> = <val2>
...

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

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>source = &lt;string&gt;</td>
<td>Sets the source field for events from this input. You can use this setting when using the MonitorNoHandle input and want to set the source to the name of the file you are monitoring. Otherwise, do not use it.</td>
<td>the input file path (except in the case of MonitorNoHandle, where it is MonitorNoHandle)</td>
</tr>
</tbody>
</table>
override unless absolutely necessary. Consider use of source types, tagging, and search wildcards instead. The input layer usually provides a more accurate string to aid in problem analysis and investigation by accurately recording the file from which the data was retrieved.

Splunk Enterprise prepends the `<string>` with `source::`.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>crcSalt = <code>&lt;string&gt;</code></td>
<td>Forces Splunk Enterprise to index files that have matching cyclic redundancy checks (CRCs). By default, the software only performs CRCs against the first few lines of a file. This behavior prevents indexing of the same file twice, even though you might have renamed it, such as with rolling log files. However, because the CRC counts only the first few lines of the file, it is possible for legitimately different files to have matching CRCs.) If set, Splunk Enterprise adds <code>&lt;string&gt;</code> to the CRC. If set to <code>&lt;SOURCE&gt;</code>, Splunk Enterprise adds the full source path to the CRC. This ensures that each file being monitored has a unique CRC. Use caution with this setting for rolling log files. It can lead to the log file being re-indexed after it has rolled. This setting is case sensitive.</td>
<td>N/A</td>
</tr>
<tr>
<td>ignoreOlderThan = <code>&lt;time_window&gt;</code></td>
<td></td>
<td>0 (disabled)</td>
</tr>
</tbody>
</table>
Causes the input to stop checking files for updates if the file modification time (modtime) has passed the `<time_window>` threshold. This improves the speed of file tracking operations when monitoring directory hierarchies with large numbers of historical files (for example, when active log files share a directory with old files that no longer get writes).

Splunk Enterprise does not index files whose modification time falls outside `<time_window>` when it first attempts to monitor the file.

You must specify `<number><unit>`. For example, "7d" indicates one week. Valid units are "d" (days), "h" (hours), "m" (minutes), and "s" (seconds).

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>followTail = 0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>If set to 1, monitoring begins at the end of the file (like *nix tail -f). This only applies to files the first time Splunk Enterprise attempts to monitor them. After that, Splunk Enterprise keeps track of the file using its internal file position records.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>whitelist = &lt;regular expression&gt;</td>
<td>If set, Splunk Enterprise only monitors files whose names match the specified regular expression.</td>
<td>N/A</td>
</tr>
<tr>
<td>blacklist = &lt;regular expression&gt;</td>
<td>If set, Splunk Enterprise does NOT monitor files whose names match the specified regular expression.</td>
<td>N/A</td>
</tr>
<tr>
<td>alwaysOpenFile = 0</td>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td>If set to 1, Splunk Enterprise opens a file to check if it has already been indexed. This is only useful for files that don't update their modification</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Setting</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>recursive = true</td>
<td>false</td>
<td>If set to false, Splunk Enterprise does not look into subdirectories that it finds within a monitored directory.</td>
</tr>
<tr>
<td>time_before_close = &lt;integer&gt;</td>
<td>The modification time delta required before Splunk Enterprise can close a file on End-of-file (EOF). Tells the system not to close files that have been updated in the past &lt;integer&gt; seconds.</td>
<td>3</td>
</tr>
<tr>
<td>followSymlink = true</td>
<td>false</td>
<td>If false, Splunk Enterprise ignores symbolic links that it finds within a monitored directory.</td>
</tr>
</tbody>
</table>

**MonitorNoHandle syntax**

On Windows systems only, the `MonitorNoHandle` input monitors files without using Windows file handles. This lets you read special log files like the Windows DNS server log files.

You must specify a valid path to a file when you use the `MonitorNoHandle` input stanza. You cannot specify a directory. If you specify a file that already exists, Splunk Enterprise does not index the existing data in the file. It only indexes new data that the system writes to the file.

`MonitorNoHandle` sets the source for files you monitor to `MonitorNoHandle`. To specify the file name as the source, you must use the `source` setting in the stanza for the `MonitorNoHandle` input for the file.

You can only configure `MonitorNoHandle` on a Windows machine with inputs.conf or the Command Line Interface (CLI). You cannot configure it in Splunk Web or on a *nix machine.
**Batch syntax**

Use batch to set up a one-time, destructive input of data from a source.

For continuous, non-destructive inputs, use `monitor`. Splunk enterprise deletes data that it has indexed with the `batch` monitor.

```plaintext
[batch://<path>]
move_policy = sinkhole
<setting1> = <val1>
<setting2> = <val2>
...
```

When you define batch inputs, you must include the setting `move_policy = sinkhole`. This loads the file destructively. Do not use the batch input type for files that you do not want to delete after indexing.

To ensure that Splunk Enterprise indexes new events when you copy over an existing file with new contents, set the `CHECK_METHOD = modtime` setting in `props.conf` for the input source. This checks the modification time of the file and re-indexes it when the time changes. Splunk Enterprise indexes the entire file, which can result in duplicate events.

**Examples of monitor input stanzas**

**Single *nix file**

This example stanza configures Splunk Enterprise to index the single file, `/var/log/messages`.

```plaintext
[monitor://var/log/messages]
disabled = 0
sourcetype = unixlog
```
Single Windows directory

This Windows example configures Splunk Enterprise to monitor the directory, C:\Windows\Logs, and all the files in it.

[monitor://C:\Windows\Logs]
disabled = 0

Multiple Windows directories

This Windows example tells Splunk Enterprise to monitor all of the directories in C:\Windows\Debug.

[monitor://C:\Windows\Debug\*]
disabled = 0

Multiple *nix directories with a wildcard

This example configures Splunk Enterprise to monitor directories like /apache/foo/log, /apache/bar/log, etc.

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

Multiple *nix files in one directory with a wildcard

This *nix example configures Splunk Enterprise to monitor multiple files in one directory, such as /apache/*.log.

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

MonitorNoHandle, single Windows file

This Windows example uses the MonitorNoHandle input to monitor a file that Windows has open for writing, such as C:\Windows\System32\DNS\DNS.log.

[MonitorNoHandle://<path>]
disabled = 0
source = <path>
<setting1> = <val1>
<setting2> = <val2>
...
**Batch**

This batch example loads and deletes all files from the directory `system/flight815/`.

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

### Specify input paths with wildcards

You can configure inputs manually by editing the `inputs.conf` file. Input path specifications in `inputs.conf` do not use regular expressions (regexes) but rather Splunk-defined wildcards. This topic discusses how to specify these wildcards in a path in `inputs.conf`. To specify wildcards, you must use `inputs.conf` to specify file and directory monitor inputs.

### Wildcard overview

A wildcard is a character that you can substitute for one or more unspecified characters when searching text or selecting multiple files or directories. You can use wildcards to specify the input path for a file or directory monitor input.

<table>
<thead>
<tr>
<th>Wildcard</th>
<th>Description</th>
<th>Reg. Exp. equivalent</th>
<th>Example(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>...</code></td>
<td>The ellipsis wildcard recurses through directories and any number of levels of subdirectories to find matches. If you specify a folder separator (for example, <code>//var/log/.../file</code>), it does not match the first folder level, only subfolders.</td>
<td>.*</td>
<td>/foo/.../bar.log matches the files /foo/1/bar.log, /foo/2/bar.log, /foo/1/2/bar.log, etc., but does not match /foo/bar.log, or /foo/3/notbar.log Because a single ellipse recurses through all folders and subfolders, /foo/.../bar.log matches the same as /foo/.../.../bar.log.</td>
</tr>
<tr>
<td><code>*</code></td>
<td>The asterisk wildcard matches anything in that specific folder path segment.</td>
<td>[^/]*</td>
<td>/foo/*/bar matches the files /foo/1/bar, /foo/2/bar, etc., but does not match /foo/bar or /foo/1/2/bar.</td>
</tr>
</tbody>
</table>
Unlike "...", "*" does not recurse through subfolders.

<table>
<thead>
<tr>
<th>Wildcard</th>
<th>Description</th>
<th>Reg. Exp. equivalent</th>
<th>Example(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>/foo/m*r/bar matches /foo/mr/bar, /foo/mir/bar, /foo/moor/bar, etc.</td>
<td>/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.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A single period (.) is not a wildcard, and is the regular expression equivalent of .</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**Wildcards and regular expression metacharacters**

When determining the set of files or directories to monitor, Splunk Enterprise splits elements of a monitoring stanza into segments. Segments are blocks of text between directory separator characters ("/" or \\"\") in the stanza definition. If you specify a monitor stanza that contains segments with both wildcards and regular expression metacharacters (such as (, ), [, ], and |), those characters behave differently depending on where the wildcard is in the stanza.

If a monitoring stanza contains a segment with regular expression metacharacters before a segment with wildcards, the metacharacters are treated literally, as if you wanted to monitor files or directories with those characters in the file or directory names. For example:

```
[monitor://var/log/log(a|b).log]
```

monitors the /var/log/log(a|b).log file. The (a|b) is not treated as a regular expression because no wildcards are present.

```
[monitor://var/log()/log*.log]
```

monitors all files in the /var/log()/ directory that begin with log and have the extension .log. The () is not treated as a regular expression because it is in the
segment before the wildcard.

If the regular expression metacharacters occur within or after a segment that contains a wildcard, Splunk Enterprise treats the metacharacters as a regular expression and matches files to monitor accordingly. For example:

```
[monitor:///var/log()/log(a|b)*.log]
```

monitors all files in the `/var/log()` directory that begin with either `loga` or `logb` and have the extension `.log`. The first set of `()` is not treated as a regular expression because the wildcard is in the following segment. The second set of `()` does get treated as a regular expression because it is in the same segment as the wildcard `*`.

```
[monitor:///var/.../log(a|b).log]
```

monitors all files in any subdirectory of the `/var/` directory named `loga.log` and `logb.log`. Splunk Enterprise treats `(a|b)` as a regular expression because of the wildcard `...` in the previous stanza segment.

```
[monitor:///var/.../log[A-Z0-9]*.log]
```

monitors all files in any subdirectory of the `/var/` directory that:

- begin with `log`, then
- contain a single capital letter (from A-Z) or number (from 0-9), then
- contain any other characters, then
- end in `.log`.

The expression `[A-Z0-9]*` is treated as a regex because of the wildcard `...` in the previous stanza segment.

**Input examples**

To monitor `/apache/foo/logs`, `/apache/bar/logs`, `/apache/bar/1/logs`:

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

To monitor `/apache/foo/logs`, `/apache/bar/logs`, but not `/apache/bar/1/logs` or `/apache/bar/2/logs`:

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

To monitor any file directly under `/apache/` that ends in `.log`:

```
[monitor:///apache/*/logs]
```
[monitor:///apache/*\.log]
To monitor all log files recursively in D:\Program Files\Splunk\etc\apps:

[monitor://D:\Program Files\Splunk\etc\apps\*\...\*\.log]
To monitor any file under /apache/ under any level of subdirectory that ends in .log:

[monitor:///apache/.../*.log]
The "..." followed by a folder separator will imply that the wildcard level folder will be excluded.

[monitor:///var/log/.../*.log]
the tailing logic will become '^/var/\log/.*[\^/]*/\.*\.$'

Therefore, /var/log/subfolder/test.log will match, but /var/log/test.log will not match and be excluded. To monitor all files in all folders use:

[monitor:///var/log/]

whitelist=\.*\.$
recursive=true

#true by default

**Wildcards and whitelisting**

Splunk Enterprise defines whitelists and blacklists with standard Perl-compatible Regular Expression (PCRE) syntax.

When you specify wildcards in a file input path, Splunk Enterprise creates an implicit whitelist for that stanza. The longest wildcard-free path becomes the monitor stanza, and Splunk Enterprise translates the wildcards into regular expressions.

Splunk Enterprise anchors the converted expression 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 Enterprise translates this into
On Windows, if you specify

```
[monitor://C:\\Windows\\foo\\bar*.log]
```

Splunk Enterprise translates it into

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

**Note:** In Windows, whitelist and blacklist rules do not support regular expressions that include backslashes. Use two backslashes (\\) to escape wildcards.

## Whitelist or blacklist specific incoming data

You can use **whitelist** and **blacklist** rules to specify which files to consume or exclude when monitoring directories. You can also apply these settings to batch inputs. When you define a whitelist, Splunk Enterprise only indexes the files you specify. When you define a blacklist, the software ignores the specified files and processes all other files. You define whitelists and blacklists in the input stanza in `inputs.conf`. If you want to apply the same blacklists and whitelists across all your forwarders, you can set up a deployment server.

It is not necessary to define both a whitelist and a blacklist in a stanza. They are independent settings. If you do define both and a file matches both, Splunk Enterprise does not index that file as blacklist overrides whitelist.

Whitelist and blacklist rules use regular expression syntax to define the match on the file name or path. They must be contained within a configuration stanza, for example `[monitor://<path>]`. The Splunk platform ignores whitelists and blacklists that are not inside a stanza. When you define whitelist and blacklist entries, you must use exact regular expression syntax.

To learn more about how to build regular expressions, see Regular-expressions.info (http://regular-expressions.info) website.

## Route and filter data

Instead of whitelisting or blacklisting your data inputs, you can filter specific events and send them to different queues or indexes.
Whitelist (allow) files

- Add the following line to your `monitor` stanza in the `/local/inputs.conf` file for the app context that you defined the input.

  
  ```
  whitelist = <your_custom_regex>
  ```

  For example, to monitor only files with the `.log` extension:

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

  **Whitelist multiple files**

  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, you can whitelist an exact match.

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

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

Blacklist (ignore) files

- Add the following line to your `monitor` stanza in the `/local/inputs.conf` file for the app context that you defined the input in.

  
  ```
  blacklist = <your_custom_regex>
  ```

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

  **Example: Blacklist only files with a `.txt` extension**

  To ignore and not monitor only files with the `.txt` extension:

  ```
  [monitor:///mnt/logs]
  blacklist = \.txt$
  ```
Example 2: Blacklist files with either a .txt or .gz extension

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)$
```

Example 3: Blacklist an entire directory

To ignore entire directories beneath a monitor input:

```
[monitor:///mnt/logs]
    blacklist = archive|historical|\.bak$
```
This example tells Splunk Enterprise to ignore all files under /mnt/logs/ within the archive or historical directories and all files ending in ".bak.

Example 4: Blacklist a file whose name contains a string

To ignore files whose names contain a specific string, you can do:

```
[monitor:///mnt/logs]
    blacklist = 2009022[89]file\.txt$
```
This example ignores the `webserver20090228file.txt` and `webserver20090229file.txt` files under /mnt/logs/.

Example 5: Blacklist Windows Event Code #4662 events whose "Message" field contains a specific value

To ignore Windows Event Code #4662 events whose Message field contains events with the value: `Account Name: "example account"`

```
[WinEventLog:Security]
    blacklist1 = EventCode = "4662" Message = "Account Name:\s+(example account)"
```
This example ignores Windows Security events with Event Code 4662 and Message values with Account Name: `example account`. 
How Splunk Enterprise handles log file rotation

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

The monitoring processor picks up new files and reads the first 256 bytes of the file. The processor then hashes this data into a begin and end cyclic redundancy check (CRC), which functions as a fingerprint representing the file content. Splunk Enterprise uses this CRC to look up an entry in a database that contains all the beginning CRCs of files it has seen before. If successful, the lookup returns a few values, but the important ones are a seekAddress, meaning the number of bytes into the known file that Splunk Enterprise has already read, and a seekCRC which is a fingerprint of the data at that location.

Using the results of this lookup, Splunk Enterprise can categorize the file.

There are three possible outcomes of a CRC check:

- No matching record for the CRC from the file beginning in the database. This indicates a new file. Splunk Enterprise picks it up and consumes its data from the start of the file. Splunk Enterprise updates the database with the new CRCs and Seek Addresses as it consumes the file.

- A matching record for the CRC from the file beginning in the database, the content at the Seek Address location matches the stored CRC for that location in the file, and the size of the file is larger than the Seek Address that Splunk Enterprise stored. While Splunk Enterprise has seen the file before, data has been added since it was last read. Splunk Enterprise opens the file, seeks to Seek Address--the end of the file when Splunk Enterprise last finished with it--and starts reading the new from that point.

- A matching record for the CRC from the file beginning in the database, but the content at the Seek Address location does not match the stored CRC at that location in the file. Splunk Enterprise has read some file with the same initial data, but either some of the material that it read has been modified in place, or it is in fact a wholly different file which begins with the same content. Because the database for content tracking is keyed to the beginning CRC, it has no way to track progress independently for the two different data streams, and further configuration is required.

Because the CRC start check runs against only the first 256 bytes of the file by default, it is possible for non-duplicate files to have duplicate start CRCs,
particularly if the files are ones with identical headers. To handle such situations you can:

- Use the `initCrcLength` attribute in `inputs.conf` to increase the number of characters used for the CRC calculation, and make it longer than your static header.
- Use the `crcSalt` attribute when configuring the file in `inputs.conf`, as described in "Monitor files and directories with inputs.conf" in this manual. The `crcSalt` attribute, when set to `<SOURCE>`, ensures that each file has a unique CRC. The effect of this setting is that Splunk Enterprise assumes that each path name contains unique content.

Do not use `crcSalt = <SOURCE>` with rolling log files, or any other scenario in which logfiles get renamed or moved to another monitored location. Doing so prevents Splunk Enterprise from recognizing log files across the roll or rename, which results in the data being reindexed.
Get data from network sources

Get data from TCP and UDP ports

You can configure Splunk Enterprise to accept an input on any TCP or UDP port. Splunk Enterprise consumes any data that arrives on these ports. Use this method to capture data from network services such as syslog (default port is UDP 514). You can also set up the netcat service and bind it to a port.

For security, Splunk Cloud accepts connections only from forwarders with the correct Secure Sockets Layer (SSL) certificates. If you want to send data from a TCP or UDP source such as syslog, use the Splunk Universal Forwarder to listen to the source and forward the data to your Splunk Cloud deployment.

TCP is the network protocol that underlies the Splunk Enterprise data distribution scheme. It is the recommended protocol for sending data from any remote host to your Splunk Enterprise server. Splunk Enterprise can index remote data from syslog-ng or any other application that transmits via TCP.

Splunk Enterprise supports monitoring over UDP, but you should use TCP to send network data instead whenever possible. UDP is not desirable as a transport because, among other reasons, it does not guarantee delivery of network packets.

When you monitor TCP network ports, the user Splunk Enterprise runs as must have access to the port you want to monitor. On many Unix operating systems, by default, you must run Splunk Enterprise as the root user to listen directly on a port below 1024.

See Working with UDP connections on the Splunk Community Wiki for recommendations if you must send network data with UDP.

Confirm how your network device handles external monitoring before you use the network monitoring input

Before you begin monitoring the output of a network device with the Splunk Enterprise network monitor, confirm how the device interacts with external network monitors.

If you configure TCP logging on some network devices, such as a Cisco Adaptive
Security Appliance (ASA), and the device cannot connect to the monitor, it might cause reduced performance or stop logging, or worse. By default, the Cisco ASA stops accepting incoming network connections when it encounters network congestion or connectivity problems.

**Add a network input using Splunk Web**

To add inputs from network ports using Splunk Web:

*Go to the Add New page*

You can get there through two routes.

**By Splunk Settings:**

1. Click **Settings**.
2. Click **Data Inputs**.
3. Choose **TCP** or **UDP**.
4. Click **New** to add an input.

**By Splunk Home:**

1. Click the **Add Data** link in Splunk Home.
2. Click **Monitor** to monitor a network port on the local machine, or **Forward** to receive network data from another machine.

**Note:** Forwarding a file requires additional setup.

3. If you selected **Forward**, choose or create the group of forwarders you want this input to apply to.
4. Click **Next**.

**Specify the network input**

1. In the left pane, click **TCP / UDP** to add an input.
2. Click the **TCP** or **UDP** button to choose between a TCP or UDP input.
3. In the Port field, enter a port number.

4. In the Source name override field, enter a new source name to override the default source value, if necessary.

**Note:** Consult Splunk Support before changing the "Source name override" value.

5. If this is a TCP input, specify whether this port should accept connections from all hosts or only one host in the Only accept connections from field. If you only want the input to accept connections from one host, enter the host name or IP address of the host. You can use wildcards to specify hosts.

6. Click Next to continue to the Input Settings page.

**Specify input settings**

The Input Settings page lets you specify source type, application context, default host value, and index. All of these parameters are optional.

1. Set the Source type. This is a default field that Splunk Enterprise adds to events and uses to determine processing characteristics, such as timestamps and event boundaries.

2. Set the Host name value. You have several choices:
   - **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.

   Learn more about setting the host value in "About hosts".

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

3. Set the Index that Splunk Enterprise should send data to for this input. 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 Enterprise has a number of utility indexes, which also appear in this dropdown box.
4. Click **Review**.

**Review your choices**

After specifying all your input settings, review your selections. Splunk Enterprise lists the options you selected, including the type of monitor, the source, the source type, the application context, and the index.

1. Review the settings.

2. If they are not what you want, click < to go back to the previous step in the wizard. Otherwise, click **Submit**.

Splunk Enterprise then loads the "Success" page and begins indexing the specified network input.

**Add a network input using the CLI**

To access the Splunk Enterprise CLI, navigate to the `$SPLUNK_HOME/bin/` directory and use the `../splunk` command.

If you get stuck, the CLI has help. Access the main CLI help by typing `splunk help`. Individual commands have their own help pages as well and can be accessed by typing `splunk help <command>`.

The following CLI commands are available for network input configuration:

<table>
<thead>
<tr>
<th>Command</th>
<th>Command syntax</th>
<th>Action</th>
</tr>
</thead>
</table>
| add     | add tcp|udp <port> [-parameter value] ... | Add inputs from <port>.
| edit    | edit tcp|udp <port> [parameter value] ... | Edit a previously added input for <port>.
| remove  | remove tcp|udp <port> | Remove a previously added data input. |
| list    | list tcp|udp [ <port>] | List the currently configured monitor. |

The `<port>` is the port number on which to listen for data. The user you run Splunk as must have access to this port.

You can modify the configuration of each input by setting any of these additional parameters:
### Table: Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sourcetype</td>
<td>No</td>
<td>Specify a sourcetype field value for events from the input source.</td>
</tr>
<tr>
<td>index</td>
<td>No</td>
<td>Specify the destination index for events from the input source.</td>
</tr>
<tr>
<td>hostname</td>
<td>No</td>
<td>Specify a host name to set as the host field value for events from the input source.</td>
</tr>
<tr>
<td>remotehost</td>
<td>No</td>
<td>Specify an IP address to exclusively accept data from.</td>
</tr>
<tr>
<td>resolvehost</td>
<td>No</td>
<td>Set to true or false (T</td>
</tr>
<tr>
<td>restrictToHost</td>
<td>No</td>
<td>Specify a host name or IP address that this input should accept connections from only.</td>
</tr>
</tbody>
</table>

### Examples

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

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

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

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

For information on best practices for using UDP, see Best practices for configuring Syslog input in the Community Wiki.

### Change restricted hosts on a TCP network input

If you decide to only accept connections from a specific host when you create a TCP input, once you save that input, you can neither change nor remove that host later, either from Splunk Web or the CLI.

To change or remove the restricted host of a port, you must first delete the input that contains the old restricted host. Then, you must add a new input that either contains the new restricted host or has no restriction.
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 the Admin 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 Enterprise uses the
defaults that are preset in $SPLUNK_HOME/etc/system/default/ (noted below).

Configure a TCP input

[tcp://<remote server>:<port>]
<attribute1> = <val1>
<attribute2> = <val2>
...
Tells Splunk Enterprise to listen to <remote server> on <port>. If <remote server>
is blank, Splunk Enterprise listens to all connections on the specified port.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>host = &lt;string&gt;</td>
<td>Sets the host key/field to a static value for this stanza. Also sets the</td>
<td>The IP address or fully-qualified</td>
</tr>
<tr>
<td></td>
<td>host key initial value. Splunk Enterprise uses the key during parsing and</td>
<td>domain name of the host where the data</td>
</tr>
<tr>
<td></td>
<td>indexing, in particular to set the host field. It also uses the host</td>
<td>originated.</td>
</tr>
<tr>
<td></td>
<td>field at search time.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The &lt;string&gt; is prepended with 'host:'.</td>
<td></td>
</tr>
<tr>
<td>index = &lt;string&gt;</td>
<td>Sets the index where Splunk Enterprise should store the events from this</td>
<td>main or whatever you set the default</td>
</tr>
<tr>
<td></td>
<td>input. The &lt;string&gt; is prepended with 'index:'.</td>
<td>index to</td>
</tr>
<tr>
<td>sourcetype =</td>
<td>Sets the sourcetype key/field for events from this input. Also declares the</td>
<td>Splunk Enterprise picks a source type</td>
</tr>
<tr>
<td>&lt;string&gt;</td>
<td>source type for this data, instead of letting Splunk Enterprise determine</td>
<td>based on various aspects of the data.</td>
</tr>
<tr>
<td></td>
<td>it. This is important both for</td>
<td>There is no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------------</td>
</tr>
</tbody>
</table>
| source = <string>  | Sets the sourcetype key initial value. Splunk Enterprise uses the key during parsing and indexing, in particular to set the source type field during indexing. Splunk Enterprise uses the source type field used search time.  

The `<string>` is prepended with 'sourcetype::'. | hard-coded default.                                                                                                                                     |
| queue = parsingQueue | 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. | parsingQueue |
| connection_host = ip | "ip" sets the host to the IP address of the remote server.  

"dns" sets the host to the DNS entry of the remote server. | ip |
Configure a TCP input over SSL

[tcp-ssl:<port>]
Use this stanza type if you receive encrypted, unparsed data from a forwarder or third-party system. Set `<port>` to the port on which the forwarder or third-party system is sending unparsed, encrypted data.

Configure a UDP input

[udp://<remote server>:<port>]
<attribute1> = <val1>
<attribute2> = <val2>
...
This type of input stanza is similar to the TCP type, except that it listens on a UDP port.

- If you specify `<remote server>`, the specified port only accepts data from that host.
- If you specify nothing for `<remote server>` - [udp://<port>] - the port accepts data sent from any host.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>host = &lt;string&gt;</td>
<td>Sets the host key/field to a static value for this stanza. Also sets the host key initial value. Splunk Enterprise uses this key during parsing and indexing, in particular to set the host field. It also uses the host field at search time. The <code>&lt;string&gt;</code> is prepended with 'host::'.</td>
<td>The IP address or fully-qualified domain name of the host where the data originated.</td>
</tr>
<tr>
<td>index = &lt;string&gt;</td>
<td>Sets the index where Splunk Enterprise should store events from this input. The <code>&lt;string&gt;</code> is prepended with 'index::'.</td>
<td>main or whatever you set the default index to</td>
</tr>
<tr>
<td>sourcetype = &lt;string&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>source = &lt;string&gt;</td>
<td>Sets the sourcetype key/field for events from this input. The <code>&lt;string&gt;</code> is prepended with 'sourcetype::'.</td>
<td>Sets the sourcetype key initial value. Splunk Enterprise uses the key during parsing and indexing, in particular to set the source type field during indexing. It also uses the source type field used search time. The <code>&lt;string&gt;</code> is prepended with 'sourcetype::'. Splunk Enterprise picks a source type based on various aspects of the data. There is no hard-coded default.</td>
</tr>
<tr>
<td>queue = parsingQueue</td>
<td>Sets the source key/field for events from this input. The <code>&lt;string&gt;</code> is prepended with 'source::'.</td>
<td>Sets where the input processor should deposit the input file path</td>
</tr>
<tr>
<td>queue = indexQueue</td>
<td>Note: Do not override the source key unless absolutely necessary. The input layer provides a more accurate string to aid in problem analysis and investigation by recording the file from which the data was retrieved. Consider use of source types, tagging, and search wildcards before overriding this value.</td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>_rcvbuf = &lt;integer&gt;</td>
<td>Sets the receive buffer for the UDP port, in bytes. If the value is 0 or negative, Splunk Enterprise ignores the value.</td>
<td>1,572,864 unless the value is too large for an OS. In this case, Splunk Enterprise halves the value from this default continuously until the buffer size is at an acceptable level.</td>
</tr>
<tr>
<td>no_priority_stripping = true</td>
<td>Sets how Splunk Enterprise handles receiving syslog data. If you set this attribute to true, Splunk Enterprise does not strip the <code>&lt;priority&gt;</code> syslog field from received events. Depending on how you set this attribute, Splunk Enterprise also sets event timestamps differently. When set to true, Splunk Enterprise honors the timestamp as it comes from the source. When set to false, Splunk Enterprise assigns events the local time.</td>
<td>false (Splunk Enterprise strips <code>&lt;priority&gt;</code>.)</td>
</tr>
<tr>
<td>no_appending_timestamp = true</td>
<td>Sets how Splunk Enterprise applies timestamps and hosts to events. If you set this attribute to true, Splunk Enterprise does not append a timestamp and host to received events.</td>
<td>false (Splunk Enterprise append timestamps and hosts to events)</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>Do not set this attribute if you want to append timestamp and host to received events.</td>
<td></td>
</tr>
</tbody>
</table>

**UDP packets and line merging**

Splunk Enterprise does not index each UDP packet as an independent event. Instead, it performs event merging on the data stream and merges events together if they don't have a clear timestamp.

You can avoid this problem by editing the underlying source type in `props.conf` and setting the `SHOULD_LINEMERGE` attribute to `false`. This keeps Splunk Enterprise from merging packets together.

**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,

**How Splunk Enterprise handles syslog data over UDP**

Splunk Enterprise can act as a syslog server or a syslog message sender. If you have Splunk Cloud, you cannot configure your deployment as a syslog server or a syslog message sender, but you can configure the Splunk Universal Forwarder to listen on a UDP network port and forward data to your Splunk Cloud deployment.

While it is possible to configure Splunk Enterprise to receive syslog events directly, refrain from doing so for the following reasons:

- Splunk best practice involves setting up a separate machine that runs a syslog service to handle syslog tasks
- Splunk Enterprise modifies syslog data by default as part of the indexing process (it assigns a timestamp and a host to the event.)
- Syslog data streams to only one Splunk Enterprise instance in this scenario. In a deployment with multiple indexers, you must perform additional work to distribute the data across those indexers
If Splunk Enterprise fails for any reason, any syslog messages that were sent during the downtime would be irrecoverably lost.

Do not substitute Splunk Enterprise for a syslog server in regular use unless you have no other options.

If you must retain raw syslog data (for example, a data retention policy requires access to untouched events), consider using a tool such as syslog-ng to simultaneously save the raw data to a log file and forward events to your Splunk deployment. This gives you the added advantage of indexing the log file later if you want.

See the diagrams later in this topic for a description of how Splunk Enterprise handles syslog events over UDP.

**How Splunk Enterprise handles syslog inputs**

When you configure a UDP network input to listen to a syslog in Splunk Enterprise, any syslog events that arrive through the input receive a timestamp and connected host field. Splunk Enterprise prepends these fields to each event before indexing.

You can change this behavior by setting the `no_appending_timestamp` attribute in `inputs.conf`.

If the data contains a syslog header, Splunk Enterprise strips it out unless you set the `no_priority_stripping` attribute in the stanza.

Splunk Enterprise does not modify TCP packets in this fashion. If you send syslog data over TCP, Splunk Enterprise does not strip priority information from the events. It does, however, prepend a host name and time stamp to the event unless you tell it not to.

**How Splunk Enterprise handles syslog outputs**

Splunk Enterprise can also forward events to another syslog server. When it does, it prepends the priority information to the event so that the downstream syslog server can translate the events properly.

When the event reaches the downstream syslog server, that host prepends a timestamp, priority, and connected host name, which is the Splunk Enterprise instance.
You can also prepend a timestamp and host name to the event at the time you forward the event to the syslog server.

For information on configuring routing, filtering, and usage of source types, see Route and filter data in the *Forwarding Data* manual and the props.conf spec file in the *Admin* manual.

**How Splunk Enterprise moves syslog events when you configure it to use syslog source type**

The following diagram shows how Splunk Enterprise moves two syslog messages from one syslog server to another. In the diagram, Splunk Enterprise listens on a UDP network port and indexes incoming events. On the other side, the same instance forwards events to a second, third-party syslog server.
In the diagram, Message A originates as a syslog event and Message B originates as a similar event that does not have priority information associated with it. Upon receipt, Splunk Enterprise tags the events with a timestamp and the host that generated the event.

If you configured the instance as a forwarder, Splunk Enterprise then transforms the events by adding a priority header (that you specify in `outputs.conf`) before it forwards the events on to the syslog server. Once they arrive at the syslog server, that server prepends timestamp and host data to the events as it received them from the Splunk Enterprise instance.

**How Splunk Enterprise moves syslog events when you configure a custom source type**

In this diagram, Splunk Enterprise has been configured to use a non-syslog source type.

The initial Messages A and B are identical to the first example. In this example, Splunk Enterprise prepends the event with an originating host name or IP address.
How Splunk Enterprise moves syslog events when you configure it with timestamping

You can also configure Splunk Enterprise to add timestamps to syslog events when you forward those events. You could time stamp the events when you don’t want the downstream server to add its own timestamp. The following diagram shows the required attribute and depicts how Splunk Enterprise deals with the data.

The initial Messages A and B are identical to the first and second examples. Splunk Enterprise prepends the events with a timestamp and an originating host name or IP address.
Send SNMP events to your Splunk deployment

Simple Network Management Protocol (SNMP) traps are alerts that remote devices send out. This topic describes how to send SNMP traps to a Splunk deployment.

**Note:** The procedures shown in this topic (for both *nix and Windows) are examples only. There are a number of ways to send SNMP traps to a Splunk deployment. For example, instead of using Net-SNMP, you can use other tools, such as Snare or SNMPGate, to write SNMP traps to files that you can monitor.

### How to index SNMP traps

For Splunk Enterprise, the most effective way to index SNMP traps is to write them to a file on the Splunk Enterprise server and configure Splunk Enterprise to monitor the file. If you have Splunk Cloud, write the data to a file that is monitored by the Splunk Universal Forwarder.

To configure Splunk Enterprise to consume a SNMP trap data:

1. Configure the remote devices to send their traps directly to the Splunk Enterprise instance IP address. The default port for SNMP traps is `udp:162`.

2. Write the SNMP traps to a file on the Splunk Enterprise instance, as described in "Write SNMP traps to a file on the Splunk Enterprise server."

3. Configure Splunk Enterprise to monitor the file, as described in "Monitor files and directories".

**Note:** This topic does not cover SNMP polling, which is a way to query remote devices.

### Write SNMP traps to a file on the Splunk Enterprise instance

Use your favorite SNMP software to write the SNMP traps to a file. For information about available SNMP software, visit the SNMP portal (http://www.snmplink.org) website.

**For *nix**

On *nix, you can use the Net-SNMP project `snmptrapd` binary to write SNMP traps to a file.
Before you install `snmptrapd` on your system, see the local documentation for the version of `snmptrapd` that comes with your distribution of *nix. See also the manual page for `snmptrapd`.

The simplest configuration is:

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

**Note:** Versions 5.3 and later of `snmptrapd` apply access control checks to all incoming notifications instead of accepting and logging them automatically (even if no explicit configuration was provided). If you run `snmptrapd` without suitable access control settings, then it does not process those traps. You can avoid this by specifying:

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

To see the version of `snmptrapd`, run `snmptrapd --version` from the command prompt.

**Troubleshoot problems with SNMP**

If you experience problems sending SNMP traps to your Splunk deployment, consider that:

- UDP port 162 is a privileged network port. If you need to use this port, then you must either run `snmptrapd` as root or specify `snmptrapd` with a port that is higher than 1024.
- You can use the `-f` flag to keep `snmptrapd` in the foreground while testing.
- You can use the `-Lo` flags 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
```

**For Windows**

To log SNMP traps to a file on Windows.

1. Download and install the latest version of NET-SNMP for Windows from the NET-SNMP website.

**Note:** The OpenSSL library must not be installed on the system because it conflicts with NET-SNMP.
2. Register `snmptrapd` as a service using the script included in the NET-SNMP install.

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

```plaintext
snmpTrapdAddr [System IP]:162
authCommunity log [community string]
```

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

### Use Management Information Bases (MIBs)

Management Information Bases (MIBs) provide a map between numeric object IDs (OIDs) reported by the SNMP trap and a textual human readable form. Though `snmptrapd` can work without any MIB files at all, it won't display the results in exactly the same way.

The vendor of the device you receive SNMP traps from can provide a specific MIB. For example, all Cisco device MIBs can be located using the online Cisco SNMP Object Navigator.

To add a new MIB file:

1. Download and copy the MIB file into the MIB search directory. On the *nix version of Net-SNMP, the default location is `/usr/local/share/snmp/mibs`. You can set a different directory by providing the `-m` argument to `snmptrapd`.

2. Instruct `snmptrapd` to load the MIB(s) by passing a colon-separated list to the `-m` argument.

**Note:**

- If you add a leading `+' character for the parameters in the `-m` argument, `snmptrapd` loads the MIB in addition to the default list, instead of overwriting the list.
- The special keyword `ALL` tells `snmptrapd` to load all MIB modules in the MIB directory.

For example, to load all MIB modules in the MIB directory:

```plaintext
snmptrapd -m +ALL
```
Get Windows data

Monitoring Windows data with Splunk Enterprise

Splunk software can index many different kinds of Windows data. This data can be pretty much anything: an Event Log channel, the Registry, or Active Directory. You also have available the standard set of Splunk inputs, such as files and directories, the network monitoring inputs, and scripted inputs.

The following specialized inputs are available only on Windows installations of Splunk Enterprise. If you have Splunk Cloud and want to monitor these inputs, use the Splunk Universal Forwarder.

- **Windows Event Logs.** Monitor events generated by the Windows Event Log service on any available event log channel on the machine. You can collect events on the local machine or remotely by using either a universal forwarder or Windows Management Instrumentation (WMI).

- **Performance monitoring.** Collect performance data on Windows machines with Splunk Enterprise and then alert or report on that data. Any performance counter that is available in Performance Monitor is also available to Splunk Enterprise. You can monitor performance locally or remotely through a universal forwarder or WMI.

- **Remote monitoring over WMI.** Splunk Enterprise can use WMI to access event log and performance data on remote machines.

- **Registry monitoring.** You can monitor changes to the local Windows Registry using the Registry monitoring capability. You can use a universal forwarder to gather Registry data from remote machines.

- **Active Directory monitoring.** Splunk Enterprise can audit any changes to the Active Directory including changes to user, group, machine, and group policy objects. You can forward Active Directory data to another Splunk Enterprise server.
The Splunk App for Windows Infrastructure

The Splunk App for Windows Infrastructure provides data inputs, searches, reports, alerts, and dashboards for Windows server and desktop management. You can monitor, manage, and troubleshoot Windows operating systems from one place. The app includes inputs for CPU, disk I/O, memory, event logs, configurations, and user data, plus a web-based setup UI for indexing Windows event logs.

Initial considerations for deploying Splunk Enterprise on Windows

When you install and deploy Splunk Enterprise on Windows, consider the following:

- **Authentication.** To perform any operations on remote Windows machines in your network, Splunk Enterprise must run as a user with credentials to access those machines. Make these credentials available before deploying. See "Considerations for deciding how to monitor remote Windows data."

- **Disk bandwidth.** Splunk Enterprise indexers require lots of disk I/O bandwidth, particularly when indexing large amounts of data. Make sure that you configure any installed antivirus software to avoid monitoring Splunk Enterprise directories or processes, because such scans significantly reduce performance.

- **Shared hosts.** Before you install Splunk Enterprise on a host that runs other services, such as Exchange, SQL Server, or a hypervisor, see Introduction to capacity planning for Splunk Enterprise in the Capacity Planning manual.

The most efficient way to gather data from any Windows server is to install universal forwarders on the hosts that you want to gather data. Universal forwarders use limited resources. In some cases, such as Registry monitoring, you must use a forwarder, because you cannot collect Registry data over WMI.

How to get Windows data into your Splunk deployment
You can collect the following Windows data with Splunk software:

- Windows Event Logs
- File system changes
- Active Directory
- data over the Windows Management Instrumentation (WMI) infrastructure
- Registry data
- Performance metrics
- Host information
- Print information
- Network information

Since only Windows machines provide this data, only the Windows version of Splunk Enterprise can get the data. Other operating systems cannot collect Windows data directly. You can send Windows data from Windows machines to Splunk Enterprise instances that do not run Windows. If you have Splunk Cloud and want to monitor these inputs, the Splunk Universal Forwarder is your only option.

How Splunk Enterprise interacts with Windows modular and scripted inputs on start-up and shutdown

When you configure a scripted or modular Windows data input in Splunk Enterprise, the `splunkd` service sends a signal to the input to begin collecting the data. Similarly, when you shut Splunk Enterprise down cleanly, the service sends a different signal to the inputs to tell them to stop collecting data, clean up, and exit.

The following table lists the control messages that the `splunkd` service sends to modular and scripted Windows inputs during start-up and shutdown.

Control messages (signals) sent by the `splunkd` service to Windows modular and scripted inputs

<table>
<thead>
<tr>
<th>Start-up</th>
<th>Shutdown</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateProcess</td>
<td>CTRL_BREAK_EVENT</td>
</tr>
</tbody>
</table>

Use Splunk Web to collect Windows data

Nearly all Windows inputs let you use the Splunk Web interface to get the data. The exception is the `MonitorNoHandle` input, which you must set up with a configuration file.
1. Log into your Splunk deployment.
2. Click **Settings** in the upper right corner, then click **Data inputs**. The **Data inputs** page appears.
3. Find the Windows input that you want to add in the list of available inputs by clicking **Add new** in the Actions column for the input.
4. Follow the instructions in the subsequent pages for the input type you selected.
5. Click **Save**. In most cases, data collection begins immediately.

### Use configuration files to collect Windows data

In cases where you cannot use Splunk Web to configure Windows inputs, such as when you use a universal forwarder to collect the data, you must use configuration files (the universal forwarder installer on Windows lets you configure some Windows inputs at installation time.)

Configuration files offer more control over Splunk Web in many cases. Some inputs can only be configured this way.

1. Open a command prompt or PowerShell window,
2. Change to the `%SPLUNK_HOME%\etc\system\local` directory.
3. Edit `inputs.conf` in this directory. You might need to create this file.
4. Add inputs to the `inputs.conf` file by defining input stanzas.
5. Save the file and close it.
6. Restart the Splunk instance. The software reloads the configuration files and begins collecting data based on the new configuration.

### Considerations for deciding how to monitor remote Windows data

This topic discusses the considerations for monitoring remote Windows data.

### Remote Windows data overview

Splunk Enterprise collects remote Windows data for indexing in one of two ways:

- from Splunk forwarders
- via Windows Management Instrumentation (WMI)

For Splunk Cloud deployments you must use the Splunk Universal Forwarder to monitor remote Windows data.
**Use a forwarder to collect remote Windows data**

Use a universal forwarder to get remote Windows data whenever possible. The universal forwarder has these advantages:

- It uses minimal network and disk resources on the installed machines.
- You can install it as a non-privileged user, whereas you require administrative access for WMI.
- If you install it as the Local System user, then it has administrative access to the machine and requires no authentication to get data from there, as WMI does.
- It scales well in large environments and is easy to install. You can install it manually, with either a Microsoft deployment tool like System Center Configuration Manager (SCCM) or a third party distribution solution such as Puppet or IBM BigFix.

After you install a universal forwarder, it gathers information locally and sends it to a Splunk deployment. You tell the forwarder what data to gather either during the installation process or later, by distributing configuration updates manually or with a deployment server. You can also install add-ons into the universal forwarder.

There are some drawbacks to using the universal forwarder, depending on your network configuration and layout. See "Forwarders versus remote collection through WMI" in this topic.

**Use WMI to collect remote Windows data**

The Windows Management Instrumentation (WMI) framework lets Splunk Enterprise collect virtually any kind of data from remote Windows machines. In this configuration, Splunk Enterprise runs as a user that you specify at installation (or later on, in the Services control panel).

This configuration:

- Gives Splunk Enterprise as much access to the network as the specified account has for remote access.
- Lets indexers collect data from remote Windows machines across the enterprise and place that data into a central repository.
- Is ideal for small to medium-sized networks with at least one indexer in each network segment
There are some caveats to this method of collection. See Forwarders versus WMI in this topic.

Also, while Active Directory (AD) monitoring does not use WMI, it has the same authentication considerations as data inputs that do use it. For information on how Splunk Enterprise monitors AD, see Monitor Active Directory in this manual.

**Considerations for getting data over WMI**

When collecting remote Windows data over WMI, consider the following:

**Authentication for remote Windows data**

Windows requires authentication for remote operations. Failure to understand how Splunk Enterprise interacts with Windows over the network can lead to suboptimal search results, or none at all. This section provides guidelines on security for collecting remote Windows data.

When you install Splunk Enterprise, you can specify that it run as the Local System user, or another user. This choice has ramifications for both installation and data collection.

The user you tell Splunk Enterprise to run as determines the kind of data it can retrieve from remote machines. To get the data you want, you must provide an appropriate level of permission to this user.

In most cases, configure the Splunk Enterprise user account with "least-permissive" access to the data sources you want to collect. This entails:

- Adding the user to various domain security groups.
- Making changes to the access control lists of various AD objects, depending on the data sources you need to access.

If your AD domain security policy enforces password changes regularly, you must also:

- Confirm that either the Splunk Enterprise user password never expires, or that you manually change the password before it expires, as defined by the password policy.
- Restart Splunk services that run as that account on all hosts in your network, once you change the password.
You should also assign the Splunk Enterprise account the "Deny log on locally" user rights assignment in Local Security Policy to prevent the user from logging in interactively to workstations. This method gives you more control and is more secure than handing out domain administrator access.

Individual Getting Data In topics in this manual that deal with remote access to Windows machines contain additional information and recommendations on how to configure the user Splunk Enterprise runs as for least-permissive access. Review the "Security and remote access considerations" section on those pages.

**Use managed system accounts to access Windows data**

On recent versions of Windows Server, you can use managed service accounts (MSAs) to address challenges with password expiry. See Managed service accounts on Windows Server 2008 and Windows 7 in the *Installation* manual.

**Network and I/O usage considerations**

Monitor network bandwidth usage closely, especially in networks with slow or thin WAN links. For this reason alone, universal forwarders are a better choice for large remote data collection operations.

Disk bandwidth is a concern as well. Anti-virus scanner drivers and drivers that intermediate between Splunk Enterprise and the operating system should always be configured to ignore the Splunk Enterprise directory and processes, regardless of the type of installation.

**Splunk forwarders versus WMI**

Use a universal forwarder to get data in from a remote Windows host. A universal forwarder offers the most types of data sources, provides more detailed data (for example, in performance monitoring metrics), minimizes network overhead, and reduces operational risk and complexity. It is also more scalable than WMI in many cases.

In circumstances where you collect data remotely (such as when corporate or security policy restricts code installation, or there are performance or interoperability concerns,) you can use the native WMI interface to collect event logs and performance data.

These are the main areas of tradeoff between WMI and forwarders:

- Performance
• Deployment
• Management

Performance

With respect to performance, a forwarder is a better choice when:

• You collect local event logs or flat files. A forwarder requires less CPU and performs basic precompression of the data in an effort to reduce network overhead.
• You want to collect data from a machine without having to worry about authentication. When you install a forwarder as the Local System user, it has administrative access to the machine, letting you collect any data from it.
• You want to collect data from busy hosts such as AD domain controllers or machines that consistently experience periods of high utilization, such as Exchange, SQL Server/Oracle, VMWare, Hyper-V, or SharePoint servers. This is because WMI might have problems keeping up with the amount of data these services generate. WMI polling is best-effort by design, and Splunk Enterprise also throttles WMI calls to prevent unintentional denial-of-service attacks.
• You are concerned about CPU and network utilization. Forwarders use as little of these resources as possible, while WMI uses more CPU and network resources to transfer data.
• You are concerned about scalability. Universal forwarders scale very well. Heavy forwarders do not scale as well as universal forwarders, but both types of forwarder scale considerably better than WMI.

WMI is a better choice when you have concerns about memory usage on a system with high memory utilization. Because forwarders have more polling options available, and reside on the local machine while collecting data, they use more memory than WMI does.

Deployment

A forwarder is a better choice for deployment when:

• You have control of the base build of the OS, as is the case when you create system images.
• You have many data sources to collect, particularly if the data requires transformation of any kind.
**Note:** Except for a few cases, you cannot use a universal forwarder to process data before it reaches the indexer. If you need to make any changes to your data before you index it, you must use a heavy forwarder.

WMI is a better choice when:

- You don't have control of the base OS build, or you don't have domain administrator access, or local administrator privileges on the machines from which you want to get data.
- You want or need only a limited set of data from a large number of hosts (for example, CPU data for usage billing).

A common deployment scenario is to first test using remote polling, then add successful or useful data inputs to your forwarder configurations later, or when you do large scale forwarder installations.

**Management**

Both mechanisms offer logging and alerting to advise if a host comes on or offline or is unreachable. To prevent an unintentional denial of service attack, the WMI polling service in Splunk Enterprise polls less frequently over time if it cannot contact a host, and eventually stops polling unreachable hosts altogether. Do not use remote polling over WMI for machines that are frequently offline, such as laptops or dynamically provisioned virtual machines.

The table shows a list of data sources and indicates which data collection type(s) are appropriate for each data source.

**Data sources and collection methods**

<table>
<thead>
<tr>
<th>Data source</th>
<th>Local forwarder</th>
<th>WMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event logs</td>
<td>Yes</td>
<td>Yes*</td>
</tr>
<tr>
<td>Performance</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Registry</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Active Directory</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Log files</td>
<td>Yes</td>
<td>Yes**</td>
</tr>
<tr>
<td>Crawl</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

* For remote event log collection, you must know the name of the event log you want to collect. On local forwarders, you have the option to collect all logs, regardless of name.
** Splunk Enterprise supports remote log file collection using the \SERVERNAME\SHARE syntax; however, you must use CIFS (Common Internet File System, or Server Message Block) as your application layer file access protocol, and Splunk Enterprise must have at least read access to both the share and the underlying file system.

Search Windows data on a non-Windows instance of Splunk Enterprise

You can index and search your Windows data on a non-Windows Splunk deployment, but you must first use a Windows instance of Splunk Enterprise to get the Windows data. You can do this by installing a Splunk forwarder onto the Windows computer and configuring it to forward Windows data to the non-Windows instance of Splunk Enterprise.

There are two ways to proceed:

- Set up forwarders locally on each Windows machine that you want data. These forwarders can send the Windows data to the non-Windows receiving instance.
- Set up a forwarder on a separate Windows machine. The forwarder can use WMI to collect data from all the Windows machines in the environment and then forward the combined data to a non-Windows receiving instance of Splunk.

Monitor Active Directory

The Active Directory (AD) database (also known as the NT Directory Service (NTDS) database) is the central repository for user, computer, network, device and security objects in an AD domain or forest. You can use Splunk Enterprise to record changes to AD, such as the addition or removal of a user, host, or domain controller (DC). If you have Splunk Cloud, you must use the Splunk Universal forwarder to collect Active Directory data.

You can configure AD monitoring to watch for changes to your Active Directory forest and collect user and machine metadata. You can use this feature combined with dynamic list lookups to decorate or modify events with any information available in AD.

After you have configured Splunk Enterprise to monitor your Active Directory, it takes a baseline snapshot of the AD schema. It uses this snapshot to establish a
starting point for monitoring.

The AD monitoring input runs as a separate process called splunk-admon.exe. It runs once for every Active Directory monitoring input you define in Splunk Enterprise.

Why monitor Active Directory?

If you maintain the integrity, security, and health of your Active Directory, then what happens with it day to day is a concern. Splunk Enterprise lets you monitor what and when things changed in your AD, and who changed them.

You can transform this data into reports for corporate security compliance or forensics, for example. You can also use the data retrieved for intrusion alerts for immediate response. Additionally, you can create health reports with the data indexed for future AD infrastructure planning activities, such as assignment of operations master roles, AD replicas, and global catalogs across DCs.

What do you need to monitor Active Directory?

The following table lists the permissions you must have to monitor an Active Directory schema.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Required permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor an Active Directory schema</td>
<td>* Splunk Enterprise must run on Windows</td>
</tr>
<tr>
<td></td>
<td>* Splunk Enterprise must run as a domain user</td>
</tr>
<tr>
<td></td>
<td>* The user Splunk Enterprise runs as must have read access to all AD objects that you want to monitor</td>
</tr>
</tbody>
</table>

Technical considerations for monitoring Active Directory

For best results with monitoring AD, read and understand the following:

- The AD monitor is only available on Splunk Enterprise on Windows.
- While you cannot monitor AD changes from a *nix version of Splunk Enterprise, you can forward AD data from a Windows version of Splunk Enterprise to a *nix indexer.
- The AD monitoring process can run under a full instance or on any kind of forwarder.
- The host that monitors changes to AD must belong to the domain or forest you want to monitor.
- The user that Splunk Enterprise runs as must also be part of the domain.
• The permissions that the user has determine which parts of AD Splunk can monitor.

For information on deciding which user Splunk should run as at installation time, see Choose the user Splunk should run as in the Installation Manual.

How the AD monitor interacts with AD

When you set up an AD monitoring input, the input connects to an AD domain controller to authenticate and, if necessary, perform any security ID (SID) translations while it gathers the AD schema or change events.

The AD monitor uses the following logic to interact with Active Directory after you set it up:

1. If you specify a domain controller when you define the input (either with the targetDc setting in inputs.conf or the "Target domain controller" field in Splunk Web, then the input uses that domain controller for AD operations.
2. If you do not specify a domain controller, then the input does the following:
   1. The input attempts to use the local system cache to authenticate or resolve SIDs.
   2. If the monitor cannot authenticate or resolve SIDs that way, it attempts a connection to the domain controller that the machine that runs the input used to log on.
   3. If that does not work, then the input attempts to use the closest AD domain controller that has a copy of the Global Catalog.
3. If the domain controller that you specify is not valid, or a domain controller cannot be found, then the input generates an error message.

The AD monitor does not chase LDAP referrals

If the AD monitor makes an LDAP query and receives a referral, it does not chase this referral to complete the query. An LDAP referral represents a problem with your LDAP configuration and you or your designated administrators should determine and fix the configuration problem within AD.

Configure Active Directory monitoring

You can configure AD monitoring either in Splunk Web or by editing configuration files. More options, such as the ability to configure monitors for multiple DCs, are available when using configuration files.
Configure AD monitoring with Splunk Web

Go to the Add New page

You can get there by two routes.

- Splunk Home
- Splunk Settings

By Splunk Settings:

1. Click Settings in the upper right corner of Splunk Web.
2. Click Data Inputs.
3. Click Active Directory monitoring.
4. Click New to add an input.

By Splunk Home:

1. Click the Add Data link in Splunk Home.
2. Click Monitor to monitor Active Directory on the local Windows machine.

Select the input source

1. In the left pane, select Active Directory monitoring.
2. In the Collection name field, type in a unique name for the input that you will remember.
3. (Optional) In the Target domain controller field, enter the host name or IP address of the domain controller you want to use to monitor AD.
4. (Optional) In the Starting node field, type in the Active Directory node you want the input to begin monitoring from. Use the Lightweight Directory Access Protocol (LDAP) format, for example, DC=Splunk-Docs,DC=com.
5. (Optional) You can click the Browse button to browse through a list of available Active Directory nodes to browse through a list of available AD domains.
6. Check Monitor Subtree if you want the input to monitor all sub-nodes of the node you entered in the "Starting node" field.
7. Click Next.

Specify input settings

The Input Settings page lets you specify application context, default host value, and index. All of these parameters are optional.
**Note:** Host only sets the host field in the resulting events. It does not tell the input to look on a specific host on your network.

1. Select the appropriate Application context for this input.
2. Set the Host name value. You have several choices for this setting. Learn more about setting the host value in "About hosts".
3. Set the Index that Splunk Enterprise should send data to. 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 Enterprise has a number of utility indexes, which also appear in this dropdown box.
4. Click Review.

**Review your choices**

After specifying all your input settings, review your selections. Splunk Enterprise lists all options you selected, including the type of monitor, the source, the source type, the application context, and the index.

1. Review the settings.
2. If they do not match what you want, click < to go back to the previous step in the wizard. Otherwise, click **Submit**.

Splunk Enterprise then loads the "Success" page and begins indexing the specified Active Directory node.

**Configure AD monitoring with configuration files**

The inputs.conf configuration file controls Active Directory monitoring configurations. Edit copies of inputs.conf in the %SPLUNK_HOME%/etc/system/local directory. If you edit them in the default directory, your changes are overwritten when you upgrade. For more information about configuration file precedence, see "Configuration file precedence" in this manual.

1. Open %SPLUNK_HOME%/etc/system/local/inputs.conf for editing. You might need to create this file if it does not exist.
2. Add the appropriate AD monitoring stanzas and settings.

By default, when you enable AD monitoring inputs, Splunk Enterprise gathers AD change data from the first domain controller that it can attach to. If that is acceptable, no further configuration is necessary.
**inputs.conf settings**

`inputs.conf` contains one stanza for each AD monitoring input, with a header like the following:

```
[admon://<name of stanza>]
```

In each stanza, you can specify:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required?</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>targetDc</td>
<td>Yes</td>
<td>The unique name of the domain controller you want to use for AD monitoring.</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specify a unique name for this attribute if:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• You have a very large AD and you only want to monitor information in a</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>particular Organizational Unit (OU), subdomain, etc.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• You have a specific (read-only) domain controller that can be used for</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>monitoring purposes in a high security environment.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• You have multiple domains or forests in with transitive trusts established,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>and want to target a different tree than the one where the host that</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>runs Splunk Enterprise resides.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• You want to configure multiple AD monitoring inputs to target multiple</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>domain controllers. For example, to monitor AD replication across a</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>distributed environment.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>To target multiple DCs, add another [admon://&lt;uniquename&gt;targetDc] stanza</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>for a target in that tree.</td>
<td></td>
</tr>
<tr>
<td>startingNode</td>
<td>No</td>
<td>A fully qualified Lightweight Directory Access Protocol (LDAP) name (for</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>example: &quot;LDAP://OU=Computers,DC=ad,DC=splunk,DC=com&quot; that specifies</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>where in the AD tree that Splunk Enterprise should begin its indexing.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The software starts there and enumerates down to sub-containers,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>depending on the configuration of</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The highest root domain in the tree that Splunk</td>
<td></td>
</tr>
</tbody>
</table>
The value of `startingNode` must be within the scope of the DC you are targeting for Splunk Enterprise to get AD data.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required?</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>monitorSubtree</td>
<td>No</td>
<td>How much of the target AD container to index. A value of 0 means to index only the target container, and not traverse into subcontainers within that container. A value of 1 means to enumerate all sub-containers and domains that it has access to.</td>
<td>1 (monitor all domains that Splunk Enterprise has access to)</td>
</tr>
<tr>
<td>baseline</td>
<td>No</td>
<td>Whether or not the input enumerates all existing available AD objects when it first runs. A value of 0 means not to set a baseline. A value of 1 means to set a baseline.</td>
<td>1 (set the baseline.)</td>
</tr>
<tr>
<td>index</td>
<td>No</td>
<td>The index to route AD monitoring data to.</td>
<td>the 'default' index</td>
</tr>
<tr>
<td>disabled</td>
<td>No</td>
<td>Whether or not the Splunk should run the input. A value of 0 means that the input is enabled, and a value of 1 means that the input is disabled.</td>
<td>0 (enabled).</td>
</tr>
</tbody>
</table>

**Example AD monitoring configurations**

The following are examples of how to use `inputs.conf` to monitor desired portions of your AD network.

To index data from the top of the AD directory:

```plaintext
#Gather all AD data that this server can see

[admon://NearestDC]
targetDc =
startingNode =
```

To use a DC that is at a higher root level than an OU you want to target for monitoring:

```plaintext
# Use the pri01.eng.ad.splunk.com domain controller to get all AD
```
metadata for
# the Computers OU in this forest. We want schema data for the entire AD
tree, not
# just this node.

[admon://DefaultTargetDc]
targetDc = pri01.eng.ad.splunk.com
startingNode = OU=Computers,DC=eng,DC=ad,DC=splunk,DC=com

To monitor multiple domain controllers:

# Get change data from two domain controllers (pri01 and pri02) in the
# same AD tree.
# Index both and compare/contrast to ensure AD replication is occurring
# properly.

[admon://DefaultTargetDc]
targetDc = pri01.eng.ad.splunk.com
startingNode = OU=Computers,DC=eng,DC=ad,DC=splunk,DC=com

[admon://SecondTargetDc]
targetDc = pri02.eng.ad.splunk.com
startingNode = OU=Computers,DC=eng,DC=ad,DC=splunk,DC=com

Sample AD monitoring output

When the Splunk AD monitoring utility runs, it gathers AD change events, which
are then indexed by Splunk software. To view these events as they arrive, use
the Search app.

There are several types of AD change events that Splunk software can index.
Examples of these events follow. Some of the content of these events has been
obscured or altered for publication purposes.

Update event

When an AD object changes, Splunk generates an update event. The software
logs this change as type `admonEventType=Update`.

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
Delete event

When an AD object has been marked for deletion, Splunk software generates a delete event. The event type is similar to `admonEventType=Update`, except that it contains the `isDeleted=True` key/value pair at the end of the event.
Sync event

When AD monitoring inputs are configured, Splunk software tries to capture a baseline of AD metadata when it starts. Splunk software generates event type `admonEventType=Sync`, which represents the instance of one AD object and all its field values. Splunk software tries to capture all of the objects from the last recorded Update Sequence Number (USN).

When you restart Splunk Enterprise or the `splunk-admon.exe` process, the software logs an extra 'sync' event. This is normal.
When you restart Splunk Enterprise after configuring it for AD monitoring, it generates a schema type event: `admonEventType=schema`. This event shows the definitions of every object in the Active Directory structure. The available, required and optional fields are listed for each AD object. Failure to see all of these fields can indicate a problem with Active Directory.
Monitor Windows event log data

Windows generates log data during the course of its operation. The Windows Event Log service handles nearly all of this communication. It gathers log data published by installed applications, services and system processes and places them into event log channels. Programs such as Microsoft Event Viewer subscribe to these log channels to display events that have occurred on the system.
Splunk Enterprise can monitor event log channels and files stored on the local machine, and it can collect logs from remote machines. The event log monitor runs as an input processor within the `splunkd` service. It runs once for every event log input that you define in Splunk Enterprise. If you have Splunk Cloud and want to monitor event log channels, use the Splunk Universal Forwarder to collect the data and forward it to your Splunk Cloud deployment.

New for versions 6.4.5 and later of Splunk Enterprise, the Windows Event Log monitoring input has improved performance.

**Why monitor event logs?**

Windows event logs are the core metric of Windows machine operations - if there is a problem with your Windows system, the Event Log service has logged it. Splunk Enterprise indexing, searching, and reporting capabilities make your logs accessible.

**Prerequisites to monitoring event logs**

<table>
<thead>
<tr>
<th>Activity</th>
<th>Required permissions:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor local event logs</td>
<td>Splunk Enterprise must run on Windows&lt;br&gt;Splunk Enterprise must run as the Local System user to read all local event logs</td>
</tr>
<tr>
<td>Monitor remote event logs</td>
<td>Either Splunk Enterprise or a universal forwarder must run on the Windows machine from which you want to collect event logs&lt;br&gt;Splunk Enterprise or a universal forwarder must run as a domain or remote user with read access to Windows Management Instrumentation (WMI) on the target machine&lt;br&gt;The user that Splunk Enterprise or the universal forwarder runs as must have read access to the event logs you want to collect</td>
</tr>
</tbody>
</table>

**Security and remote access considerations**

Splunk Enterprise collects event log data from remote machines using either WMI or a universal forwarder. Splunk best practice is to use a universal forwarder to send event log data from remote machines to an indexer. See The universal forwarder in the *Universal Forwarder* manual for information about how to install, configure and use the forwarder to collect event log data.
To install forwarders on your remote machines to collect event log data, you can install the forwarder as the Local System user on these machines. The Local System user has access to all data on the local machine, but not on remote machines.

To use WMI to get event log data from remote machines, you must ensure that your network and Splunk instances are properly configured. You cannot install the Splunk platform as the Local System user, and the user you install with determines the event logs Splunk software sees. See Security and remote access considerations in Monitor WMI-based data for additional information on the requirements you must satisfy to collect remote data properly using WMI.

By default, Windows restricts access to some event logs depending on the version of Windows you run. For example, only members of the local Administrators or global Domain Admins groups can read the Security event logs by default.

**How the Windows Event Log monitor interacts with Active Directory (AD)**

When you set up an Event Log monitoring input for WMI, the input connects to an AD domain controller to authenticate and, if necessary, perform any security ID (SID) translations before it begins to monitor the data.

The Event Log monitor uses the following logic to interact with AD after you set it up:

1. If you specify a domain controller when you define the input (with the `evt_dc_name` setting in `inputs.conf`), then the input uses that domain controller for AD operations.
2. If you do not specify a domain controller, then the input does the following:
   1. The input attempts to use the local system cache to authenticate or resolve SIDs.
   2. If the monitor cannot authenticate or resolve SIDs that way, it attempts a connection to the domain controller that the machine that runs the input used to log on.
   3. If that does not work, then the input attempts to use the closest AD domain controller that has a copy of the Global Catalog.
3. If the domain controller that you specify is not valid, or a domain controller cannot be found, then the input generates an error message.
Collect event logs from a remote Windows machine

You have several choices to collect data from a remote Windows machine:

**Use a universal forwarder**

You can install a universal forwarder on the Windows machine and instruct it to collect event logs. You can do this manually, or use a deployment server to manage the forwarder configuration.

For specific instructions to install the universal forwarder, see Install a Windows universal forwarder from an installer in the Universal Forwarder manual.

1. On the Windows machine that you want to collect Windows Event Logs, download the universal forwarder software from Splunk.
2. Run the universal forwarder installation package to begin the installation process.
3. When the installer prompts you, configure a receiving indexer.
4. When the installer prompts you to specify inputs, enable the event log inputs by checking the "Event logs" checkbox.
5. Complete the installation procedure.
6. On the receiving indexer, use Splunk Web to search for the event log data. An example search string follows:

   ```
   host=<name of remote Windows machine> sourcetype=Wineventlog
   ```

**Use WMI**

If you choose to collect event logs remotely using WMI, you must install Splunk Enterprise to run as an Active Directory domain user. If the selected domain user is not a member of the Administrators or Domain Admins groups, then you must configure event log security to give the domain user access to the event logs.

To change event log security for access to the event logs from remote machines, you must:

- Have administrator access to the machine from which you are collecting event logs.
- Understand how the Security Description Definition Language (SDDL) works, and how to assign permissions with it. See Security Description Definition Language (SDDL) for more information.

You can use the `wevtutil` utility to set event log security.
See Considerations for deciding how to monitor remote Windows data for information on collecting data from remote Windows machines.

1. Download Splunk Enterprise instance onto a Windows machine.
2. Double-click the installer file to begin the installation.
3. When the installer prompts you to specify a user, choose Domain user.
4. On the next installer pane, enter the domain user name and password that Splunk Enterprise should use when it runs.
5. Follow the prompts to complete installation of the software.
6. Once the software has installed, log into the instance.
7. Use Splunk Web to add the remote event log input, as described in Configure remote event log monitoring.

**Anomalous machine names are visible in event logs on some systems**

On some Windows systems, you might see some event logs with randomly-generated machine names. This is the result of those systems logging events before the user has named the system, during the OS installation process.

This anomaly occurs only when you collect logs from the above-mentioned versions of Windows remotely over WMI.

**Use Splunk Web to configure event log monitoring**

To get local Windows event log data, point your Splunk instance at the Event Log service.

*Go to the Add New page*

You can get there by two routes:

- Splunk Home
- Splunk Settings

**By Splunk Settings:**

1. Click Settings in the upper right corner of Splunk Web.
2. Click Data Inputs.
3. Click Local event log collection.
4. Click New to add an input.
By Splunk Home:

1. Click the Add Data link in Splunk Home.
2. Click Monitor to monitor Event Log data on the local Windows machine, or Forward to forward Event Log data from another Windows machine. Splunk Enterprise loads the "Add Data - Select Source" page.
3. If you selected Forward, choose or create the group of forwarders you want this input to apply to. See "Forward data" in this manual.
4. Click Next.

Select the input source

1. In the left pane, select Local Event Logs
2. In the Select Event Logs list box, choose the Event Log channels you want this input to monitor.
3. Click once on each Event Log channel you want to monitor. Splunk Enterprise moves the channel from the "Available items" window to the "Selected items" window.
4. To unselect a channel, click on its name in the "Available Items" window. Splunk Enterprise moves the channel from the "Selected items" window to the "Available items" window.
5. To select or unselect all of the event logs, click on the "add all" or "remove all" links. Important: Selecting all of the channels can result in the indexing of a lot of data, possibly more than your license allows.
6. Click Next.

Specify input settings

The Input Settings page lets you specify application context, default host value, and index. All of these parameters are optional.

Host only sets the host field in the resulting events. It does not direct Splunk Enterprise to look on a specific machine on your network.

1. Select the appropriate Application context for this input.
2. Set the Host name value. You have several choices for this setting. Learn more about setting the host value in About hosts.
3. Set the Index that Splunk Enterprise should send data to. 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 Enterprise has a number of utility indexes, which also appear in this dropdown box.
4. Click Review.
Review your choices

After you specify all your input settings, you can review your selections. Splunk Enterprise lists all options you selected, including the type of monitor, the source, the source type, the application context, and the index.

1. Review the settings.
2. If they do not match what you want, click < to go back to the previous step in the wizard. Otherwise, click Submit.

Splunk Enterprise then loads the "Success" page and begins indexing the specified Event Log channels.

Configure remote event log monitoring

The process for configuring remote event log monitoring is nearly identical to the process for monitoring local event logs.

Selecting all of the Event Log channels can result in the indexing of a lot of data, possibly more than your Splunk license can support.

1. Follow the instructions to get to the Add New page, as described in Go to the Add New page.
2. In the left pane, locate and select Remote Event Logs.
3. In the Event Log collection name field, enter a unique name for this input that you will remember.
4. In the Choose logs from this host field, enter the host name or IP address of the machine that contains the Event Log channels you want to monitor.
5. Click the Find logs button to refresh the page with a list of available Event Log channels on the machine you entered.
6. Click once on each Event Log channel you want to monitor. Splunk Enterprise moves the channel from the "Available items" window to the "Selected items" window.
7. To unselect a channel, click on its name in the "Available Items" window. Splunk Enterprise moves the channel from the "Selected items" window to the "Available items" window.
8. To select or unselect all of the event logs, click on the "add all" or "remove all" links.
9. In the Collect the same set of logs from additional hosts field, enter host names or IP addresses of additional machines that contain the Event Logs you selected previously. Separate multiple machines with commas.
10. Click the green Next button.
11. Follow the instructions to specify input settings, as described in "Specify input settings."
12. Follow the instructions to review your choices, as described in "Review your choices."

Use inputs.conf to configure event log monitoring

Edit inputs.conf to configure event log monitoring.

1. Using Notepad or a similar editor, open %SPLUNK_HOME%\etc\system\local\inputs.conf for editing. You might need to create this file if it does not exist.
2. Enable Windows event log inputs by adding input stanzas that reference Event Log channels.
3. Save the file and close it.
4. Restart Splunk Enterprise.

For more information on configuring data inputs with inputs.conf, see Configure your inputs.

Specify global settings for Windows Event Log inputs

As you define Windows Event Log inputs in inputs.conf, confirm that you explicitly specify global settings in the correct place.

If you specify global settings for Windows Event Log inputs, such as host, sourcetype, and so on, you can place those settings in one of the following areas:

- Under the [WinEventLog] global stanza. This stanza is equal to the [default] stanza for other monitoring inputs. For example:

  [default]
  _meta = hf_proxy::meta_test

  [WinEventLog]
  _meta = hf_proxy::meta_test
  host = WIN2K16_DC
  index = wineventlog

  [WinEventLog://Applications]
  disabled = 0
• Under the Windows Event Log input stanza for the Event Log channel that you want to monitor. For example:

    [default]
    _meta = hf_proxy::meta_test

    [WinEventLog]
    host = WIN2K16_DC
    index = wineventlog

    [WinEventLog://Applications]
    disabled = 0
    _meta = hf_proxy::meta_test

You can always review the defaults for a configuration file by looking at the examples in %SPLUNK_HOME%\etc\system\default or at the spec file in the Admin Manual.

**Event log monitor configuration values**

Windows event log (*.evt) files are in binary format. You cannot monitor them like you do a normal text file. The splunkd service monitors these binary files by using the appropriate APIs to read and index the data within the files.

Splunk Enterprise uses the following stanzas in inputs.conf to monitor the default Windows event logs:

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

**Monitor non-default Windows event logs**

You can also configure Splunk Enterprise to monitor non-default Windows event logs. Before you can do this, you must import them to the Windows Event Viewer. After you import the logs, you can add them to your local copy of inputs.conf, as follows:

```plaintext
[WinEventLog://DNS Server]
disabled = 0
[WinEventLog://Directory Service]
disabled = 0
```
Use the "Full Name" log property in Event Viewer to specify complex Event Log channel names properly

You can use the "Full Name" Event Log property in Event Viewer to ensure that you specify the correct Event Log channel in an inputs.conf stanza.

For example, to monitor the Task Scheduler application log (Microsoft-Windows-TaskScheduler-Operational):

1. Launch Event Viewer.
2. Expand Applications and Services Logs > Microsoft > Windows > TaskScheduler.
3. Right-click Operational and select Properties.
4. In the dialog that appears, copy the text in the "Full Name" field.
5. Append this text into the WinEventLog:// stanza:

    [WinEventLog://Microsoft-Windows-TaskScheduler/Operational]
    disabled = 0

Disable an event log stanza

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

Configuration settings for monitoring Windows Event Logs

Splunk software uses the following settings in inputs.conf to monitor Event Log files:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_from</td>
<td>How events are to be read. Acceptable values are oldest (meaning read logs from the oldest to the newest) and newest (meaning read logs from the newest to the oldest.) You cannot set this attribute to newest while also setting the current_only attribute to 1.</td>
<td>oldest</td>
</tr>
<tr>
<td>current_only</td>
<td>How events are to be indexed. Acceptable values are 1 (where the input acquires events that arrive after the input starts for</td>
<td>0</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>start_from</td>
<td>The first time, like <code>tail -f</code> on *nix systems) or 0 (where the input gets all existing events in the log and then continues to monitor incoming events in real time.) You cannot set this attribute to 1 and also set the <code>start_from</code> attribute to <code>newest</code>.</td>
<td></td>
</tr>
<tr>
<td>checkpointInterval</td>
<td>How frequently, in seconds, the Windows Event Log input saves a checkpoint. Checkpoints store the eventID of acquired events to enable Splunk software to resume monitoring at the correct event after a shutdown or outage.</td>
<td>0</td>
</tr>
<tr>
<td>evt_resolve_ad_ds</td>
<td>The domain controller Splunk software uses to interact with Active Directory while indexing Windows Event Log channels. Valid only when you set the <code>evt_resolve_ad_obj</code> attribute to 1 and omit the <code>evt_dc_name</code> attribute. Valid values are <code>auto</code> (meaning choose the nearest domain controller to bind to for AD object resolution) or <code>PDC</code> (meaning bind to the primary domain controller for the AD site that the host is in.) If you also set the <code>evt_dc_name</code> attribute, Splunk software ignores this attribute.</td>
<td>auto</td>
</tr>
<tr>
<td>evt_resolve_ad_obj</td>
<td>How Splunk software interacts with Active Directory while indexing Windows Event Log channels. Valid values are 1 (meaning resolve Active Directory objects like Globally Unique IDentifier (GUID) and Security IDentifier (SID) objects to their canonical names for a specific Windows event log channel) and 0 (meaning not to attempt any resolution.) When you set this value to 1, you can optionally specify the Domain Controller</td>
<td>0</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>name and/or DNS name of the domain to bind to, which Splunk software uses to resolve the AD objects. If you do not set this value, Splunk software attempts to resolve the AD objects.</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>evt_dc_name</td>
<td>Which Active Directory domain controller to bind to resolve AD objects. This name can be the NetBIOS name of the domain controller, the fully-qualified DNS name of the domain controller, or an environment variable name, specified as $Environment_variable. If you set this attribute, then Splunk software ignores the evt_resolve_ad_ds attribute, which controls how the software determines the best domain controller to bind to for AD object resolution. If you specify an environment variable, you must prepend a dollar sign ($) to the environment variable name. Splunk software uses the specified environment variable as the domain controller to connect to for AD object resolution. For example, to use the %LOGONSERVER% variable, specify evt_dc_name = $logonserver. You can precede either format with two backslash characters. This attribute does not have a default.</td>
<td>N/A</td>
</tr>
<tr>
<td>evt_dns_name</td>
<td>The fully-qualified DNS name of the domain to bind to resolve AD objects.</td>
<td>N/A</td>
</tr>
<tr>
<td>suppress_text</td>
<td>Whether to include the message text that comes with a security event. A value of 1 suppresses the message text, and a value of 0 preserves the text.</td>
<td>0</td>
</tr>
<tr>
<td>use_old_eventlog_api</td>
<td>Whether or not to read Event Log events with the Event Logging API. (Use the API that</td>
<td>false</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>use_threads</td>
<td>Specifies the number of threads, in addition to the default writer thread, that can be created to filter events with the blacklist/whitelist regular expression. This is an advanced setting. Contact Splunk Support before you change it.</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>The maximum number of threads is 15.</td>
<td></td>
</tr>
<tr>
<td>thread_wait_time_msec</td>
<td>The interval, in milliseconds, between attempts to re-read Event Log files when a read error occurs. This is an advanced setting. Contact Splunk Support before you change it.</td>
<td>5000</td>
</tr>
<tr>
<td>suppress_checkpoint</td>
<td>Whether or not the Event Log strictly follows the 'checkpointInterval' setting when it saves a checkpoint. This is an advanced setting. Contact Splunk Support before you change it.</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>By default, the Event Log input saves a checkpoint from between zero and checkpointInterval seconds, depending on incoming event volume.</td>
<td></td>
</tr>
<tr>
<td>suppress_sourcename</td>
<td>Whether or not to exclude the 'sourcename' field from events. This is an advanced setting. Contact Splunk Support before you change it.</td>
<td>false</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>suppress_keywords</td>
<td>When set to true, the input excludes the 'sourcename' field from events and thruput performance (the number of events processed per second) improves.</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>Whether or not to exclude the 'keywords' field from events.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This is an advanced setting. Contact Splunk Support before you change it.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>When set to true, the input excludes the 'keywords' field from events and thruput performance (the number of events processed per second) improves.</td>
<td></td>
</tr>
<tr>
<td>suppress_type</td>
<td>Whether or not to exclude the 'type' field from events.</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>Whether or not to exclude the 'type' field from events.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This is an advanced setting. Contact Splunk Support before you change it.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>When set to true, the input excludes the 'type' field from events and thruput performance (the number of events processed per second) improves.</td>
<td></td>
</tr>
<tr>
<td>suppress_task</td>
<td>Whether or not to exclude the 'task' field from events.</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>Whether or not to exclude the 'task' field from events.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This is an advanced setting. Contact Splunk Support before you change it.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>When set to true, the input excludes the 'task' field from events and thruput performance (the number of events processed per second) improves.</td>
<td></td>
</tr>
<tr>
<td>suppress_opcode</td>
<td>Whether or not to exclude the 'opcode' field from events.</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>Whether or not to exclude the 'opcode' field from events.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This is an advanced setting. Contact Splunk Support before you change it.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>When set to true, the input excludes the 'opcode' field from events and thruput performance (the number of events processed per second) improves.</td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>whitelist</td>
<td>Whether to index events that match the specified text string. This attribute is optional. You can specify one of two formats: • One or more Event Log event codes or event IDs (Event Code/ID format.) • One or more sets of keys and regular expressions (Advanced filtering format.) You cannot mix formats in a single entry. You also cannot mix formats in the same stanza. Splunk software processes whitelists first, then blacklists. If no whitelist is present, all events are indexed. When you use the Event Code/ID format: • For multiple codes/IDs, separate the list with commas. • For ranges, use hyphens (for example &quot;0-1000,5000-1000&quot;). When using the advanced filtering format: • Use '=' between the key and the regular expression that represents your filter (for example &quot;whitelist = EventCode=%^1([8-9])$%&quot; • You can have multiple key/regular expression sets in a single advanced filtering entry. Splunk Enterprise conjuncts the sets logically. This</td>
<td>N/A</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>blacklist</td>
<td>Do not index events that match the text string specified. This attribute is optional.</td>
<td></td>
</tr>
</tbody>
</table>

You can specify one of two formats:

- One or more Event Log event codes or event IDs (Event Log code/ID format.)
- One or more sets of keys and regular expressions. (Advanced filtering format.)

You cannot mix formats in a single entry. You also cannot mix formats in the same stanza.

Whitelists are processed first, then blacklists. If no blacklist is present, all events are indexed.

When using the Event Log code/ID format:

- For multiple codes/IDs, separate the list with commas.
- For ranges, use hyphens (for example "0-1000,5000-1000").

When using the advanced filtering format:

- Use '=' between the key and the regular expression that represents your filter (for example "blacklist = EventCode=%^1([8-9])$%"
You can have multiple key/regular expression sets in a single advanced filtering entry. Splunk software conjuncts the sets logically. This means that the entry is valid only if all of the sets in the entry are true.

- You can specify up to 10 blacklists per stanza by adding a number to the end of the `blacklist` attribute, for example `blacklist1...blacklist9`.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>renderXml</code></td>
<td>Render event data as XML supplied by the Windows Event Log subsystem. This setting is optional. A value of '1' or 'true' means to render the events as XML. A value of '0' or 'false' means to render the events as plain text. If you set <code>renderXml</code> to true, if you want to also create whitelists or blacklists to filter event data, you must use the <code>$XmlRegex</code> special key in your whitelists or blacklists.</td>
<td>0 (false)</td>
</tr>
<tr>
<td><code>index</code></td>
<td>The index that this input should send the data to. The default index</td>
<td></td>
</tr>
<tr>
<td><code>disabled</code></td>
<td>Whether or not the input should run. Valid values are 0 (meaning that the input should run) and 1 (meaning that the input should not run).</td>
<td>0</td>
</tr>
</tbody>
</table>

**Use the Security event log to monitor changes to files**

You can monitor changes to files on your system by enabling security auditing on a set of files and/or directories and then monitoring the Security event log channel for change events. The event log monitoring input includes three attributes which you can use in `inputs.conf`. For example:

```
[WinEventLog://Security]
```
To enable security auditing for a set of files or directories, read "Auditing Security Events How To" (http://technet.microsoft.com/en-us/library/cc727935%28v=ws.10%29.aspx) on MS Technet.

You can also use the suppress_text attribute to include or exclude the message text that comes with a security event.

When you set suppress_text to 1 in a Windows Event Log Security stanza, the entire message text does not get indexed. This includes any contextual information about the security event. If you need this contextual information, do not set suppress_text in the stanza.

To use a specific domain controller, set the evt_dc_name attribute:

```
[WinEventLog://Security]
disabled = 0
start_from = oldest
current_only = 0
evt_resolve_ad_obj = 1
checkpointInterval = 5
# suppress message text, we only want the event number.
suppress_text = 1
# only index events with these event IDs.
whitelist = 0-2000,2001-10000
# exclude these event IDs from being indexed.
blacklist = 2001-3000
```

To use a specific domain controller, set the `evt_dc_name` attribute:

```
[WinEventLog://Security]
disabled = 0
start_from = oldest
current_only = 0
evt_resolve_ad_obj = 1
evt_dc_name = boston-dc1.contoso.com
checkpointInterval = 5
# suppress message text, we only want the event number.
suppress_text = 1
# only index events with these event IDs.
whitelist = 0-2000,2001-10000
```
To use the primary domain controller to resolve AD objects, set the `evt_resolve_ad_ds` attribute to PDC. Otherwise, it locates the nearest domain controller:

```
[WinEventLog://Security]
disabled = 0
start_from = oldest
current_only = 0
evt_resolve_ad_obj = 1
evt_resolve_ad_ds = PDC
checkpointInterval = 5
# suppress message text, we only want the event number.
suppress_text = 1
# only index events with these event IDs.
whitelist = 0-2000,2001-10000
blacklist = 2001-3000
```

Create advanced filters with 'whitelist' and 'blacklist'

You can perform advanced filtering of incoming events with the `whitelist` and `blacklist` settings in addition to filtering based solely on event codes. To do this, specify the key/regular expression format in the setting:

```
whitelist = key=<regular expression> [key=<regular expression>] ...
```

In this format, `key` is a valid entry from the following list:

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TimeGenerated</td>
<td>The time that the computer generated the event. Splunk Enterprise only generates the time string as the event.</td>
</tr>
<tr>
<td>$Timestamp</td>
<td>The time that the event was received and recorded by the Event Log service. Splunk Enterprise only generates the time string as the event.</td>
</tr>
<tr>
<td>$XmlRegex</td>
<td>A special key that tells Splunk Enterprise to filter on events generated in XML. To use this key, set it to the value that you want Splunk Enterprise to filter on. You must configure the input that you want to blacklist or whitelist to render events in XML by using the <code>renderXml = true</code> setting under that input stanza. Splunk Enterprise conjuncts multiple entries in a single whitelist or blacklist line; all of the entries must match for the filter to trigger.</td>
</tr>
<tr>
<td>Category</td>
<td>The category number for a specific event source.</td>
</tr>
<tr>
<td>CategoryString</td>
<td>A string translation of the category. The translation depends on the event source.</td>
</tr>
<tr>
<td>ComputerName</td>
<td>The name of the computer that generated the event.</td>
</tr>
<tr>
<td>Key</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EventCode</td>
<td>The event ID number for an event. Corresponds to &quot;Event ID&quot; in Event Viewer.</td>
</tr>
<tr>
<td>EventType</td>
<td>A numeric value that represents one of the five types of events that can be logged (Error, Warning, Information, Success Audit, and Failure Audit.) Available only on machines that run Windows Server 2003 and earlier or clients running Windows XP and earlier. See &quot;Win32_NTLogEvent class (Windows)&quot; (<a href="http://msdn.microsoft.com/en-us/library/aa394226(v=vs.85).aspx">http://msdn.microsoft.com/en-us/library/aa394226(v=vs.85).aspx</a>) on MSDN.</td>
</tr>
<tr>
<td>Keywords</td>
<td>An element used to classify different types of events within an event log channel. The Security Event Log channel has this element, for example.</td>
</tr>
<tr>
<td>LogName</td>
<td>The name of the Event Log channel that received the event. Corresponds to &quot;Log Name&quot; in Event Viewer.</td>
</tr>
<tr>
<td>Message</td>
<td>The text of the message in the event.</td>
</tr>
<tr>
<td>OpCode</td>
<td>The severity level of the event (&quot;OpCode&quot; in Event Viewer.)</td>
</tr>
<tr>
<td>RecordNumber</td>
<td>The Windows Event Log record number. Each event on a Windows machine gets a record number. This number starts at 0 with the first event generated on the system, and increases with each new event generated, until it reached a maximum of 4294967295. It then rolls back over to 0.</td>
</tr>
<tr>
<td>Sid</td>
<td>The Security Identifier (SID) of the principal (such as a user, group, computer, or other entity) that was associated with or generated the event. See &quot;Win32_UserAccount class&quot; (<a href="http://msdn.microsoft.com/en-us/library/windows/desktop/aa394507%28v=vs.85%29.aspx">http://msdn.microsoft.com/en-us/library/windows/desktop/aa394507%28v=vs.85%29.aspx</a>) on MSDN.</td>
</tr>
<tr>
<td>SidType</td>
<td>A numeric value that represents the type of SID that was associated with the event. See &quot;Win32_UserAccount class&quot; (<a href="http://msdn.microsoft.com/en-us/library/windows/desktop/aa394507%28v=vs.85%29.aspx">http://msdn.microsoft.com/en-us/library/windows/desktop/aa394507%28v=vs.85%29.aspx</a>) on MSDN.</td>
</tr>
<tr>
<td>SourceName</td>
<td>The source of the entity that generated the event (&quot;Source&quot; in Event Viewer)</td>
</tr>
<tr>
<td>TaskCategory</td>
<td>The task category of the event. Event sources let you define categories so that you can filter them with Event Viewer (using the &quot;Task Category&quot; field. See Event Categories (Windows) (<a href="http://msdn.microsoft.com/en-us/library/aa363649%28VS.85%29.aspx">http://msdn.microsoft.com/en-us/library/aa363649%28VS.85%29.aspx</a>) on MSDN.</td>
</tr>
<tr>
<td>Type</td>
<td>A numeric value that represents one of the five types of events that can be logged (Error, Warning, Information, Success Audit, and Failure Audit.) Only available on machines that run Windows Server 2008 or later, or Windows Vista or later. See &quot;Win32_NTLogEvent class (Windows)&quot; (<a href="http://msdn.microsoft.com/en-us/library/aa394226(v=vs.85).aspx">http://msdn.microsoft.com/en-us/library/aa394226(v=vs.85).aspx</a>) on MSDN.</td>
</tr>
<tr>
<td>User</td>
<td>The user associated with the event. Correlates to &quot;User&quot; in Event Viewer.</td>
</tr>
</tbody>
</table>

and <regular expression> is any valid regular expression that represents the filters that you want to include (when used with the whitelist attribute) or
exclude (when used with the blacklist attribute).

You can specify more than one key/regular expression set on a single entry line. When you do this, Splunk Enterprise logically conjuncts the sets. This means that only events that satisfy all of the sets on the line are valid for inclusion or exclusion. For example, this entry:

```
whitelist = EventCode="^1([0-5])" Message="^Error"
```

means to include events that have an EventCode ranging from 10 to 15 and contain a Message that begins with the word Error.

You can specify up to 10 separate whitelist or blacklist entries in each stanza. To do so, add a number at the end of the whitelist or blacklist entry on a separate line:

```
whitelist = key=<regular expression>
whitelist1 = key=<regular expression> key2=<regular expression 2>
whitelist2 = key=<regular expression>
```

You cannot specify an entry that has more than one key/regular expression set that references the same key. If, for example, you specify:

```
whitelist = EventCode="^1([0-5])" EventCode="^2([0-5])"
```

Splunk Enterprise ignores the first set and only attempts to include events that match the second set. In this case, only events that contain an EventCode between 20 and 25 match. Events that contain an EventCode between 10 and 15 do not match. Only the last set in the entry ever matches. To resolve this problem, specify two separate entries in the stanza:

```
whitelist = EventCode="^1([0-5])"
whitelist1 = EventCode="^2([0-5])"
```

**Resolve Active Directory objects in event log files**

To specify whether Active Directory objects like globally unique identifiers (GUIDs) and security identifiers (SIDs) are resolved for a given Windows event log channel, use the `evt_resolve_ad_obj` attribute (1=enabled, 0=disabled) for that channel's stanza in your local copy of `inputs.conf`. The `evt_resolve_ad_obj` attribute is on by default for the Security channel.

For example:
To specify a domain controller for the domain that Splunk should bind to in order to resolve AD objects, use the `evt_dc_name` attribute.

The string specified in the `evt_dc_name` attribute can represent either the domain controller NetBIOS name, or its fully-qualified domain name (FQDN). 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

To specify the FQDN of the domain to bind to, use the `evt_dns_name` attribute.

For example:

```
[WinEventLog://Security]
disabled = 0
start_from = oldest
current_only = 0
evt_resolve_ad_obj = 1
evt_dc_name = ftw-dc-01.splunk.com
evt_dns_name = splunk.com
checkpointInterval = 5
```

**Constraints for using the evt_dc_name and evt_resolve_ad_obj attributes**

When you use the `evt_resolve_ad_obj` and `evt_dc_name` attributes:

- Splunk software first attempts to resolve SIDs and GUIDs using the domain controller (DC) specified in the `evt_dc_name` attribute first. If it cannot resolve SIDs using this DC, it attempts to bind to the default DC to perform the translation.
- If Splunk software cannot contact a DC to translate SIDs, it attempts to use the local machine for translation.
- If none of these methods works, then Splunk prints the SID as it was captured in the event.
Splunk software cannot translate SIDs that are not in the format
\texttt{S-1-N-NN-NNNNNNNNNN-NNNNNNNNNN-NNNNNNNNNN-NNNN}.

If you discover that SIDs are not being translated properly, review
\texttt{%SPLUNK_HOME%\var\log\splunkd.log} for clues on what the problem might be.

**Specify whether to start index at the earliest or the most recent event**

Use the \texttt{start_from} attribute to specify whether events are indexed starting at
the earliest event or the most recent. By default, indexing starts with the oldest
data and moves forward. Do not change this setting, because Splunk software
stops indexing after it has indexed the backlog using this method.

Use the \texttt{current_only} attribute to specify whether to index all preexisting events
in a given log channel. When set to 1, only events that appear from the moment
the Splunk deployment was started are indexed. When set to 0, all events are
indexed.

For example:

```
[WinEventLog://Application]
disabled = 0
start_from = oldest
current_only = 1
```

**Display Windows Event Log events in XML**

To have Splunk Enterprise generate Windows Event Log events in XML, use the
\texttt{renderXml} setting in a Windows Event Log input stanza:

```
[WinEventLog://System]
disabled = 0
renderXml = 1
evt_resolve_ad_obj = 1
evt_dns_name = "$SV5DC02"
```

This input stanza generates events like the following:

```xml
<Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'>
  <System>
    <Provider Name='Service Control Manager' Guid='(555908d1-a6d7-4695-8e1e-26931d2012f4)' EventSourceName='Service Control Manager'/>
    <EventID Qualifiers='16384'>7036</EventID>
    <Version>0</Version>
  </System>
</Event>
```
When you instruct Splunk Enterprise to render events in XML, event keys within the XML event render in English regardless of the machine system locale.

Compare the following events generated on a French version of Windows Server:

**Standard event:**

04/29/2014 02:50:23 PM
LogName=Security
SourceName=Microsoft Windows security auditing.
EventCode=4672
EventType=0
Type=Information
ComputerName=sacreblue
TaskCategory=Ouverture de session spéciale
OpCode=Informations
RecordNumber=2746
Keywords=Succès de l'audit
Message=Privilèges spéciaux attribués à la nouvelle ouverture de session.

Sujet :

<table>
<thead>
<tr>
<th>ID de sécurité :</th>
<th>AUTORITE NT\Système</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nom du compte :</td>
<td>Système</td>
</tr>
<tr>
<td>Domaine du compte :</td>
<td>AUTORITE NT</td>
</tr>
<tr>
<td>ID d'ouverture de session :</td>
<td>0x3e7</td>
</tr>
</tbody>
</table>

Privilèges :
SeAssignPrimaryTokenPrivilege
SeTcbPrivilege
SeSecurityPrivilege
XML event:

<Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'>
  <System>
    <Provider Name='Microsoft-Windows-Security-Auditing' Guid='{54849625-5478-4994-A5BA-3E3B0328C30D}' />
    <EventID>4672</EventID>
    <Version>0</Version>
    <Level>0</Level>
    <Task>12548</Task>
    <Opcode>0</Opcode>
    <Keywords>0x8020000000000000</Keywords>
    <TimeCreated SystemTime='2014-04-29T22:15:03.280843700Z' />
    <EventRecordID>2756</EventRecordID>
    <Correlation />
    <Execution ProcessID='540' ThreadID='372' />
    <Channel>Security</Channel>
    <Computer>sacreblue</Computer>
    <Security />
  </System>
  <EventData>
    <Data Name='SubjectUserSid'>AUTORITE NT\Système</Data>
    <Data Name='SubjectUserName'>Système</Data>
    <Data Name='SubjectDomainName'>AUTORITE NT</Data>
    <Data Name='SubjectLogonId'>0x3e7</Data>
    <Data Name='PrivilegeList'>SeAssignPrimaryTokenPrivilege
      SeTcbPrivilege
      SeSecurityPrivilege
      SeTakeOwnershipPrivilege
      SeLoadDriverPrivilege
      SeBackupPrivilege
      SeRestorePrivilege
      SeDebugPrivilege
      SeAuditPrivilege
      SeSystemEnvironmentPrivilege
      SeImpersonatePrivilege</Data>
  </EventData>
</Event>
Use blacklists and whitelists to filter on XML-based events

If you render events in XML, and you want to use whitelists and blacklists to filter on those events, you must use the special key $XmlRegex when you build your whitelists or blacklists. The whitelist or blacklist triggers when the value that you specify with $XmlRegex is found anywhere in the rendered XML event. $XmlRegex does not function if you do not explicitly specify the input to render in XML with the renderXml = true setting.

Following is an example of using whitelists on XML events. Splunk Enterprise indexes all XML events that contain the word "Error":

[WinEventLog://System]
disabled = 0
renderXml = 1
evt_resolve_ad_obj = 1
evt_dns_name = "SV5DC02"
whitelist = $XmlRegex='Error'
See "Create advanced filters with 'whitelist' and 'blacklist' for additional information and syntax.

Use the CLI to configure event log monitoring

You can use the CLI to configure local event log monitoring. Before you use the CLI, create stanza entries in inputs.conf first. See "Use inputs.conf to configure event log monitoring" in this topic.

The CLI is not available for remote Event Log collections.

To list all configured Event Log channels on the local machine:

> splunk list eventlog
You can also list a specific channel by specifying its name:

> splunk list eventlog <ChannelName>
To enable an Event Log channel:
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 that contains the exported files.

Do not attempt to monitor an .evt or .evtx file that is open for writing. Windows does not allow read access to these files. Use the event log monitoring feature instead.

Constraints for monitoring Windows Event log files directly

- As a result of API and log channel processing constraints on Windows XP and Server 2003 systems, imported .evt files from those systems do not contain the "Message" field. This means that the contents of the "Message" field do not appear in your Splunk index.
- Splunk Enterprise on Windows Vista and later and Server 2008/2008 R2 and later can index both .evt and .evtx files.
- If your .evt or .evtx file is not from a standard event log channel, you must make sure that any dynamic link library (DLL) files required by that channel are present on the computer on which you are indexing.
- Splunk Enterprise indexes an .evt or .evtx file in the primary locale/language of the computer that collects the file.
- Files that have been exported from another machine do not work with the Splunk Web Upload feature. This is because those files contain information that is specific to the machine that generated them. Other machines won't be able to process the files in their unaltered form.

When producing .evt or .evtx files on one system, and monitoring them on another, it's possible that not all of the fields in each event expand as they would on the system producing the events. This is caused by variations in DLL versions, availability and APIs. Differences in OS version, language, Service Pack level and installed third party DLLs, etc. can also have this effect.
Monitor file system changes

Splunk Enterprise supports the monitoring of Windows file system changes through the Security Event Log channel. File change monitoring requires the enabling of security auditing for the files and folders you want to monitor for changes and using the Event Log monitor to monitor the Security event log channel. This procedure of monitoring file system changes replaces the deprecated file system change monitor input.

If you have Splunk Cloud and want to monitor Windows file system changes through the Security Event Log channel, use the Splunk Universal Forwarder.

What do you need to monitor file system changes?

<table>
<thead>
<tr>
<th>Activity: Monitor file system changes</th>
<th>Required permissions:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Splunk Enterprise must run on Windows AND</td>
</tr>
<tr>
<td></td>
<td>• Splunk Enterprise must run as the Local System user OR as a domain user with specific security policy rights to read the Security event log AND</td>
</tr>
<tr>
<td></td>
<td>• You must enable security auditing for the file(s) or director(ies) you want Splunk Enterprise to monitor changes to</td>
</tr>
</tbody>
</table>

Use the Security event log to monitor changes to files

You can monitor changes to files on your system by enabling security auditing on a set of files and/or directories and then monitoring the Security event log channel for change events. The event log monitoring input includes three attributes which you can use in `inputs.conf`.

You can use these attributes outside of the context of the Security event log and file system changes. Also, this list of attributes is only a subset of the available attributes for `inputs.conf`. For additional attributes, read Monitor Windows event log data in this manual.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>whitelist</td>
<td>Index events that match the text string specified. This attribute is optional. You can specify one of two formats:</td>
<td>N/A</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>blacklist</td>
<td>Do not index events that match the text string specified. This attribute is optional.</td>
<td>N/A</td>
</tr>
</tbody>
</table>

You can specify one of two formats:

- One or more Event Log event codes or event IDs (Event Log code/ID format.)
- One or more sets of keys and regular expressions (Advanced filtering format.)
You cannot mix formats in a single entry. You also cannot mix formats in the same stanza.

Splunk Enterprise processes whitelists first, then blacklists. If no whitelist is present, Splunk Enterprise indexes all events.

When using the Event Code/ID format:

- For multiple codes/IDs, separate the list with commas.
- For ranges, use hyphens (for example "0-1000,5000-1000").

When using the advanced filtering format:

- Use '=' between the key and the regular expression that represents your filter (for example "whitelist = EventCode=%^1([8-9])$%").
- You can have multiple key/regular expression sets in a single advanced filtering entry. Splunk Enterprise joins the sets logically. This means that the entry is valid only if all of the sets in the entry are true.
- You can specify up to 10 blacklists per stanza by adding a number to the end of the `blacklist` attribute, for example `blacklist1...blacklist9`.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>suppress_text</td>
<td>Whether or not to include the message text that comes with a security event. A value of 1 suppresses the message text. A value of 0 preserves the text.</td>
<td>0</td>
</tr>
</tbody>
</table>

**Create advanced filters with `whitelist` and `blacklist`**

You can perform advanced filtering of incoming events with the `whitelist` and `blacklist` attributes in addition to filtering based solely on event codes. To do this, specify the key/regular expression format in the attribute:

`whitelist = key=<regular expression> [key=<regular expression>] ...`

In this format, `key` is a valid entry from the following list:
<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TimeGenerated</td>
<td>The time that the computer generated the event. Only generates the time string as the event.</td>
</tr>
<tr>
<td>$Timestamp</td>
<td>The time that the event was received and recorded by the Event Log service. Splunk Enterprise only generates the time string as the event.</td>
</tr>
<tr>
<td>Category</td>
<td>The category number for a specific event source.</td>
</tr>
<tr>
<td>CategoryString</td>
<td>A string translation of the category. The translation depends on the event source.</td>
</tr>
<tr>
<td>ComputerName</td>
<td>The name of the computer that generated the event.</td>
</tr>
<tr>
<td>EventCode</td>
<td>The event ID number for an event. Corresponds to &quot;Event ID&quot; in Event Viewer.</td>
</tr>
<tr>
<td>EventType</td>
<td>A numeric value that represents one of the five types of events that can be logged (&quot;Error&quot;, &quot;Warning&quot;, &quot;Information&quot;, &quot;Success Audit&quot;, and &quot;Failure Audit&quot;). Available only on machines running Windows Server 2003 and earlier or clients running Windows XP and earlier. See Win32_NTLogEvent class (Windows) (<a href="http://msdn.microsoft.com/en-us/library/aa394226(v=vs.85).aspx">http://msdn.microsoft.com/en-us/library/aa394226(v=vs.85).aspx</a>) on MSDN.</td>
</tr>
<tr>
<td>Keywords</td>
<td>An element used to classify different types of events within an event log channel. The Security Event Log channel has this element, for example.</td>
</tr>
<tr>
<td>LogName</td>
<td>The name of the Event Log channel that received the event. Corresponds to &quot;Log Name&quot; in Event Viewer.</td>
</tr>
<tr>
<td>Message</td>
<td>The text of the message in the event.</td>
</tr>
<tr>
<td>OpCode</td>
<td>The severity level of the event (&quot;OpCode&quot; in Event Viewer.)</td>
</tr>
<tr>
<td>RecordNumber</td>
<td>The Windows Event Log record number. Each event on a Windows server gets a record number. This number starts at 0 with the first event generated on the system, and increases with each new event generated, until it reached a maximum of 4294967295. It then rolls back over to 0.</td>
</tr>
<tr>
<td>Sid</td>
<td>The Security Identifier (SID) of the principal (such as a user, group, computer, or other entity) that was associated with or generated the event. See Win32_UserAccount class (<a href="http://msdn.microsoft.com/en-us/library/windows/desktop/aa394507%28v=vs.85%29.aspx">http://msdn.microsoft.com/en-us/library/windows/desktop/aa394507%28v=vs.85%29.aspx</a>) on MSDN.</td>
</tr>
<tr>
<td>SidType</td>
<td>A numeric value that represents the type of SID that was associated with the event. See Win32_UserAccount class (<a href="http://msdn.microsoft.com/en-us/library/windows/desktop/aa394507%28v=vs.85%29.aspx">http://msdn.microsoft.com/en-us/library/windows/desktop/aa394507%28v=vs.85%29.aspx</a>) on MSDN.</td>
</tr>
<tr>
<td>SourceName</td>
<td>The source of the entity that generated the event (&quot;Source&quot; in Event Viewer)</td>
</tr>
<tr>
<td>TaskCategory</td>
<td>The task category of the event. Event sources allow you to define categories so you can filter them with Event Viewer (using the &quot;Task Category&quot; field. See Event Categories (Windows) (<a href="http://msdn.microsoft.com/en-us/library/aa363649%28VS.85%29.aspx">http://msdn.microsoft.com/en-us/library/aa363649%28VS.85%29.aspx</a>).</td>
</tr>
</tbody>
</table>
### Key Description

<table>
<thead>
<tr>
<th>Key</th>
<th>MSDN.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>A numeric value that represents one of the five types of events that can be logged (&quot;Error&quot;, &quot;Warning&quot;, &quot;Information&quot;, &quot;Success Audit&quot;, and &quot;Failure Audit&quot;). Only available on server machines that run Windows Server 2008 or later, or clients that run Windows Vista or later. See Win32_NTLogEvent class (Windows) (<a href="http://msdn.microsoft.com/en-us/library/aa394226(v=vs.85).aspx">http://msdn.microsoft.com/en-us/library/aa394226(v=vs.85).aspx</a>) on MSDN.</td>
</tr>
<tr>
<td>User</td>
<td>The user associated with the event. Correlates to &quot;User&quot; in Event Viewer.</td>
</tr>
</tbody>
</table>

<regular expression> is any valid regular expression that represents the filters that you want to include (when used with the whitelist attribute) or exclude (when used with the blacklist attribute).

To learn more about regular expressions and how to use them, visit the Regularexpressions.info (http://www.regular-expressions.info) website.

You can specify more than one regular expression on a single entry line. Only events that satisfy all of the entries on the line are included or excluded. For example, this entry:

```plaintext
whitelist = EventCode="^1([0-5])" Message="^Error"
```

means to include events that have an EventCode ranging from 10 to 15 and contain a Message that begins with the word Error.

You can specify up to 10 separate whitelist or blacklist entries in each stanza. To do so, add a number at the end of the whitelist or blacklist entry on a separate line:

```plaintext
whitelist = key=<regular expression>
whitelist1 = key=<regular expression> key2=<regular expression 2>
whitelist2 = key=<regular expression>
```

You cannot specify an entry that has more than one expression that references the same key. If, for example, you specify:

```plaintext
whitelist = EventCode="^1([0-5])" EventCode="^2([0-5])"
```

Splunk software ignores the first expression and only attempts to include events that match the second expression. In this case, only events that contain an EventCode between 20 and 25 match. Events that contain an EventCode between 10 and 15 do not match. Only the last expression in the entry ever matches.

To resolve this problem, specify two separate entries in the stanza:
whitelist = EventCode="^1\([0-5]\)\$"
whitelist1 = EventCode="^2\([0-5]\)\$"

**Monitor file system changes**

1. Confirm that you have administrator privileges.
2. Enable security auditing. Search for "Enable security auditing" for the version of Windows that you run.
3. Configure the Splunk Enterprise event log monitor input to monitor the Security event log channel.

For instructions on how to configure the Event Log monitor input, see [https://docs.splunk.com/Documentation/Splunk/8.0.1/Data/MonitorWindowseventlogData](https://docs.splunk.com/Documentation/Splunk/8.0.1/Data/MonitorWindowseventlogData)

**Examples of file system change monitoring**

Following are `inputs.conf` stanzas that show examples of how to monitor file system changes.

This stanza collects security events with event ID codes 0 to 2000 and 3001-10000.

```plaintext
[WinEventLog:Security]
disabled = 0
start_from = oldest
current_only = 0
evt_resolve_ad_obj = 1
checkpointInterval = 5
# only index events with these event IDs.
whitelist = 0-2000,2001-10000
# exclude these event IDs from being indexed.
blacklist = 2001-3000
```

This stanza collects security events with event ID codes 0 to 2000 and 3001-10000. It also suppresses the message text that comes in the event ID.

```plaintext
[WinEventLog:Security]
disabled = 0
start_from = oldest
current_only = 0
evt_resolve_ad_obj = 1
checkpointInterval = 5
# suppress message text, we only want the event number.
suppress_text = 1
# only index events with these event IDs.
```
Monitor data through Windows Management Instrumentation (WMI)

Splunk Enterprise supports the use of Windows Management Instrumentation (WMI) providers for agentless access to Windows performance and event log data on remote machines. You can pull event logs from all the Windows machines in your environment without installing anything on those machines.

If possible, use a universal forwarder rather than WMI to collect data from remote machines. The resource load of WMI can exceed that of a Splunk universal forwarder in many cases. Use a forwarder if you collect multiple event logs or performance counters from each host, or from very busy hosts like domain controllers. See Considerations for deciding how to monitor remote Windows data in this manual. If you have Splunk Cloud, you must use the universal forwarder to collect data from WMI providers and forward it to your Splunk Cloud deployment.

WMI-based data inputs can connect to multiple WMI providers. The input runs as a separate process called splunk-wmi.exe. It is a scripted input.

What do you need to monitor WMI-based data?

Here are the basic minimum requirements to monitor WMI-based data. You might need additional permissions based on the logs or performance counters you want to monitor.

For additional details on what's required to monitor WMI-based data, see "Security and remote access considerations" later in this topic.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Required permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor remote event logs over WMI</td>
<td>* Splunk Enterprise must run on Windows</td>
</tr>
<tr>
<td></td>
<td>* Splunk Enterprise must run as a domain user with at least read access to WMI</td>
</tr>
<tr>
<td></td>
<td>* Splunk Enterprise must run as a domain user with appropriate access to the desired event logs</td>
</tr>
<tr>
<td>Monitor remote performance monitor counters over WMI</td>
<td>* Splunk Enterprise must run on Windows</td>
</tr>
<tr>
<td></td>
<td>* Splunk Enterprise must run as a domain user with at least read access to WMI</td>
</tr>
<tr>
<td>Activity</td>
<td>Required permissions</td>
</tr>
<tr>
<td>----------</td>
<td>----------------------</td>
</tr>
<tr>
<td>* Splunk Enterprise must run as a domain user with appropriate access to the Performance Data Helper libraries</td>
<td></td>
</tr>
</tbody>
</table>

**Security and remote access considerations**

Splunk Enterprise and your Windows network must be correctly configured for WMI data access. Review the following prerequisites before attempting to use Splunk Enterprise to get WMI data.

Before Splunk Enterprise can get WMI-based data:

- It must be installed with a user that has permissions to perform remote network connections.
- The user Splunk Enterprise runs as must be a member of an Active Directory (AD) domain or forest and must have appropriate privileges to query WMI providers.
- The Splunk user must also be a member of the local Administrators group on the computer that runs Splunk Enterprise.
- The computer that runs Splunk Enterprise must be able to connect to the remote machine and must have permissions to get the desired data from the remote machine once it has connected.
- Both the Splunk Enterprise instance and the target machines must be part of the same AD domain or forest.

The user that Splunk Enterprise runs as does not need to be a member of the Domain Admins group (and for security reasons, should not be). However, you must have domain administrator privileges to configure access for the user. If you don't have domain administrator access, find someone who can either configure Splunk user access or give domain administrator rights to you.

If you install Splunk Enterprise as the Local System user, remote authentication over WMI does not work. The Local System user has no access to other machines on the network. It is not possible to grant privileges to a Local System account for access to another host.

You can give the Splunk user access to WMI providers by doing one of the following:

- Adding it to the local Administrators group on each member host you want to poll (not recommended for security reasons).
• Adding it to the Domain Admins global group (not recommended for security reasons).
• Assigning least-permissive rights as detailed below (recommended).

**Group memberships and resource access control lists (ACLs)**

To maintain security integrity, place Splunk users into a domain global group and assign permissions on Windows machines and resource ACLs to that group, instead of assigning permissions directly to the user. Assignment of permissions directly to users is a security risk, and can cause problems during security audits or future changes.

**Configure WMI for least permissive access**

If the user you configured Splunk Enterprise to run as is not a domain administrator, you must configure WMI to provide access to this user. Grant only least-permissive access to all Windows resources, including WMI. In order to grant this type of access, follow this checklist. For additional information and step-by-step instructions, see Prepare your Windows network for a Splunk Enterprise installation in the *Installation* manual.

You must grant several levels of access to the user Splunk Enterprise runs as for Splunk Enterprise to collect data over WMI using the least-permissive method:

To deploy these user rights assignments domain-wide, use the **Domain Security Policy** (*dompol.msc*) Microsoft Management Console (MMC) snap-in. After deployment, member hosts inherit those rights assignments on the network during the next AD replication cycle. Restart Splunk Enterprise instances on those machines for the changes to take effect.

To extend this access to domain controllers specifically, assign the rights using the **Domain Controller Security Policy** (*dcpol.msc*) snap-in.

- **Local Security Policy Permissions.** The Splunk user needs the following Local Security Policy user rights assignments defined on each machine you poll for WMI-based data:
  - Access this Computer from the Network
  - Act as part of the operating system
  - Log on as a batch job
  - Log on as a service
  - Profile System Performance
  - Replace a process level token
• **Distributed Component Object Model (DCOM) configuration and permissions.** DCOM must be enabled on every machine you want to monitor. In addition, the Splunk Enterprise user must be assigned permissions to access DCOM. There are many methods available to do this, but the best is to nest the "Distributed COM Users" domain global group into the "Distributed COM Users" local group on each machine you want to monitor, then add the Splunk Enterprise user to the "Distributed COM Users" domain global group. See "Securing a Remote WMI Connection" ([http://msdn.microsoft.com/en-us/library/aa393266(VS.85).aspx](http://msdn.microsoft.com/en-us/library/aa393266(VS.85).aspx)) on MSDN for advanced options to give the Splunk Enterprise user access to DCOM.

• **Performance Monitor configuration and permissions.** The Splunk Enterprise user must be a member of the "Performance Log Users" local group in order for Splunk Enterprise to access remote performance objects over WMI. The best way to do this is to nest the "Performance Log Users" domain global group into the "Performance Log Users" local group on each member host and then assign the user to the global group.

• **WMI namespace security.** The WMI namespace that Splunk Enterprise accesses (most commonly `Root\CIMV2`) must have proper permissions. These permissions must be set manually on each host in your enterprise, as there is no global WMI security. Use the WMI Security MMC snap-in (`wmimgmt.msc`) to enable the following permissions on the WMI tree for each host at the Root namespace for the Splunk user:
  • Execute Methods
  • Enable Account
  • Remote Enable
  • Read Security


**Note:** There is no standard facility for deploying WMI security settings remotely to multiple machines at once using Group Policy. However, Set WMI namespace security via GPO ([http://blogs.msdn.com/spatdsg/archive/2007/11/21/set-wmi-namespace-security-via-gpo-script.aspx](http://blogs.msdn.com/spatdsg/archive/2007/11/21/set-wmi-namespace-security-via-gpo-script.aspx)) on MSDN Blogs offers instructions on how to create a startup script that you can place inside a Group Policy Object (GPO), which sets the namespace security once the GPO applies to the desired hosts. You can then deploy this GPO.
domain-wide or to one or more Organizational Units (OUs).

- **Firewall configuration.** If you have a firewall enabled, you must configure it to allow access for WMI. If you use the Windows Firewall included with recent versions of Windows, the exceptions list explicitly includes WMI. You must set this exception for both the originating and the target machines. See Connecting to WMI Remotely Starting with Vista (http://msdn.microsoft.com/en-us/library/aa822854(VS.85).aspx) on MSDN for more details.


**Test access to WMI providers**

After you configure WMI and set up the Splunk user for access to your domain, test access to the remote machine.

This procedure includes steps to temporarily change the Splunk Enterprise data store directory (the location \SPLUNK\_DB points to). You must do this before testing access to WMI. Failure to do so can result in missing WMI events. This is because the splunk-wmi.exe process updates the WMI checkpoint file every time it runs.

If you attempt to log into a domain controller, you might have to change your domain controller security policy to assign the "Allow log on locally" policy for the designated user.

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

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

3. Go to the **bin** subdirectory under your Splunk Enterprise installation (for example, `cd c:\Program Files\Splunk\bin`).

4. Determine where Splunk Enterprise currently stores its data by running:

   > splunk show datastore-dir

**Note:** Remember where Splunk Enterprise stores its data. You will recall it later.
5. Run the following command to change where Splunk Enterprise stores its data temporarily:

```
> splunk set datastore-dir %TEMP%
```

**Note:** This example sets the data store directory to the current directory specified in the TEMP environment variable. If you want to set it to a different directory, you can do so, but the directory must already exist.

6. Restart Splunk Enterprise:

```
> splunk restart
```

**Note:** It might take a while for Splunk Enterprise to restart.

7. Once Splunk Enterprise has restarted, test access to WMI providers, replacing `<host>` with the name of the remote host:

```
> splunk cmd splunk-wmi -wql "select * from win32_service" -namespace \\<host>\root\cimv2
```

- If you see data streaming back and no error messages, then Splunk Enterprise was able to connect to the WMI provider and query successfully.
- If there is an error, a message with a reason on what caused the error appears. Look for the `error="<msg>"` string in the output for clues on how to correct the problem.

After testing WMI access, point Splunk Enterprise back to the correct database directory by running the following command, and then restarting Splunk Enterprise:

```
> splunk set datastore-dir <directory shown from Step 4>
```

**Configure WMI-based inputs**

All remote data collection in Splunk Enterprise on Windows happens through either WMI providers or a forwarder. See Considerations for deciding how to monitor remote Windows data in this manual.

You can configure WMI-based inputs either in Splunk Web or by editing configuration files. More options are available when using configuration files.

**Configure WMI-based inputs with Splunk Web**

To add WMI-based inputs, use the "Remote event log monitoring" and "Remote Performance monitoring" data inputs. See Configure remote Windows
performance monitoring with Splunk Web. See also Configure remote Windows event log monitoring

**Configure WMI-based inputs with configuration files**

wmi.conf handles remote data collection configurations. Review this file to see the default values for WMI-based inputs. If you want to make changes to the default values, edit a copy of wmi.conf in %SPLUNK_HOME%/etc/system/local. Set values only for the attributes you want to change for a given type of data input. See About configuration files in the Admin manual.

wmi.conf contains several stanzas:

- The [settings] stanza, which specifies global WMI parameters.
- One or more input-specific stanzas, which define how to connect to WMI providers to get data from the remote machine.

**Global settings**

The [settings] stanza specifies global WMI parameters. The entire stanza and every parameter within it are optional. If the stanza is not present, Splunk Enterprise assumes system defaults.

When Splunk Enterprise cannot connect to a defined WMI provider, it generates an error in splunkd.log:

```
05-12-2011 02:39:40.632 -0700 ERROR ExecProcessor - message from "C:\Program Files\Splunk\bin\splunk-wmi.exe" WMI - Unable to connect to WMI namespace "\w2k3m1\root\cimv2" (attempt to connect took 42.06 seconds) (error="The RPC server is unavailable." HRESULT=800706BA)
```

The following attributes control how Splunk Enterprise reconnects to a given WMI provider when an error occurs.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial_backoff</td>
<td>How long, in seconds, to wait the first time after an error occurs before trying to reconnect to the WMI provider. If connection errors continue to occur, Splunk Enterprise doubles the wait time until it reaches the value specified in</td>
<td>5</td>
</tr>
</tbody>
</table>
## Attribute Description Default value

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_backoff</td>
<td>How long, in seconds, to wait between connection attempts, before invoking max_retries_at_max_backoff.</td>
<td>20</td>
</tr>
<tr>
<td>max_retries_at_max_backoff</td>
<td>If the wait time between connection attempts reaches max_backoff, how many times to try to reconnect to the provider, every max_backoff seconds. If Splunk Enterprise continues to encounter errors, it gives up, and won’t attempt to connect to the problem provider again until you restart. It will continue to log errors such as the example shown above.</td>
<td>2</td>
</tr>
<tr>
<td>checkpoint_sync_interval</td>
<td>How long, in seconds, to wait for state data (event log checkpoint) to be written to disk.</td>
<td>2</td>
</tr>
</tbody>
</table>

### Input-specific settings

Input-specific stanzas tell Splunk Enterprise how to connect to WMI providers. They are defined by one of two attributes that specify the type of data Splunk Enterprise should gather. The stanza name can be anything, but usually begins with `WMI:`, for example:

```
[WMI:AppAndSys]
```

When you configure WMI-based inputs in Splunk Web, Splunk Enterprise uses this naming convention for input-specific stanza headers.

You can specify one of two types of data inputs in an input-specific stanza:

- **Event log.** The `event_log_file` attribute tells Splunk Enterprise to expect event log data from the sources defined in the stanza.
- **Windows Query Language (WQL).** The `wql` attribute tells Splunk Enterprise to expect data from a WMI provider. You must also specify a valid WQL statement. You must use this attribute when you collect performance data.

Do not define both of these attributes in one stanza. Use only one or the other. Otherwise, the input defined by the stanza will not run.
The common attributes for both types of inputs are:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>server</td>
<td>A comma-separated list of hosts from which to get data. If this attribute is missing, Splunk Enterprise assumes that you want to connect to the local machine.</td>
<td>The local host</td>
</tr>
<tr>
<td>interval</td>
<td>Tells Splunk Enterprise how often, in seconds, to poll for new data. If this attribute is not present and defined, the input that the stanza defines will not run.</td>
<td>N / A</td>
</tr>
<tr>
<td>disabled</td>
<td>Tells Splunk Enterprise whether this input is enabled or disabled. Set this attribute to 1 to disable the input, and 0 to enable it.</td>
<td>0 (enabled)</td>
</tr>
</tbody>
</table>

The event log-specific parameters are:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_log_file</td>
<td>A comma-separated list of event log channels to monitor.</td>
<td>N / A</td>
</tr>
<tr>
<td>current_only</td>
<td>Whether or not to collect events that occur only when it is running. If events are generated when Splunk Enterprise is stopped, it will not attempt to index those events when it is started again. Set to 1 to collect events that occur only when it is running, and 0 to collect all events.</td>
<td>0 (gather all events)</td>
</tr>
<tr>
<td>disable_hostname_normalization</td>
<td>Do not normalize the host name that is retrieved from a WMI event. By default, Splunk Enterprise normalizes host names by producing a single name for the host by identifying various equivalent host names for the local system. Set this parameter to 1 to disable host name normalization in events, and 0</td>
<td>0 (normalize host names for WMI events)</td>
</tr>
</tbody>
</table>
The WQL-specific parameters are:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>wql</td>
<td>A valid WQL statement.</td>
<td>N/A</td>
</tr>
<tr>
<td>namespace</td>
<td>(Optional) Specifies the path to the WMI provider. The local machine must be able to connect to the remote machine using delegated authentication. If you do not specify a path to a remote machine, Splunk Enterprise connects to the default local namespace (\Root\CIMV2). This default namespace 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 (<a href="http://msdn.microsoft.com/en-us/library/aa394084(VS.85).aspx">http://msdn.microsoft.com/en-us/library/aa394084(VS.85).aspx</a>).</td>
<td>&lt;local server&gt;\Root\CIMV2</td>
</tr>
<tr>
<td>current_only</td>
<td>Whether or not an event notification query is expected. See &quot;WQL query types: event notification versus standard&quot; in this topic for additional information. Set this attribute to 1 to tell Splunk Enterprise to expect an event notification query, and 0 to expect a standard query.</td>
<td>0 (expect a standard query)</td>
</tr>
</tbody>
</table>

**WQL query types: event notification versus standard**

The `current_only` attribute in WQL stanzas determines the type of query the stanza expects to use to collect WMI-based data. When you set the attribute to 1, the stanza expects event notification data. Event notification data is data that alerts you of an incoming event. To get event notification data, you must use an event notification query.

For example, to find out when a remote host spawns processes, you must use an event notification query. Standard queries have no facilities for notifying you when an event has occurred, and can only return results on information that already exists.

Conversely, if you want to know what already-running processes on your system begin with the word "splunk", you must use a standard query. Event notification queries cannot tell you about static, preexisting information.

Event notification queries require that the WQL statement defined for the stanza be structurally and syntactically correct. Improperly formatted WQL will cause the
input defined by the stanza to not run. Review the wmi.conf configuration file reference for specific details and examples.

**WQL query stanzas do not update the WMI checkpoint file**

When you use a WQL query stanza to gather data through WMI, Splunk Enterprise does not update the WMI checkpoint file - the file that determines if WMI data has been indexed. This is by design - a WQL query of any type returns dynamic data and a context for saving a checkpoint for the data produced cannot be built. This means that Splunk Enterprise indexes WMI data that it collects through WQL query stanzas as fresh data each time the stanza runs. This can result in the indexing of duplicate events and possibly impact license volume.

If you need to index data regularly, such as event logs, use the appropriate monitor on a universal forwarder. If you must use WMI, use a standard WMI query type.

**Examples of wmi.conf**

The following is an example of a wmi.conf file:

```plaintext
[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_ReadFormattedData_PerfProc_Process where Name = "splunk-wmi"
disabled = 0

# Listen from three event log channels, capturing log events that occur only
# while Splunk Enterprise runs. Gather data from three machines.
[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' and
TargetInstance.Description='USB Mass Storage Device'
disabled = 0
current_only = 1

Fields for WMI data

When Splunk Enterprise indexes data from WMI-based inputs, it sets the
originating host from the data received. It sets the source for received events to
wmi. It sets the source type of the incoming events based on the following
conditions:

- For event log data, Splunk Enterprise sets the source type to
  WinEventLog:<name of log file>. For example,
  WinEventLog:Application.
- For WQL data, Splunk Enterprise sets the source type to the name of the
  stanza that defines the input. For example, for a stanza named
  [WMI:LocalSplunkdProcess], Splunk sets the source type to
  WMI:LocalSplunkdProcess.

WMI and event transformations

WMI events are not available for transformation at index time. You cannot modify
or extract WMI events as Splunk Enterprise indexes them. This is because WMI
events arrive as a single source (a scripted input), which means they can be
matched only as a single source.

You can modify and extract WMI events at search time. You can also address
WMI-based inputs at parse time by specifying the sourcetype [wmi].

149
For information on how to transform events as they arrive in Splunk Enterprise, see About indexed field extraction in this manual.

**Troubleshooting WMI inputs**

If you encounter problems receiving events through WMI providers or are not getting the results you expect, see Common Issues with Splunk and WMI in the Troubleshooting Manual.

**Monitor Windows Registry data**

The Windows Registry is the central configuration database on a Windows machine. Nearly all Windows processes and third-party programs interact with it. Without a healthy Registry, Windows does not run. Splunk Enterprise supports the capture of Windows Registry settings and lets you monitor changes to the Registry in real time.

When a program makes a change to a configuration, it writes those changes to the Registry. Later, when the program runs again, it looks into the Registry to read those configurations. You can learn when Windows programs and processes add, update, and delete Registry entries on your system. When a Registry entry changes, Splunk Enterprise captures the name of the process that made the change, as well as the entire path to the entry being changed.

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

If you have Splunk Cloud, you must use the universal forwarder to collect data from the Windows Registry and forward it to your Splunk Cloud deployment.

**Why monitor the Registry?**

The Registry is probably the most used, yet least understood component of Windows operation. Many programs and processes read from and write to it at all times. When something is not functioning, Microsoft often instructs administrators and users alike to make changes to the Registry directly using the RegEdit tool. The ability to capture those edits, and any other changes, in real time is the first step in understanding the importance of the Registry.

Registry health is very important. Splunk Enterprise tells you when changes to the Registry are made and also if those changes were successful. If programs
and processes can’t write to or read from the Registry, a system failure can occur. Splunk Enterprise can alert you to problems interacting with the Registry so that you can restore it from a backup and keep your system running.

**What do you need to monitor the Registry?**

The following table lists the explicit permissions you need to monitor the Registry. You might need additional permissions based on the Registry keys that you want to monitor.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Required permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor the Registry</td>
<td>* Splunk Enterprise must run on Windows AND * Splunk Enterprise must run as either the local system user OR * Splunk Enterprise must run as a domain user with read access to the Registry hives or keys that you want to monitor</td>
</tr>
</tbody>
</table>

**Performance considerations**

When you enable Registry monitoring, you specify which Registry hives to monitor: the user hive (represented as `HKEY_USERS` in RegEdit) and/or the machine hive (represented as `HKEY_LOCAL_MACHINE`). The user hive contains user-specific configurations required by Windows and programs, and the machine hive contains configuration information specific to the machine, such as the location of services, drivers, object classes and security descriptors.

Because the Registry plays a central role in the operation of a Windows machine, enabling both Registry paths results in a lot of data for Splunk Enterprise to monitor. To achieve the best performance, filter the amount of Registry data that Splunk Enterprise indexes by configuring `inputs.conf`.

Similarly, you can capture a baseline snapshot of the current state of your Windows Registry when you first start Splunk Enterprise, and again every time a specified amount of time has passed. The snapshot lets you compare what the Registry looks like at a certain point in time and provides for easier tracking of the changes to the Registry over time.

The snapshot process can be somewhat CPU-intensive, and might take several minutes to complete. You can postpone taking a baseline snapshot until you have narrowed the scope of the Registry entries to those you specifically want.
Enable Registry monitoring in Splunk Web

Go to the Add New page

You can get there by two routes:

- Splunk Home
- Splunk Settings

By Splunk Settings:

1. Click **Settings** in the upper right corner of Splunk Web.
2. Click **Data Inputs**.
3. Click **Registry monitoring**.
4. Click **New** to add an input.

By Splunk Home:

1. Click the **Add Data** link in Splunk Home.
2. Click **Monitor** to monitor Registry data on the local Windows machine.

Select the input source

1. In the left pane, locate and select **Registry monitoring**.
2. In the **Collection Name** field, enter a unique name for the input that you will remember.
3. In the **Registry hive** field, enter the path to the Registry key that you want Splunk Enterprise to monitor.
4. (Optional) If you are not sure of the path, click the **Browse** button to select the Registry key path that you want Splunk Enterprise to monitor. The Registry hive window opens and displays the Registry in tree view. Hives, keys and subkeys display as folders, and values display as document icons. The **HKEY_USERS**, **HKEY_CURRENT_USER**, **HKEY_LOCAL_MACHINE**, and **HKEY_CURRENT_CONFIG** hives display as top-level objects. The **HKEY_CLASSES_ROOT** hive is not shown because of the number of subkeys present in the first sublevel of that hive. To access **HKEY_CLASSES_ROOT** items, choose **HKEY_LOCAL_MACHINE\Software\Classes**.
5. In the **Registry hive** window, choose the desired Registry key by clicking on the name of the key. The qualified key name appears in the **Qualified name** field at the bottom of the window.
6. Click **Select** to confirm the choice and close the window.
7. (Optional) Select **Monitor subnodes** if you want to monitor the child nodes below the starting hive.

The **Monitor subnodes** node determines what Splunk Enterprise adds to the inputs.conf file that it creates when you define a Registry monitor input in Splunk Web.

If you use the tree view to select a key or hive to monitor and check **Monitor subnodes**, then Splunk Enterprise adds a **regular expression** to the stanza for the input you are defining. This regular expression (`\\\?\.*`) filters out events that do not directly reference the selected key or any of its subkeys.

If you do not check **Monitor subnodes**, then Splunk Enterprise adds a regular expression to the input stanza which filters out events that do not directly reference the selected key (including events that reference subkeys of the selected key.)

If you do not use the tree view to specify the desired key to monitor, then Splunk Enterprise adds the regular expression only if you have checked **Monitor subnodes** and have not entered your own regular expression in the **Registry hive** field.

8. Under **Event types**, select the Registry event types that you want Splunk Enterprise to monitor for the chosen Registry hive:

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Set</strong></td>
<td>Splunk Enterprise generates a Set event when a program executes a SetValue method on a Registry subkey, thus setting a value or overwriting an existing value on an existing Registry entry.</td>
</tr>
<tr>
<td><strong>Create</strong></td>
<td>Splunk Enterprise generates a Create event when a program executes a CreateSubKey method within a Registry hive, thus creating a new subkey within an existing Registry hive.</td>
</tr>
<tr>
<td><strong>Delete</strong></td>
<td>Splunk Enterprise generates a Delete event when a program executes a DeleteValue or DeleteSubKey method. This method either removes a value for a specific existing key, or removes a key from an existing hive.</td>
</tr>
<tr>
<td><strong>Rename</strong></td>
<td>Splunk Enterprise generates a Rename event when you rename a Registry key or subkey in RegEdit.</td>
</tr>
<tr>
<td><strong>Open</strong></td>
<td></td>
</tr>
</tbody>
</table>
Splunk Enterprise generates an Open event when a program executes an OpenSubKey method on a Registry subkey, such as what happens when a program needs configuration information contained in the Registry.

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close</td>
<td>Splunk Enterprise generates a Close event when a program executes a Close method on a Registry key. This happens when a program is done reading the contents of a key, or after you make a change to a key's value in RegEdit and exit the value entry window.</td>
</tr>
<tr>
<td>Query</td>
<td>Splunk Enterprise generates a Query event when a program executes the GetValue method on a Registry subkey.</td>
</tr>
</tbody>
</table>

9. Specify which processes Splunk Enterprise should monitor for changes to the Registry by entering appropriate values in the **Process Path** field. Or, leave the default of `C:\.*` to monitor all processes.

10. Specify whether or not you want to take a baseline snapshot of the whole Registry before monitoring Registry changes. To set a baseline, click **Yes** under **Baseline index**.

   The baseline snapshot is an index of your entire Registry, at the time the snapshot is taken. Registry events within the snapshot retain their original indexing timestamps. Scanning the Registry to set a baseline index is a CPU-intensive process and might take some time.

11. Click **Next**.

**Specify input settings**

The **Input Settings** page lets you specify application context, default host value, and index. All of these parameters are optional.

1. Select the appropriate **Application context** for this input.
2. Set the **Host** name value. You have several choices for this setting. Learn more about setting the host value in **About hosts**.
   **Host** only sets the **host** field in the resulting events. It does not direct Splunk Enterprise to look on a specific host on your network.
3. Set the **Index** that Splunk Enterprise should send data to. 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 Enterprise has a number of utility indexes, which also appear in this dropdown box.
4. Click **Review**.
Review your choices

After specifying all your input settings, review your selections. Splunk Enterprise lists all options you selected, including the type of monitor, the source, the source type, the application context, and the index.

1. Review the settings.
2. If they do not match what you want, click < to go back to the previous step in the wizard. Otherwise, click Submit.

Splunk Enterprise then loads the "Success" page and begins indexing the specified Registry nodes.

View Registry change data

To view Registry change data that Splunk Enterprise indexed, go to the Search app and search for events with a source of WinRegistry. An example event, which Group Policy generates when a user logs in to a domain, follows:

3:03:28.505 PM 06/19/2011 15:03:28.505
event_status="(0) The operation completed successfully."
pid=340
process_image="c:\WINDOWS\system32\winlogon.exe"
registry_type="SetValue"
key_path="HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Group Policy\History\DCName"
data_type="REG_SZ"
data="\ftw.ad.splunk.com"

Each registry monitoring event contains the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
</table>
| event_status | The result of the registry change attempt. This should always be
                   "(0) The operation completed successfully.". If it is not,
                   there might be problems with the Registry that might eventually
                   require a restore from a backup.                              |
<p>| pid          | The process ID of the process that attempted to make the Registry change.    |
| process_image| The name of the process that attempted to make the Registry change.           |
| registry_type| The type of Registry operation that the process_image attempted to invoke.   |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key_path</td>
<td>The Registry key path that the process_image attempted to make a change to.</td>
</tr>
<tr>
<td>data_type</td>
<td>The type of Registry data that the process_image making the Registry change tried to get or set.</td>
</tr>
<tr>
<td>data</td>
<td>The data that the process_image making the Registry change tried to read or write.</td>
</tr>
</tbody>
</table>

### Filter incoming Registry events

Windows Registries generate a great number of events due to their near-constant use. This can cause problems with licensing. Splunk Registry monitoring can generate hundreds of megabytes of data per day.

Splunk Windows Registry monitoring uses a configuration file to determine what to monitor on your system, inputs.conf. This file needs to reside in $SPLUNK_HOME\etc\system\local\ on the server that runs Registry monitoring.

inputs.conf contains the specific regular expressions you create to refine and filter the Registry hive paths you want Splunk to monitor.

Each stanza in inputs.conf represents a particular filter whose definition includes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>proc</td>
<td>A regular expression containing the path to the process or processes you want to monitor.</td>
</tr>
</tbody>
</table>
| hive      | A regular expression that contains 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 |
There is no direct mapping for HKEY_CURRENT_USER or HKCU, as the Registry monitor runs in kernel mode. Use `\REGISTRY\USER\.*` (note the period and asterisk at the end) to generate events that contain the security identifier (SID) of the logged-in user.

- 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 user.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>The subset of event types to monitor. Can be one or more of delete, set, create, rename, open, close or query. The values here must be a subset of the values for event_types that you set in inputs.conf.</td>
</tr>
<tr>
<td>baseline</td>
<td>Whether or not to capture a baseline snapshot for that particular hive path. Set to 1 for yes, and 0 for no.</td>
</tr>
<tr>
<td>baseline_interval</td>
<td>How much time, in seconds, must have elapsed since the last baseline was taken before Splunk Enterprise takes another baseline on startup. For example, if you set baseline_interval to 600, then when Splunk Enterprise starts or restarts, it takes a baseline if the existing baseline is more than 600 seconds old. If no baseline exists, then Splunk Enterprise takes a baseline immediately. This setting has no effect if you do not also set baseline to 1. The default value is 86,400 seconds (1 day).</td>
</tr>
<tr>
<td>disabled</td>
<td>Whether or not a filter is enabled. Set to 1 to disable the filter, and 0 to enable it.</td>
</tr>
</tbody>
</table>

Get a baseline snapshot

When you enable Registry monitoring, you can record a baseline snapshot of the Registry hives the next time Splunk Enterprise starts. By default, the snapshot covers the HKEY_CURRENT_USER and HKEY_LOCAL_MACHINE hives. It also establishes a timeline for when to retake the snapshot. By default, if the baseline is more than 24 hours old, when Splunk Enterprise next starts, it retakes the baseline snapshot. You can customize this value for each of the filters in inputs.conf by setting the value of baseline_interval, in seconds.

When you create a baseline snapshot, the snapshot uses the index time of the Registry data, not the snapshot creation time. For example, if a change to a Registry key occurred two years ago, the timestamp for that event will be two
years ago, not when the baseline snapshot was created.

**Monitor Windows performance**

Splunk Enterprise supports the monitoring of all Windows performance counters in real time and includes support for both local and remote collection of performance data.

The Splunk Enterprise performance monitoring utility gives you the abilities of Performance Monitor in a web interface. Splunk Enterprise uses the Performance Data Helper (PDH) API for performance counter queries on local machines.

The types of performance objects, counters and instances that are available to Splunk Enterprise depend on the performance libraries installed on the system. Both Microsoft and third-party vendors provide libraries that contain performance counters. For information on performance monitoring, see "Performance Counters" on MSDN.

Both full instances of Splunk Enterprise and universal forwarders support local collection of performance metrics. Remote performance monitoring is available through WMI (Windows Management Instrumentation) and requires that Splunk Enterprise runs as a user with appropriate Active Directory credentials. If you have Splunk Cloud and want to monitor Windows performance metrics, you must use the Splunk universal Forwarder to collect the data and forward it to your Splunk Cloud deployment.

The performance monitor input runs as a process called `splunk-perfmon.exe`. It runs once for every input defined, at the interval specified in the input. You can configure performance monitoring with Splunk Web, or either `inputs.conf` (for local performance data) or `wmi.conf` (for performance data from a remote machine).

**Why monitor performance metrics?**

Performance monitoring is an important part of the Windows administrator's toolkit. Windows generates a lot of data about a system's health. Proper analysis of that data can make the difference between a healthy, well functioning system, and one that suffers downtime.
What do you need to monitor performance counters?

The following table lists the permissions you need to monitor performance counters in Windows. You might need additional permissions based on the performance objects or counters that you want to monitor.

For additional information on performance metrics monitoring requirements, see Security and remote access considerations.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Required permissions</th>
</tr>
</thead>
</table>
| Monitor local performance metrics             | * Splunk Enterprise must run on Windows.  
 * Splunk Enterprise must run as the Local System user. |
| Monitor remote performance metrics on another computer over WMI | * Splunk Enterprise must run on Windows.  
 * Splunk Enterprise must run as a domain or remote user with at least read access to WMI on the target computer.  
 * Splunk Enterprise must run as a domain or remote user with appropriate access to the Performance Data Helper libraries on the target computer. |

Security and remote access considerations

Splunk Enterprise gets data from remote machines with either a forwarder or WMI. Splunk recommends using a universal forwarder to send performance data from remote machines to an indexer.

If you install forwarders on your remote machines to collect performance data, then you can install the forwarder as the Local System user on those machines. The Local System user has access to all data on the local machine, but not to remote computers.

If you want Splunk Enterprise to use WMI to get performance data from remote machines, then you must configure both Splunk Enterprise and your Windows network. You cannot install Splunk Enterprise as the Local System user, and the user that you choose determines what Performance Monitor objects that Splunk Enterprise can see.

After you install Splunk Enterprise with a valid user, you must add that user to the following groups before you enable local performance monitor inputs:
• Performance Monitor Users (domain group)
• Performance Log Users (domain group)

To learn more about WMI security, see Security and remote access considerations in "Monitor WMI Data". To learn about how to use a universal forwarder, see About the universal forwarder.

Enable local Windows performance monitoring

You can configure local performance monitoring either in Splunk Web or with configuration files.

Splunk Web is the preferred way to add performance monitoring data inputs. You can make typos with configuration files, and it is important to specify performance monitor objects exactly as the Performance Monitor API defines them. See "Important information about specifying performance monitor objects in inputs.conf" later in this topic for a full explanation.

Configure local Windows performance monitoring with Splunk Web

Go to the Add New page

You can get there by two routes:

• Splunk Home
• Splunk Settings

By Splunk Settings:

1. Click Settings in the upper right corner of Splunk Web.
2. Click Data Inputs.
3. Click Local performance monitoring.
4. Click New to add an input.

By Splunk Home:

1. Click the Add Data link in Splunk Home.
2. Click Monitor to monitor performance data from the local Windows machine, or Forward to receive performance data from another machine.
3. If you selected Forward, choose or create the group of forwarders you want this input to apply to.
4. Click **Next**.

**Select the input source**

1. In the left pane, locate and select **Local Performance Monitoring**.
2. In the **Collection Name** field, enter a unique name for this input that you will remember.
3. Click **Select Object** to get a list of the performance objects available on this Windows machine, then choose the object that you want to monitor from the list. Splunk Enterprise displays the "Select Counters" and "Select Instances" list boxes.

You can only add one performance object per data input. This is due to how Microsoft handles performance monitor objects. Many objects enumerate classes that describe themselves dynamically upon selection. This can lead to confusion as to which performance counters and instances belong to which object, as defined in the input. If you need to monitor multiple objects, create additional data inputs for each object.

4. In the **Select Counters** list box, locate the performance counters you want this input to monitor.
5. Click once on each counter you want to monitor. Splunk Enterprise moves the counter from the "Available counter(s)" window to the "Selected counter(s)" window.
6. To unselect a counter, click on its name in the "Available Items" window. Splunk Enterprise moves the counter from the "Selected counter(s)" window to the "Available counter(s)" window.
7. To select or unselect all of the counters, click on the "add all" or "remove all" links.

Selecting all of the counters can result in the indexing of a lot of data and possibly lead to license violations.

8. In the **Select Instances** list box, select the instances that you want this input to monitor by clicking once on the instance in the "Available instance(s)" window. Splunk Enterprise moves the instance to the "Selected instance(s)" window.

The "_Total" instance is a special instance, and appears for many types of performance counters. This instance is the average of any associated instances under the same counter. Data collected for this instance can be significantly different than for individual instances under the same counter. For example, when you monitor performance data for the "Disk Bytes/Sec"
performance counter under the "PhysicalDisk" object on a system with two disks installed, the available instances include one for each physical disk - "0 C:" and "1 D:" - and the "_Total" instance, which is the average of the two physical disk instances.

9. In the **Polling interval** field, enter the time, in seconds, between polling attempts for the input.
10. Click the green **Next** button.

**Specify input settings**

The **Input Settings** page lets you specify application context, default host value, and index. All of these parameters are optional.

Setting the **Host** on this page only sets the **host** field in the resulting events. It does not direct Splunk Enterprise to look on a specific host on your network.

1. Select the appropriate **Application context** for this input.
2. Set the **Host** name value. You have several choices for this setting. Learn more about setting the host value in **About hosts**.
3. Set the **Index** that Splunk Enterprise should send data to. 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 Enterprise has a number of utility indexes, which also appear in this dropdown box.
4. Click **Review**.

**Review your choices**

After you specify input settings, review your selections. Splunk Enterprise lists all options you selected, including the type of monitor, the source, the source type, the application context, and the index.

1. Review the settings.
2. If they do not match what you want, click < to go back to the previous step in the wizard. Otherwise, click **Submit**.

Splunk Enterprise then loads the "Success" page and begins indexing the specified performance metrics. For more information on getting data from files and directories, see **Monitor Windows performance** in this manual.
Configure local Windows performance monitoring with configuration files

inputs.conf controls performance monitoring configurations. To set up performance monitoring using configuration files, you must create or edit inputs.conf in %SPLUNK_HOME%\etc\system\local. If you have not worked with configuration files before, see About configuration files.

The [perfmon://<name>] stanza defines performance monitoring inputs in inputs.conf. You specify one stanza per performance object that you wish to monitor.

In each stanza, you can specify the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>interval</td>
<td>Yes</td>
<td>How often, in seconds, to poll for new data. If this attribute is not present, the input runs every 300 seconds (5 minutes).</td>
</tr>
<tr>
<td>object</td>
<td>Yes</td>
<td>The performance object(s) that you want to capture. Specify either a string which exactly matches (including case) the name of an existing Performance Monitor object or use a regular expression to reference multiple objects. If this attribute is not present and defined, the input will not run, as there is no default.</td>
</tr>
<tr>
<td>counters</td>
<td>Yes</td>
<td>One or more valid performance counters that are associated with the object specified in object. Separate multiple counters with semicolons. You can also use an asterisk (*) to specify all available counters under a given object. If this attribute is not present and defined, the input will not run, there is no default.</td>
</tr>
<tr>
<td>instances</td>
<td>No</td>
<td>One or more valid instances associated with the performance counter specified in counters. Multiple instances are separated by semicolons. Specify all instances by using an asterisk (*), which is the default if you do not define the attribute in the stanza.</td>
</tr>
<tr>
<td>index</td>
<td>No</td>
<td>The index to route performance counter data to. If not present, the default index is used.</td>
</tr>
<tr>
<td>disabled</td>
<td>No</td>
<td>Whether or not to gather the performance data defined in this input. Set to 1 to disable this stanza, and 0 to enable it. If not present, it defaults to 0 (enabled).</td>
</tr>
<tr>
<td>Attribute</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>showZeroValue</td>
<td>No</td>
<td><strong>Advanced option.</strong> Whether or not Splunk Enterprise should collect events that have values of zero. Set to 1 to collect zero-value events, and 0 to ignore these events. If not present, it defaults to 0 (ignore zero-value events.)</td>
</tr>
<tr>
<td>samplingInterval</td>
<td>No</td>
<td><strong>Advanced option.</strong> How often, in milliseconds, that Splunk should collect performance data. Enables high-frequency performance sampling. When you enable high-frequency performance sampling, Splunk Enterprise collects performance data every interval and reports the average of the data as well as other statistics. It defaults to 100 ms, and must be less than what you specify with the interval attribute.</td>
</tr>
<tr>
<td>stats</td>
<td>No</td>
<td><strong>Advanced option.</strong> A semicolon-separated list of statistic values that Splunk Enterprise reports for high-frequency performance sampling. Allowed values are: average, min, max, dev, and count. The default is no setting (disabled).</td>
</tr>
<tr>
<td>mode</td>
<td>No</td>
<td><strong>Advanced option.</strong> When you enable high-performance sampling, this attribute controls how Splunk Enterprise outputs events. Allowed values are: single, multikv, multiMS, and multikvMS When you enable either multiMS or multikvMS, Splunk Enterprise outputs two events for each performance metric it collects. The first event is the average value, and the second is the statistics event. The statistics event has a special sourcetype depending on which output mode you use (perfmonMSStats for multiMS and perfmonMKMSStats for multikvMS) If you do not enable high-performance sampling, the multikvMS output mode is the same as the multikv output mode.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| `useEnglishOnly`         | No        | **Advanced option.** Controls how Splunk Enterprise indexes performance metrics on systems whose locale is not English. Specifically, it dictates which Windows Performance Monitor API to use when it indexes performance metrics on hosts that do not use the English language.  
If set to true, Splunk Enterprise collects the performance metrics in English regardless of the system locale. It uses the `PdhAddEnglishCounter()` API to add the counter string. It also disables regular expression and wildcard matching for the object and counter attributes.  
If set to false, Splunk Enterprise collects the performance metrics in the system language and expects you to configure the object and counter attributes in that language. It uses the `PdhAddCounter()` API to add the counter string. You can use wildcards and regular expressions, but you must specify valid object, counters, and instances values that are specific to the locale of the operating system.  
The default is false. |
| `useWinApiProcStats`    | No        | **Advanced option.** When enabled, `useWinApiProcStats` the Performance Monitor input uses process kernel mode and user mode times to calculate CPU usage for a process. Currently, the input uses the standard Performance Data Helper (PDH) APIs to calculate CPU usage for a process.  
When this setting is configured to "true", the input uses the `GetProcessTime()` function in the core Windows API to calculate CPU usage for a process, for the following Performance Monitor counters, only:  
- Processor Time  
- User Time  
- Privileged Time  
It is a best practice to enable the `useWinApiProcStats` function on multicore Windows machines. |
Identify how many processors are in your system. The total number of cores can be verified on the "system information" page in your Windows deployment.

Each processor core in your system is equal to a maximum processor performance percentage of 100%. So for each core in a multicore windows deployment, 100% is added to the total maximum available processor performance percentage. For example, an 8 core windows environment will have a maximum process capability of 800%.

Total processor capability can be validated in your Splunk platform deployment by navigating to **Data inputs > Local performance monitoring> Select system.**

the APIs this setting uses are English only. If your Windows machine uses a non-English system locale, you must also set `useEnglishOnly` to `true`.

See Performance Monitor inputs show maximum values of 100 percent usage for a process on multicore Microsoft Windows machines in the release notes for more information on calculating CPU usage on Windows multicore machines.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>formatString</td>
<td>No</td>
<td><strong>Advanced option.</strong> Controls how Splunk Enterprise formats the output of floating-point values for performance counter events. Windows often prints performance counter events as floating point values. When not formatted, the events print with all significant digits to the right of the decimal point. The <code>formatString</code> attribute controls the number of significant digits that print as part of each event. The attribute uses format specifiers from the C++ <code>printf</code> function. The function includes many kinds of specifiers, depending on how you want to output the event text. A reference with examples can be found at &quot;printf - C++ reference&quot;(<a href="http://www.cplusplus.com/reference/cstdio/printf/">http://www.cplusplus.com/reference/cstdio/printf/</a>) on cplusplus.com.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When specifying the format, do not use quotes (&quot;&quot;'). Specify only the valid characters needed to format the string the way you want.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is %.20g.</td>
</tr>
</tbody>
</table>

**Collect performance metrics in English regardless of system locale**

You can collect performance metrics in English even if the system that Splunk Enterprise runs on does not use the English language.

To do this, use the `useEnglishOnly` attribute in stanzas within `inputs.conf`. There is no way to configure `useEnglishOnly` in Splunk Web.

**Note:** There are caveats to using `useEnglishOnly` in an `inputs.conf` stanza. See **Caveats** later in this topic.

**Examples of performance monitoring input stanzas**

Following are some example stanzas that show you how to use `inputs.conf` to monitor performance monitor objects.

```
# Query the PhysicalDisk performance object and gather disk access data for
# all physical drives installed in the system. Store this data in the
# "perfmon" index.
# Note: If the interval attribute is set to 0, Splunk resets the
# interval to 1.
[perfmon://LocalPhysicalDisk]
interval = 0
object = PhysicalDisk
counters = Disk Bytes/sec; % Disk Read Time; % Disk Write Time; % Disk
Time
instances = *
disabled = 0
index = PerfMon

# Gather SQL statistics for all database instances on this SQL server.
# 'object' attribute uses a regular expression "\$.\.*" to specify SQL
# statistics for all available databases.
[perfmon://SQLServer_SQL_Statistics]
object = MSSQL\$.\.*:SQL Statistics
counters = *
```
instances = *

# Gather information on all counters under the "Process" and "Processor" # Perfmon objects.
# We use ".*" as a wild card to match the 'Process' and 'Processor' objects.
[perfmon://ProcessandProcessor]
object = Process.*
counters = *
instances = *

# Collect CPU processor usage metrics in English only on a French system.
[perfmon://Processor]
object = Processor
instances = _Total
counters = % Processor Time;% User Time
useEnglishOnly = 1
interval = 30
disabled = 0

# Collect CPU processor usage metrics in the French system's native locale.
# Note that you must specify the counters in the language of that locale.
[perfmon://FrenchProcs]
counters = *
disabled = 0
useEnglishOnly = 0
interval = 30
object = Processeur
instances = *

# Collect CPU processor usage metrics. Format the output to two decimal places only.
[perfmon://Processor]
counters = *
disabled = 0
interval = 30
object = Processor
instances = *
formatString = %.20g

Important information about specifying performance monitor objects in inputs.conf

Use all lower case when specifying the perfmon keyword

When you create a performance monitor input in inputs.conf, you must use all lower case for the perfmon keyword, for example:
If you use capital or mixed-case letters for the keyword, Splunk Enterprise warns of the problem on start-up, and the specified performance monitor input does not run.

**Specify valid regular expressions to capture multiple performance monitor objects**

To specify multiple objects in a single performance monitor stanza, you must use a valid regular expression to capture those objects. For example, to specify a wildcard to match a string beyond a certain number of characters, do not use `*`, but rather `.\*`. If the object contains a dollar sign or similar special character, you might need to escape it with a backslash (`\`).

**Values must exactly match what is in the Performance Monitor API if you do not use regular expressions**

When you specify values for the `object`, `counters` and `instances` attributes in `[perfmon://]` stanzas, be sure that those values exactly match those defined in the Performance Monitor API, including case, or the input might return incorrect data, or no data at all. If the input cannot match a performance object, counter, or instance value that you've specified, it logs that failure to `splunkd.log`. For example:

```
01-27-2011 21:04:48.681 -0800 ERROR ExecProcessor - message from "C:\Program Files\Splunk\bin\splunk-perfmon.exe" -noui" splunk-perfmon - PerfmonHelper::enumObjectByNameEx: PdhEnumObjectItems failed for object - 'USB' with error (0xc0000bb8): The specified object is not found on the system.
```

Use Splunk Web to add performance monitor data inputs to ensure that you add them correctly.

**Enable remote Windows performance monitoring over WMI**

You can configure remote performance monitoring either in Splunk Web or by using configuration files.

When you collect performance metrics over WMI, you must configure Splunk Enterprise to run as an AD user with appropriate access for remote collection of performance metrics. You must do this before attempting to collect those metrics. Both the machine that runs Splunk Enterprise and the machine(s) Splunk collects
performance data from must reside in the same AD domain or forest.

WMI self-throttles by design to prevent denial-of-service attacks. Splunk Enterprise also reduces the number of WMI calls it makes over time as a precautionary measure if these calls return an error. Depending on the size, configuration, and security profile of your network, installing a local forwarder on the host that you want to collect performance metrics might be a better choice. See Considerations for deciding how to monitor remote Windows data in this manual.

**WMI-based performance values versus Performance Monitor values**

When you gather remote performance metrics through WMI, some metrics return zero values or values that are not in line with values that Performance Monitor returns. A limitation in the implementation of WMI for performance monitor counters causes this problem. This is not an issue with Splunk Enterprise or how it retrieves WMI-based data.

WMI uses the `Win32_PerfFormattedData_*` classes to gather performance metrics. More info on the specific classes is available at "Win32 Classes" on MSDN.

WMI defines the data structures within these classes as either 32- or 64-bit unsigned integers, depending on the version of Windows you run. The PDH API defines Performance Monitor objects as floating-point variables. This means that you might see WMI-based metrics that appear anomalous, due to rounding factors.

For example, if you collect data on the "Average Disk Queue Length" Performance Monitor counter at the same time you collect the `Win32_PerfFormattedData_PerfDisk_PhysicalDisk\AvgDiskQueueLength` metric through WMI, the WMI-based metric might return zero values even though the Performance Monitor metric returns values that are greater than zero (but less than 0.5). This is because WMI rounds the value down before displaying it.

If you require additional granularity in your performance metrics, configure the performance monitoring inputs on a universal forwarder on each machine from which you wish to collect performance data. You can then forward that data to an indexer. Data retrieved using this method is more reliable than data gathered remotely using WMI-based inputs.
Configure remote Windows performance monitoring with Splunk Web

Go to the Add New page

You can get there by two routes:

- Splunk Home
- Splunk Settings

By Splunk Settings:

1. Click Settings in the upper right corner of Splunk Web.
2. Click Data Inputs.
3. Click Remote performance monitoring.
4. Click New to add an input.

By Splunk Home:

1. Click the Add Data link in Splunk Home.
2. Click Monitor to monitor performance data from the local Windows machine, or Forward to forward performance data from another Windows machine. Splunk Enterprise loads the "Add Data - Select Source" page.

   Forwarding performance data requires additional setup.

3. In the left pane, locate and select Local Performance Monitoring.

Select the input source

Win32_PerfFormattedData_* classes do not show up as available objects in Splunk Web. If you want to monitor Win32_PerfFormattedData_* classes, you must add them directly in wmi.conf.

1. In the Collection Name field, enter a unique name for this input that you will remember.
2. In the Select Target Host field, enter the host name or IP address of the Windows computer you want to collect performance data from.
3. Click "Query" to get a list of the performance objects available on the Windows machine you specified in the "Select Target Host" field.
4. Choose the object that you want to monitor from the Select Class list. Splunk Enterprise displays the "Select Counters" and "Select Instances" list boxes.
You can only add one performance object per data input. This is due to how Microsoft handles performance monitor objects. Many objects enumerate classes that describe themselves dynamically upon selection. This can lead to confusion as to which performance counters and instances belong to which object, as defined in the input. If you need to monitor multiple objects, create additional data inputs for each object.

5. In the **Select Counters** list box, locate the performance counters you want this input to monitor.

6. Click once on each counter you want to monitor. Splunk Enterprise moves the counter from the "Available counter(s)" window to the "Selected counter(s)" window.

7. To unselect a counter, click on its name in the "Available Items" window. Splunk Enterprise moves the counter from the "Selected counter(s)" window to the "Available counter(s)" window.

8. To select or unselect all of the counters, click on the "add all" or "remove all" links. **Important**: Selecting all of the counters can result in the indexing of a lot of data, possibly more than your license allows.

9. In the **Select Instances** list box, select the instances that you want this input to monitor by clicking once on the instance in the "Available instance(s)" window. Splunk Enterprise moves the instance to the "Selected instance(s)" window.

   The "_Total" instance is a special instance, and appears for many types of performance counters. This instance is the average of any associated instances under the same counter. Data collected for this instance can be significantly different than for individual instances under the same counter. For example, when you monitor performance data for the "Disk Bytes/Sec" performance counter under the "PhysicalDisk" object on a host with two disks installed, the available instances include one for each physical disk - "0 C:" and "1 D:" - and the "_Total" instance, which is the average of the two physical disk instances.

10. In the **Polling interval** field, enter the time, in seconds, between polling attempts for the input.

11. Click **Next**.

**Specify input settings**

The **Input Settings** page lets you specify application context, default host value, and index. All of these parameters are optional.
Setting the **Host** only sets the **host** field in the resulting events. It does not direct Splunk Enterprise to look on a specific host on your network.

1. Select the appropriate **Application context** for this input.
2. Set the **Host** name value. You have several choices for this setting. Learn more about setting the host value in About hosts.
3. Set the **Index** that Splunk Enterprise should send data to. 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 Enterprise has a number of utility indexes, which also appear in this dropdown box.
4. Click the green **Review** button.

**Review your choices**

After specifying all your input settings, you can review your selections. Splunk Enterprise lists all options you selected, including the type of monitor, the source, the source type, the application context, and the index.

1. Review the settings.
2. If they do not match what you want, click < to go back to the previous step in the wizard. Otherwise, click **Submit**.

Splunk Enterprise then loads the "Success" page and begins indexing the specified performance metrics.

For more information on getting performance monitor data from remote machines, see Monitor WMI data in this manual.

**Configure remote Windows performance monitoring with configuration files**

wmi.conf controls remote performance monitoring configurations. To set up remote performance monitoring using configuration files, create and/or edit wmi.conf in %SPLUNK_HOME%/etc/system/local. If you haven’t worked with configuration files before, read About configuration files before you begin.

Use Splunk Web to create remote performance monitor inputs unless you do not have access to it. The names of performance monitor objects, counters, and instances must exactly match what the Performance Monitor API defines, including case. Splunk Web uses WMI to get the properly-formatted names, eliminating the potential for typos.
wmi.conf contains one stanza for each remote performance monitor object that you want to monitor. In each stanza, you specify the following content.

**Global settings**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required?</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial_backoff</td>
<td>No</td>
<td>How long, in seconds, to wait before retrying a connection to a WMI provider when an error occurs. If problems persist on connecting to the provider, then the wait time between connection attempts doubles until either it can connect, or until the wait time is greater than or equal to the max_backoff attribute.</td>
<td>5</td>
</tr>
<tr>
<td>max_backoff</td>
<td>No</td>
<td>The maximum amount of time, in seconds to attempt to reconnect to a WMI provider.</td>
<td>20</td>
</tr>
<tr>
<td>max_retries_at_max_backoff</td>
<td>No</td>
<td>How many times, after max_backoff seconds has been reached between reconnection attempts with a WMI provider, to continue to attempt to reconnect to that provider.</td>
<td>2</td>
</tr>
<tr>
<td>checkpoint_sync_interval</td>
<td>No</td>
<td>How long, in seconds, to wait for state data to be flushed to disk.</td>
<td>2</td>
</tr>
</tbody>
</table>

**Input-specific settings**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required?</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>interval</td>
<td>Yes</td>
<td>How often, in seconds, to poll for new data. If this attribute is not present, the input will not run, as there is no default.</td>
<td>N/A</td>
</tr>
<tr>
<td>Attribute</td>
<td>Required?</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>----------------------------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>server</td>
<td>No</td>
<td>A comma-separated list of one or more valid hosts on which you want to monitor performance.</td>
<td>The local machine</td>
</tr>
<tr>
<td>event_log_file</td>
<td>No</td>
<td>The names of one or more Windows event log channels to poll. This attribute tells Splunk Enterprise that the incoming data is in event log format.</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Do not use the <code>event_log_file</code> attribute in a stanza that already contains the <code>wql</code> attribute.</td>
<td></td>
</tr>
<tr>
<td>wql</td>
<td>No</td>
<td>A valid Windows Query Language (WQL) statement that specifies the performance objects, counters, and instances you want to poll remotely. This attribute tells Splunk Enterprise to expect data from a WMI provider.</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Do not use the <code>wql</code> attribute in a stanza that already contains the <code>event_log_file</code> attribute.</td>
<td></td>
</tr>
<tr>
<td>namespace</td>
<td>No</td>
<td>The namespace in which the WMI provider you want to query resides. The value for this attribute can be either relative (<code>Root\CIMV2</code>) or absolute (<code>\SERVER\Root\CIMV2</code>), but must be relative if you specify the <code>server</code> attribute.</td>
<td>Root\CIMV2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Only use the <code>namespace</code> attribute in a stanza that contains the <code>wql</code> attribute.</td>
<td></td>
</tr>
<tr>
<td>index</td>
<td>No</td>
<td>The desired index to route performance counter data to.</td>
<td>default</td>
</tr>
<tr>
<td>current_only</td>
<td>No</td>
<td>The characteristics and interaction of WMI-based event collections.</td>
<td>N/A</td>
</tr>
</tbody>
</table>
- if `wql` is defined, this attribute tells Splunk Enterprise whether or not it should expect an event notification query. Set to 1 to tell Splunk to expect an event notification query, and 0 to tell it expect a standard query. See below for additional requirements on WQL and event notification queries.

- if `event_log_file` is defined, tells Splunk whether or not to only capture events that occur when Splunk is running. Set to 1 to tell Splunk to only capture events that occur when Splunk is running, and 0 to gather events from the last checkpoint or, if no checkpoint exists, the oldest events available.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required?</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>disabled</td>
<td>No</td>
<td>Tells Splunk whether or not to gather the performance data defined in this input. Set this to 1 to disable performance monitoring for this stanza, and 0 to enable it.</td>
<td>0</td>
</tr>
</tbody>
</table>

**Examples of using wmi.conf**

The following example of `wmi.conf` gathers local disk and memory performance metrics and places them into the 'wmi_perfmon' index:

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

# Gather disk and memory performance metrics from the local system every second.
# Store event in the "wmi_perfmon" Splunk index.
```

176
[WMI:LocalPhysicalDisk]
interval = 1
wql = select Name, DiskBytesPerSec, PercentDiskReadTime, PercentDiskWriteTime, PercentDiskTime from \n   Win32_PerfFormattedData_PerfDisk_PhysicalDisk
disabled = 0
index = wmi_perfmon

[WMI:LocalMainMemory]
interval = 10
wql = select CommittedBytes, AvailableBytes, PercentCommittedBytesInUse, Caption from \n   Win32_PerfFormattedData_PerfOS_Memory
disabled = 0
index = wmi_perfmon

Additional information on WQL query statements

WQL queries must be structurally and syntactically correct. If they are not, you might get undesirable results or no results at all. In particular, when writing event notification queries (by specifying current_only=1 in the stanza in which a WQL query resides), your WQL statement must contain one of the clauses that specify such a query (WITHIN, GROUP, and/or HAVING). Review this MSDN article on Querying with WQL for additional information.

Splunk Web eliminates problems with WQL syntax by generating the appropriate WQL queries when you use it to create performance monitor inputs.

Caveats to using the performance monitoring input

Increased memory usage during collection of performance metrics

When you collect data on some performance objects, such as the "Thread" object and its associated counters, you might notice increased memory usage in Splunk. This is normal, as certain performance objects consume more memory than others during the collection process.

Processor Time counters do not return values of higher than 100

Due to how Microsoft tallies CPU usage with the Processor:% Processor Time and Process:% Processor Time counters, these counters do not return a value of more than 100 regardless of the number of CPUs or cores in the system. This is by design - these counters subtract the amount of time spent on the Idle process from 100%.
On non-English installations, the useEnglishOnly attribute has usage limitations

When you edit inputs.conf on a non-English system to enable performance monitoring, there are some limitations to how the useEnglishOnly attribute works.

If you set the attribute to true, you cannot use wildcards or regular expressions for the object and counters attributes. These attributes must contain specific entries based on valid English values as defined in the Performance Data Helper library. You can specify a wildcard for the instances attribute. Here’s an example:

```
[perfmon://Processor]
object = Processor
instances = _Total
counters = % Processor Time;% User Time
useEnglishOnly = 1
interval = 30
disabled = 0
```

The counters attribute contain values in English even though the system language is not English.

If you set the attribute to false, you can use wildcards and regular expressions for these attributes, but you must specify values based on the operating system’s language. An example of a stanza on a system running in French follows:

```
[perfmon://FrenchProcs]
counters = *
disabled = 0
useEnglishOnly = 0
interval = 30
object = Processeur
instances = *
```

Note in this example that the object attribute has been set to Processeur, which is the French equivalent of Processor. If you specify English values here, Splunk Enterprise will not find the performance object or instance.

Additional impacts of using the useEnglishOnly attribute

There are additional items to consider when using the attribute.

- When you use Splunk Web to create performance monitor inputs on a non-English operating system, it always specifies useEnglishOnly = false.
Additionally, you can enable, disable, clone, or delete these stanzas within Splunk Web. You cannot, however, edit them in Splunk Web unless the operating system's locale matches the locale specified in the stanza.

You can use Splunk Web to enable, disable, clone, or delete a performance monitor stanza with the useEnglishOnly attribute set to true. However, you cannot edit them in Splunk Web unless the system's locale is English.

Monitor Windows data with PowerShell scripts

PowerShell is a scripting language that comes with many versions of Windows. It lets you handle Windows operations from a command-line interface. You can create scripts with the language and output the results of those scripts as objects to other scripts.

Splunk Enterprise supports the monitoring of events received through PowerShell scripts. You can use the PowerShell input to run a single PowerShell command or reference a PowerShell script. Splunk Enterprise then indexes the output of these commands or scripts as events.

If you have Splunk Cloud and want to monitor script output, use the universal forwarder to consume the output and forward it to your Splunk Cloud deployment.

What do you need to monitor data with PowerShell scripts?

<table>
<thead>
<tr>
<th>Activity</th>
<th>Required permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor data with PowerShell scripts</td>
<td>Splunk Enterprise must run on Windows. Splunk Enterprise must run as the Local System user to run all PowerShell scripts. PowerShell v3.0 or later must be installed on the host. Microsoft .NET version 4.5 or later must be installed on the host.</td>
</tr>
</tbody>
</table>

Configure inputs with configuration files

1. Write a PowerShell command or script to capture the information you want.
2. On the Splunk instance that is to run the script, open a PowerShell window.
3. Copy `inputs.conf` from `%SPLUNK_HOME%/etc/system/default` to `etc/system/local`.
4. Open the `inputs.conf` and edit it to enable a Windows PowerShell input.
5. In the input, specify the command or the full path to your script in the `script` setting.
6. (Optional) Specify a schedule on which the command or script should run with the `schedule` setting.
7. Save `inputs.conf`.
8. Restart Splunk Enterprise to enable the input.

**PowerShell input configuration values**

Splunk uses the following stanzas in `inputs.conf` to monitor data gathered by PowerShell.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>script</code></td>
<td>The PowerShell command or script file to execute.</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>When you specify a script file (.ps1), prepend the script name with a period and a space (&quot;.&quot;).</td>
<td></td>
</tr>
<tr>
<td><code>schedule</code></td>
<td>How often the command or script should execute.</td>
<td>Script runs once</td>
</tr>
<tr>
<td></td>
<td>You can specify either a number to indicate the interval, in seconds, or a valid <code>cron</code> schedule format.</td>
<td></td>
</tr>
<tr>
<td><code>disabled</code></td>
<td>Whether or not to enable the input.</td>
<td>0 (enabled)</td>
</tr>
<tr>
<td></td>
<td>Set to 1 to disable and 0 to enable</td>
<td></td>
</tr>
</tbody>
</table>

Following are some examples of how to configure the input:

**Single command example:** This example runs the `Get-Process` cmdlet and pipes that output to the `Select-Object` cmdlet using the host name that Splunk software has been installed on as an argument. It runs the command every 5 minutes.

```
[powershell://Processes-EX1]
script = Get-Process | Select-Object Handles, NPM, PM, WS, VM, Id, ProcessName, @{n="SplunkHost";e={$Env:SPLUNK_SERVER_NAME}}
schedule = 0 */5 * * *
sourcetype = Windows:Process
```

**Script example:** This example runs the `getprocesses.ps1` script located in `%SPLUNK_HOME%/etc/apps/My-App`. It sets the source type for these events to
Windows:Process. The script runs every 20 minutes from 9:00am to 4:40pm on Mondays to Fridays.

[powershell://Processes-EX2]
script = . "$SplunkHome/etc/apps/My-App/bin/getprocesses.ps1"
schedule = 0 */20 9-16 * 1-5
sourcetype = Windows:Process

For information on writing PowerShell scripts, see Write scripts for the PowerShell input.

Configure inputs with Splunk Web

1. Select Settings > Data inputs from the system bar.
2. Select PowerShell v3 modular input.
3. Click New.
4. Enter an input name in the Name field.
5. Enter a command or path to a script in the Command or Script Path field.
6. Enter an interval or cron schedule in the Cron Schedule field.
7. Click the More Settings checkbox to select the source type, host, and default index.
8. Click Next.

Write scripts for the PowerShell input

Architecture

Splunk Enterprise provides one modular PowerShell input handler. The PowerShell handler supports Microsoft PowerShell version 3 and later.

The PowerShell modular input provides a single-instance, multi-threaded script host that provides a supporting schema, XML configuration through the stdin stream, and XML streaming output.

You can define many PowerShell stanzas and run them simultaneously. You can schedule each stanza through the cron syntax. Because all scripts run within the same process, scripts share environment variables such as the current working directory.

Note: The input does not set a host variable in your PowerShell environment. When you write a script for the input, do not refer to $host or use the Write-Host or Out-Host PowerShell cmdlets. Instead, use either the Write-Output or
Write-Error cmdlets.

The input converts all output to key/value pairs based on public properties that are defined in the schema.

Splunk Enterprise also includes a PowerShell module called **LocalStorage**, which exposes three cmdlets:

- `Get-LocalStoragePath`
- `Export-LocalStorage`
- `Import-LocalStorage`

These cmdlets use the Splunk Enterprise checkpoint directory and let you maintain key/value pairs of data between scheduled runs of your script. Normally, data does not persist from one invocation to the next.

**Specify paths**

The input sets the **SplunkHome** variable so you can easily address scripts in add-ons by writing paths like this:

```
[powershell://MSExchange_Health]
script=

$SplunkHome/etc/apps/TA-Exchange-2010/powershell/health.ps1
```

Besides **$SplunkHome**, there are several other read-only constant variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SplunkServerName</strong></td>
<td>The name configured for this machine to use in events</td>
</tr>
<tr>
<td><strong>SplunkServerUri</strong></td>
<td>The Splunk Enterprise REST API address.</td>
</tr>
<tr>
<td><strong>SplunkSessionKey</strong></td>
<td>The session key (authentication token) needed for accessing the Splunk Enterprise REST API.</td>
</tr>
<tr>
<td><strong>SplunkCheckpointPath</strong></td>
<td>The path for storing persistent state</td>
</tr>
<tr>
<td><strong>SplunkServerHost</strong></td>
<td>The name of the Splunk Enterprise instance that you want to communicate with.</td>
</tr>
<tr>
<td><strong>SplunkStanzaName</strong></td>
<td>The name of the inputs.conf stanza that defined this script.</td>
</tr>
</tbody>
</table>
Handle output of PowerShell scripts

Splunk Enterprise takes each object that your script produces as an output and turns it into an event, wrapped in `<event>` and tags. Splunk Enterprise converts the properties of each object into key/value pairs. However, the value can only be a quoted string, converted by calling the `.ToString()` method. Thus, the output must be simple, and you should flatten any complex nested objects in your script before the script outputs them.

There are a few special property names which have significance for Splunk Enterprise modular inputs and let you override the defaults in the inputs.conf stanza. They are:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SplunkIndex</td>
<td>Overrides the index that the output will be stored in.</td>
</tr>
<tr>
<td>SplunkSource</td>
<td>Overrides the &quot;source&quot; for the output.</td>
</tr>
<tr>
<td>SplunkHost</td>
<td>Overrides the &quot;host&quot; name for the output.</td>
</tr>
<tr>
<td>SplunkSourceType</td>
<td>Overrides the &quot;sourcetype&quot; for the output.</td>
</tr>
<tr>
<td>SplunkTime</td>
<td>Overrides the &quot;time&quot;. If you do not specify this, all objects that your script generates in a single execution will get roughly the same timestamp. This is because the script holds the objects for output until it has finished executing, and then marks the objects with the output time. You must specify this value in epoch or POSIX time, which is a positive integer that represents the seconds that have elapsed since 0:00 UTC on Thursday, January 1, 1970.</td>
</tr>
</tbody>
</table>

These properties never appear as objects in the key/value output.

If you want to set these properties and override the defaults, use a calculated expression with the `Select-Object` cmdlet or use the `Add-Member` cmdlet to add a `NoteProperty` property.

Caveats for handling PowerShell script output

The input currently requires that any PowerShell scripts it executes produce output objects that do not have any script properties. Pipe output through the `Select-Object` cmdlet to ensure proper formatting.

The input currently does not process the output of scripts until your pipeline and runspace are finished. This means the input does not process `ScriptProperty`
values. It also means that all of your output essentially has the same timestamp, unless you override it using the SplunkTime variable.

When writing your scripts, avoid long-running scripts. Do not write scripts that wait for things to happen unless the scripts exit every time there is output.

**Monitor Windows host information**

Splunk Enterprise supports the monitoring of detailed statistics about the local Windows machine. It can collect the following information about the Windows host:

- **General computer.** The make and model of the computer, its host name and the Active Directory domain it is in.
- **Operating system.** The version and build number of the operating system installed on the computer, as well as any service packs; the computer name; the last time it was started, the amount of installed and free memory, and the system drive.
- **Processor.** The make and model of the CPU(s) installed in the system, their speed and version, the number of processor(s) and core(s), and the processor ID.
- **Disk.** A listing of all drives available to the system and, if available, their file system type and total and available space.
- **Network Adapter.** Information about the installed network adapters in the system, including manufacturer, product name and MAC address.
- **Service.** Information about the installed services on the system, including name, display name, description, path, service type, start mode, state, and status.
- **Process.** Information on the running processes on the system, including the name, the command line (with arguments), when they were started, and the executable's path.

Both full instances of Splunk Enterprise and universal forwarders support local collection of host information. If you have Splunk Cloud and want to monitor host information, use the universal forwarder to collect the data and forward it to your Splunk Cloud deployment.

The host monitor input runs as a process called `splunk-winhostmon.exe`. This process runs once for every input defined, at the interval specified in the input. You can configure host monitoring using Splunk Web or `inputs.conf`. 
Why monitor host information?

Windows host monitoring gives you detailed information about your Windows hosts. You can monitor changes to the system, such as installation and removal of software, the starting and stopping of services, and uptime. When a system failure occurs, you can use Windows host monitoring information as a first step into the forensic process. With the Splunk Enterprise search language, you can give your team at-a-glance statistics on all machines in your Windows network.

What's required to monitor host information?

<table>
<thead>
<tr>
<th>Activity</th>
<th>Required permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor host information</td>
<td>* Splunk Enterprise must run on Windows.</td>
</tr>
<tr>
<td></td>
<td>* Splunk Enterprise must run as the Local System user or a local administrator account to read all local host information.</td>
</tr>
</tbody>
</table>

Security and remote access considerations

Splunk Enterprise must run as the Local System user to collect Windows host information by default.

Splunk recommends using a universal forwarder to send host information from remote machines to an indexer. Review the Universal Forwarder Manual for information about how to install, configure and use the forwarder to collect Windows host data.

If you choose to install forwarders on your remote machines to collect Windows host data, then you can install the forwarder as the Local System user on these machines. The Local System user has access to all data on the local machine, but not on remote machines.

If you run Splunk Enterprise as a user other than the "Local System" user, then that user must have local Administrator rights on the machine that you want to collect host data. It must also have other permissions, as detailed in Choose the Windows user Splunk Enterprise should run as in the Installation manual.

Use Splunk Web to configure host monitoring

Go to the Add New page

You can get there by two routes:
• Splunk Home
• Splunk Settings

By Splunk Settings:

1. Click **Settings** in the upper right corner of Splunk Web.

2. Click **Data Inputs**.

3. Click **Files & Directories**.

4. Click **New** to add an input.

By Splunk Home:

1. Click the **Add Data** link in Splunk Home.

2. Click **Monitor** to monitor host information from the local Windows machine.

**Select the input source**

1. In the left pane, locate and select **Local Windows host monitoring**.

2. In the **Collection Name** field, enter a unique name for this input that you will remember.

3. In the **Event Types** list box, locate the host monitoring event types you want this input to monitor.

4. Click once on each type you want to monitor. Splunk Enterprise moves the type from the "Available type(s)" window to the "Selected type(s)" window.

5. To unselect a type, click on its name in the "Selected type(s)" window. Splunk Enterprise moves the counter from the "Selected type(s)" window to the "Available type(s)" window.

6. (Optional) To select or unselect all of the types, click on the "add all" or "remove all" links. **Note:** Selecting all of the types can result in the indexing of a lot of data, possibly more than your license allows.

7. In the **Interval** field, enter the time, in seconds, between polling attempts for the input.
8. Click Next.

**Specify input settings**

The Input Settings page lets you specify application context, default host value, and index. All of these parameters are optional.

1. Select the appropriate Application context for this input.

2. Set the Host name value. You have several choices for this setting. Learn more about setting the host value in About hosts.

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

3. Set the Index that Splunk Enterprise should send data to. 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 Enterprise has a number of utility indexes, which also appear in this dropdown box.

4. Click Review.

**Review your choices**

After specifying all your input settings, review your selections. Splunk Enterprise lists all options you selected, including the type of monitor, the source, the source type, the application context, and the index.

1. Review the settings.

2. If they do not match what you want, click < to go back to the previous step in the wizard. Otherwise, click Submit.

Splunk Enterprise then loads the "Success" page and begins indexing the specified host information.

**Use inputs.conf to configure host monitoring**

You can edit inputs.conf to configure host monitoring. For more information on how to edit configuration files, see About configuration files in the Admin manual.

1. Create an inputs.conf in %SPLUNK_HOME%/etc/system/local and open it for editing.
2. Open `%SPLUNK_HOME%/etc/system/default/inputs.conf` and review it for the Windows event log inputs you want to enable.

3. Copy the Windows event log input stanzas you want to enable from `%SPLUNK_HOME%/etc/system/default/inputs.conf`.

4. Paste the stanzas you copied into `%SPLUNK_HOME%/etc/system/local/inputs.conf`.

5. Make edits to the stanzas to collect the Windows event log data you desire.

6. Save `%SPLUNK_HOME%/etc/system/local/inputs.conf` and close it.

7. Restart Splunk Enterprise.

Windows host monitor configuration values

Splunk Enterprise uses the following attributes in `inputs.conf` to monitor Windows host information.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>interval</td>
<td>Yes</td>
<td>How often, in seconds, to poll for new data. If you set the interval to a negative number, Splunk Enterprise runs the input one time. If you do not define this attribute, the input does not run, as there is no default.</td>
</tr>
<tr>
<td>type</td>
<td>Yes</td>
<td>The type of host information to monitor. Can be one of Computer, operatingSystem, processor, disk, networkAdapter, service, process, or driver. The input does not run if this attribute is not present.</td>
</tr>
<tr>
<td>disabled</td>
<td>No</td>
<td>Whether or not to run the input. If you set this attribute to 1, then Splunk Enterprise does not run the input.</td>
</tr>
</tbody>
</table>

Examples of Windows host monitoring configurations

Following are some examples of how to use the Windows host monitoring configuration attributes in `inputs.conf`.

```
# Queries computer information.
[WinHostMon://computer]
type = Computer
```
interval = 300

# Queries OS information.
# 'interval' set to a negative number tells Splunk Enterprise to
# run the input once only.
[WinHostMon://os]
type = operatingSystem
interval = -1

# Queries processor information.
[WinHostMon://processor]
type = processor
interval = -1

# Queries hard disk information.
[WinHostMon://disk]
type = disk
interval = -1

# Queries network adapter information.
[WinHostMon://network]
type = networkAdapter
interval = -1

# Queries service information.
# This example runs the input every 5 minutes.
[WinHostMon://service]
type = service
interval = 300

# Queries information on running processes.
# This example runs the input every 5 minutes.
[WinHostMon://process]
type = process
interval = 300

**Fields for Windows host monitoring data**

When Splunk Enterprise indexes data from Windows host monitoring inputs, it sets the **source** for received events to **windows**. It sets the **source type** of the incoming events to **WinHostMon**.

**Answers**

Have questions? Visit Splunk Answers and see what questions and answers the Splunk community has around Windows host information.
Monitor Windows printer information

Splunk Enterprise supports the monitoring of statistics about all of the printers and drivers, print jobs, and printer ports on the local Windows host. It can collect the following print system information:

- **Printer.** Information on the print subsystem, such as the status of installed printers, and when printers get added or deleted.
- **Job.** Information on print jobs, including who has printed what, details on the jobs, and the status of existing jobs.
- **Driver.** Information on the print driver subsystem, including information on existing print drivers, and when a print driver gets added or removed.
- **Port.** Information on printer ports installed on the system, and when they get added or removed.

Both full instances of Splunk Enterprise and universal forwarders support local collection of printer subsystem information. If you have Splunk Cloud and want to monitor printer subsystem information, use the universal forwarder to consume the information and forward it to your Splunk Cloud deployment.

The printer monitor input runs as a process called `splunk-winprintmon.exe`. This process runs once for every input you define, at the interval specified in the input. You can configure printer subsystem monitoring using Splunk Web or `inputs.conf`.

Why monitor printer information?

Windows printer monitoring gives you detailed information about your Windows printer subsystem. You can monitor any changes to the system, such as installation and removal of printers, print drivers, and ports, the starting and completion of print jobs, and learn who printed what when. When a printer failure occurs, you can use print monitoring information as a first step into the forensic process. With the Splunk Enterprise search language, you can give your team at-a-glance statistics on all printers in your Windows network.

What's required to monitor printer information?

<table>
<thead>
<tr>
<th>Activity</th>
<th>Required permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor host information</td>
<td>* Splunk Enterprise must run on Windows.</td>
</tr>
<tr>
<td></td>
<td>* Splunk Enterprise must run as the Local System user to</td>
</tr>
<tr>
<td></td>
<td>read all local host information.</td>
</tr>
<tr>
<td>Activity</td>
<td>Required permissions</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Security and remote access considerations</td>
<td></td>
</tr>
</tbody>
</table>

Splunk Enterprise must run as the Local System user to collect Windows print subsystem information by default.

Use a universal forwarder to send printer information from remote machines to an indexer. If you choose to install forwarders on your remote machines to collect printer subsystem data, then you can install the forwarder as the Local System user on these machines. The Local System user has access to all data on the local machine, but not on remote machines.

If you run Splunk Enterprise as a user other than the "Local System" user, then that user must have local Administrator rights to the machine, and other permissions as detailed in Choose the Windows user Splunk Enterprise should run as in the Installation manual.

**Use Splunk Web to configure printer information**

*Go to the Add New page*

You can get there by two routes:

- Splunk Home
- Splunk Settings

**By Splunk Settings:**

1. Click **Settings** in the upper right corner of Splunk Web.
2. Click **Data Inputs**.
3. Click **Local Windows print monitoring**.
4. Click **New** to add an input.

**By Splunk Home:**

1. Click the **Add Data** link in Splunk Home.
2. Click **Monitor** to monitor print information from the local Windows machine.
3. In the left pane, locate and select **Local Windows print monitoring**.

**Select the input source**

1. In the **Collection Name** field, enter a unique name for this input that you will remember.

2. In the **Event Types** list box, locate the print monitoring event types you want this input to monitor.

3. Click once on each type you want to monitor. Splunk Enterprise moves the type from the "Available type(s)" window to the "Selected type(s)" window.

4. To unselect a type, click on its name in the "Selected type(s)" window. Splunk Enterprise moves the counter from the "Selected type(s)" window to the "Available type(s)" window.

5. (Optional) To select or unselect all of the types, click on the "add all" or "remove all" links. **Important:** Selecting all of the types can result in the indexing of a lot of data, possibly more than your license allows.

6. In the **Baseline** control, click the **Yes** radio button to run the input as soon as it starts, and no further. Click **No** to run the input at the interval specified in the **Interval (in minutes)** field.

7. Click the green **Next** button.

**Specify input settings**

The **Input Settings** page lets you specify application context, default host value, and index. All of these parameters are optional.

1. Select the appropriate **Application context** for this input.

2. Set the **Host** name value. You have several choices for this setting. Learn more about setting the host value in **About hosts**.

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

3. Set the **Index** that Splunk Enterprise should send data to. 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 Enterprise has a number of
utility indexes, which also appear in this dropdown box.

4. Click Review.

**Review your choices**

After specifying all your input settings, review your selections. Splunk Enterprise lists all options you selected, including the type of monitor, the source, the source type, the application context, and the index.

1. Review the settings.

2. If they do not match what you want, click < to go back to the previous step in the wizard. Otherwise, click Submit.

Splunk Enterprise then loads the "Success" page and begins indexing the specified print information.

**Use inputs.conf to configure host monitoring**

You can edit inputs.conf to configure host monitoring. For information on how to edit configuration files, see About configuration files in the Admin manual.

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 print monitoring inputs.

4. Restart Splunk.

**Print monitoring configuration values**

Splunk Enterprise uses the following attributes in inputs.conf to monitor Windows printer subsystem information:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Yes</td>
<td>The type of host information to monitor. Can be one of printer, job, driver, or port. The input will not run if this variable is not present.</td>
</tr>
<tr>
<td>baseline</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
Whether or not to generate a baseline of the existing state of the printer, job, driver, or port. If you set this attribute to 1, then Splunk Enterprise writes a baseline. This might take additional time and CPU resources when Splunk Enterprise starts.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disabled</td>
<td>No</td>
<td>Whether or not to run the input. If you set this attribute to 1, then Splunk Enterprise does not run the input.</td>
</tr>
</tbody>
</table>

**Examples of Windows host monitoring configurations**

Following are some examples of how to use the Windows host monitoring configuration attributes in inputs.conf.

```plaintext
# Monitor printers on system.
[WinPrintMon://printer]
type = printer
baseline = 0

# Monitor print jobs.
[WinPrintMon://job]
type = job
baseline = 1

# Monitor printer driver installation and removal.
[WinPrintMon://driver]
type = driver
baseline = 1

# Monitor printer ports.
[WinPrintMon://port]
type = port
baseline = 1
```

**Fields for Windows print monitoring data**

When Splunk Enterprise indexes data from Windows print monitoring inputs, it sets the source for received events to windows. It sets the source type of the incoming events to WinPrintMon.

**Answers**

Have questions? Visit Splunk Answers and see what questions and answers the Splunk community has around Windows print monitoring.
Monitor Windows network information

Splunk Enterprise supports the monitoring of detailed statistics about network activity into or out of a Windows host. It can collect the following network information:

- **Network activity.** When a Windows machine performs any kind of network action, Splunk Enterprise can monitor it.
- **Address family.** Whether or not the network transaction was made over the IPv4 or IPv6 protocols.
- **Packet type.** The type of packet sent in the transaction (for example, a 'connect' or 'transport' packet).
- **Protocol.** Whether or not the network transaction was made over the TCP or UDP protocols.
- **Hosts.** Information about the hosts involved in the network transaction, including the local and remote hosts, the ports which the hosts used to communicate, and any available DNS information.
- **Application.** Which application initiated the network transaction.
- **User.** The user that initiated the network transaction, including his or her ID and SID.
- **Miscellany.** Miscellaneous information about the network transaction, including the transport header size and whether or not the transaction was protected by IPSec.

Both full instances of Splunk Enterprise and universal forwarders support local collection of network information. If you have Splunk Cloud and want to monitor network information, use the universal forwarder to collect the data and forward it to your Splunk Cloud deployment.

The network monitor input runs as a process called splunk-netmon.exe. This process runs once for every input defined, at the interval specified in the input. You can configure network monitoring using Splunk Web or inputs.conf.

Windows network monitoring in Splunk Enterprise is only available on 64-bit Windows systems. It does not function on 32-bit Windows systems.

**Why monitor network information?**

Windows network monitoring gives you detailed information about your Windows network activity. You can monitor all transactions on the network, such as the initiation of a network connection by a user or process or whether or not the transaction uses the IPv4 or IPv6 address families. The network monitoring
facilities in Splunk Enterprise can help you detect and interrupt an incoming (or outgoing) denial of service attack by telling you the involved machines. With Splunk Enterprise search language, you can give your team at-a-glance statistics on all Windows network operations.

**What's required to monitor network information?**

<table>
<thead>
<tr>
<th>Activity</th>
<th>Requirements</th>
</tr>
</thead>
</table>
| Monitor network information | • Splunk must run on Windows.  
• The Windows version on the machine must be one of:  
  ♦ Windows Vista.  
  ♦ Windows 7.  
  ♦ Windows 8.  
  ♦ Windows 8.1.  
  ♦ Windows 10.  
  ♦ Windows Server 2008 R2.  
  ♦ Windows Server 2012 R2.  
• The Windows system must have all available updates and service packs applied, including the Kernel-Mode Driver Framework version 1.11 update on machines that run Windows Vista, Windows 7, Windows Server 2008, and Windows Server 2008 R2.  
• Splunk must run as the Local System user or a local administrator account to read all local host information. |

**Security and remote access considerations**

Splunk Enterprise must run as the Local System user to collect Windows network information by default.

Use a universal forwarder to send host information from remote machines to an indexer when possible. If you choose to install forwarders on your remote machines to collect Windows network information, then install the forwarder as the Local System user on these machines. The Local System user has access to all data on the local machine, but not on remote machines.

If you run Splunk Enterprise as a user other than the "Local System" user, then that user must have local Administrator rights to the machine and other explicit permissions, as detailed in Choose the Windows user Splunk Enterprise should run as in the *Installation* manual.
Use Splunk Web to configure host monitoring

Go to the Add New page

You can get there by two routes:

- Splunk Home
- Splunk Settings

By Splunk Settings:

1. Click Settings in the upper right corner of Splunk Web.
2. Click Data Inputs.
3. Click Local Windows network monitoring.
4. Click New to add an input.

By Splunk Home:

1. Click the Add Data link in Splunk Home.
2. Click Monitor to monitor network information from the local Windows machine, or Forward to forward network information from another Windows machine. Splunk Web displays the "Add Data - Select Source" page.

Note: Forwarding network information requires additional setup.

3. In the left pane, locate and select Local Windows network monitoring.

Select the input source

1. In the Network Monitor Name field, enter a unique name for this input that you will remember.
2. Under Address family, check the IP address family types that you want Splunk Enterprise to monitor (either IPv4 or IPv6.)
3. Under Packet Type, check the packet types you want the input to monitor (any of connect, accept, or transport.)
4. Under **Direction**, check the network directions that you want the input to monitor (any of **inbound** (toward the monitoring host) or **outbound** (away from the monitoring host).

5. Under **Protocol**, check the network protocol types that you want the input to monitor (any of **tcp** (Transmission Control Protocol) or **udp** (User Datagram Protocol)).

6. In the **Remote address** text field, enter the host name or IP address of a remote host whose network communications with the monitoring host that you want the input to monitor.

**Note:** If you want to monitor multiple hosts, enter a regular expression in this field.

7. In the **Process** text field, enter the partial or full name of a process whose network communications you want the input to monitor.

**Note:** As with the remote address, you can monitor multiple processes by entering a regular expression.

8. In the **User** text field, enter the partial or full name of a user whose network communications you want the input to monitor.

**Note:** As with the remote address and process entries, you can monitor multiple users by entering a regular expression in this field.

9. Click **Next**.

**Specify input settings**

The **Input Settings** page lets you specify application context, default host value, and index. All of these parameters are optional.

1. Select the appropriate **Application context** for this input.

2. Set the **Host** name value. You have several choices for this setting. Learn more about setting the host value in **About hosts**.

**Note:** Host only sets the **host** field in the resulting events. It does not direct Splunk Enterprise to look on a specific host on your network.
3. Set the **Index** that Splunk Enterprise should send data to. 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 Enterprise has a number of utility indexes, which also appear in this dropdown box.

4. Click **Review**.

**Review your choices**

After specifying all your input settings, review your selections. Splunk Enterprise lists all options you selected, including the type of monitor, the source, the source type, the application context, and the index.

1. Review the settings.

2. If they do not match what you want, click < to go back to the previous step in the wizard. Otherwise, click the green **Submit** button.

Splunk Enterprise then loads the "Success" page and begins indexing the specified print information.

**Use inputs.conf to configure network monitoring**

You can edit `inputs.conf` to configure network monitoring. For information on how to edit configuration files, see About configuration files in the Admin manual.

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 network monitoring inputs.

4. Restart Splunk.

The next section describes the specific configuration values for host monitoring.

**Windows host monitor configuration values**

To define a Windows network monitoring input, use the `[WinNetMon://<name>]` stanza in `inputs.conf`. Splunk Enterprise uses the following attributes to configure the Windows network monitor input.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>disabled = [0</td>
<td>1]</td>
<td>Whether or not the input should run. Set to 1 to disable the input, and 0 to enable it.</td>
</tr>
<tr>
<td>index = &lt;string&gt;</td>
<td>The index that this input should send the data to. This attribute is optional.</td>
<td>The default index</td>
</tr>
<tr>
<td>remoteAddress = &lt;regular expression&gt;</td>
<td>Matches against the remote IP address involved in the network transaction. Accepts regular expressions that represent IP addresses only, not host names. Filters out events with remote addresses that do not match the regular expression. Passes through events with remote addresses that match the regular expression. For example: 192.163..* matches all IP addresses in the 192.163.x.x range.</td>
<td>(empty string - matches everything)</td>
</tr>
<tr>
<td>process = &lt;regular expression&gt;</td>
<td>Matches against the process or application name which performed the network access. Filters out events generated by processes that do not match the regular expression. Passes through events generated by processes that match the regular expression.</td>
<td>(empty string - matches all processes or applications)</td>
</tr>
<tr>
<td>user = &lt;regular expression&gt;</td>
<td>Matches against the user name which performed network access. Filters out events generated by users that do not match the regular expression. Passes through events generated by users that match the regular expression.</td>
<td>(empty string - includes access by all users)</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>addressFamily = [ipv4;ipv6]</td>
<td>If set, matches against the address family used in the network access. Accepts semicolon-separated values, for example &quot;ipv4;ipv6&quot;.</td>
<td>(empty string - includes all IP traffic.)</td>
</tr>
<tr>
<td>packetType = [connect;accept;transport]</td>
<td>Matches against the packet type used in the transaction. Accepts semicolon-separated values, for example &quot;connect;transport&quot;.</td>
<td>(empty string - includes all packet types.)</td>
</tr>
</tbody>
</table>
| direction = [inbound;outbound] | • If set, matches against the general direction of the network traffic.  
• "Inbound" means traffic coming into the monitoring machine, "outbound" means traffic leaving the monitoring machine.  
• Accepts semicolon-separated values, for example "inbound;outbound". | (empty string - includes both directions.)                                                     |
| protocol = [tcp;udp]   | Matches against the specified network protocol.  
"tcp" means Transmission Control Protocol, where networks use handshakes to and state to set up transactions.  
"udp" means User Datagram Protocol, a stateless, "fire and forget" protocol.  
Accepts semicolon-separated values, for example "tcp;udp". | (empty string - includes both protocol types.)                                                |
<p>| readInterval = &lt;integer&gt; | <strong>Advanced option.</strong> Use the default value unless there is a problem with input                                                                 | 100                                                                                           |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>driverBufferSize =</td>
<td><strong>Advanced option.</strong> Use the default value unless there is a problem with input performance.</td>
<td>1024</td>
</tr>
<tr>
<td>&lt;integer&gt;</td>
<td>The number of network packets it should keep in the network monitor filter driver buffer. Controls the amount of packets that the driver caches. Lower values might result in event loss, while higher values might increase the size of non-paged memory. The minimum legal value is 128 and the maximum legal value is 8192.</td>
<td></td>
</tr>
<tr>
<td>mode = &lt;string&gt;</td>
<td>How to output each event. Splunk Enterprise can output each event in either single or multikv (key-value pair) mode.</td>
<td>single</td>
</tr>
<tr>
<td>multikvMaxEventCount =</td>
<td><strong>Advanced option.</strong> Use the default value unless there is a problem with input performance.</td>
<td>100</td>
</tr>
<tr>
<td>&lt;integer&gt;</td>
<td>The maximum amount of events to output when you set</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Fields for Windows network monitoring data

When Splunk Enterprise indexes data from Windows network monitoring inputs, it sets the source for received events to windows. It sets the source type of the incoming events to WinNetMon.

### Confirm that your Windows machine is fully patched

If you encounter issues while running the network monitoring input on a Windows Vista, Windows 7, Windows Server 2008, or Windows Server 2008 R2 machine, confirm that you have updated the machine with all available patches, including the Kernel-Mode Driver Framework version 1.11 Update (http://support.microsoft.com/kb/2685811) that is part of Knowledge Base article 2685811. Network monitoring input might not function if this update is not present on your system.

### Answers

Have questions? Visit Splunk Answers and see what questions and answers the Splunk community has around Windows network monitoring.
Get data with HTTP Event Collector

Set up and use HTTP Event Collector in Splunk Web

The HTTP Event Collector (HEC) lets you send data and application events to a Splunk deployment over the HTTP and Secure HTTP (HTTPS) protocols. HEC uses a token-based authentication model. You can generate a token and then configure a logging library or HTTP client with the token to send data to HEC in a specific format. This process eliminates the need for a Splunk forwarder when you send application events.

After you enable HEC, as a developer, you can use HEC tokens in your app to send data to HEC. You do not need to include Splunk credentials in your app or supported files.

HEC functionality varies based on Splunk software type

HTTP Event Collector runs on Splunk Enterprise, and self-service and managed Splunk Cloud. How it works depends on the type of Splunk instance you have.

HEC and Splunk Enterprise

HEC offers full configurability and functionality on the Splunk Enterprise platform on-premises. It offers the following additional benefits on Splunk Enterprise over the other Splunk types:

- HEC can accept events that you send to it over the HTTP protocol in addition to the HTTPS protocol.
- HEC can forward events to another Splunk indexer with an optional forwarding output group.
- You can use the deployment server to distribute HEC tokens across indexers in a distributed deployment.

For instructions on how to enable and manage HEC on Splunk Enterprise, see Configure HTTP Event Collector on Splunk Enterprise.

HEC and self-service Splunk Cloud

HEC offers similar functionality on self-service Splunk Cloud instances as it does on Splunk Enterprise. The following exceptions apply:
• You cannot make changes to configuration files, because Splunk Cloud does not provide that access.
• You cannot change the network port that HEC listens on for connections.
• You cannot forward data that HEC receives to another set of Splunk indexers as Splunk Cloud does not support forwarding output groups.

For instructions on how to enable and manage HEC on self-service Splunk Cloud, see Configure HTTP Event Collector on self-service Splunk Cloud.

**HEC and managed Splunk Cloud**

HEC offers an experience on Splunk Cloud deployments that Splunk manages that is similar to the experience on self-service Splunk Cloud. The following exceptions apply:

• You cannot make changes to configuration files, because Splunk Cloud does not provide that access.
• You must file a ticket with Splunk Support to enable HEC for use with Kinesis Firehose. Standard HEC is enabled by default on all Splunk Cloud stacks and does not require a Splunk Support ticket.
• You cannot make changes to global settings. You can only make settings changes to tokens that you create.
• You cannot forward data that HEC receives to another set of Splunk indexers as Splunk Cloud does not support forwarding output groups.
• The index that you choose to store events that HEC receives must already exist. You cannot create a new index during the setup process.
• Indexer acknowledgment is only available for Amazon Kinesis Firehose at this time.
• After you create tokens, you can monitor progress of the token as it is deployed across your managed Splunk Cloud instance.

For instructions on how to enable and manage HEC on managed Splunk Cloud, see Configure HTTP Event Collector on managed Splunk Cloud.

**About Event Collector tokens**

Tokens are entities that let logging agents and HTTP clients connect to the HEC input. Each token has a unique value, which is a 128-bit number that is represented as a 32-character globally unique identifier (GUID). Agents and clients use a token to authenticate their connections to HEC. When the clients connect, they present this token value. If HEC receives a valid token, it accepts the connection and the client can deliver its payload of application events in either text or JavaScript Object Notation (JSON) format.
HEC receives the events and Splunk Enterprise indexes them based on the configuration of the token. HEC uses the source, source type, and index that was specified in the token. If a forwarding output group configuration exists on a Splunk Enterprise instance, HEC forwards the data to indexers in that output group.

**Use output groups to specify groups of indexers**

To index large amounts of data, you will likely need multiple indexers. You can specify groups of indexers to handle indexing your HTTP Event Collector data. These are called *output groups*. You can use output groups to, for example, index only certain kinds of data or data from certain sources. Though using output groups to route data to specific indexers is similar to the routing and filtering capabilities built into Splunk Enterprise, output groups allow you to specify groups of indexers on a token-by-token basis. When you configure output groups with multiple indexers, Splunk Enterprise evenly distributes data among the servers in your output group. You configure output groups in the `outputs.conf` file. Specifically, for HTTP Event Collector, edit the `outputs.conf` file at `$SPLUNK_HOME/etc/apps/splunk_httpinput/local/` ( `%SPLUNK_HOME%\etc\apps\splunk_httpinput\local\` on Microsoft Windows hosts). If either the `local` directory or the `outputs.conf` file doesn't exist at this location, create it (or both).

HTTP Event Collector is not an app, but it stores its configuration in the `$SPLUNK_HOME/etc/apps/splunk_httpinput/` directory ( `%SPLUNK_HOME%\etc\apps\splunk_httpinput\` on Windows) so that its configuration can be easily deployed using built-in app deployment capabilities.

**Configure HTTP Event Collector on Splunk Enterprise**

**Enable HTTP Event Collector**

Before you can use Event Collector to receive events through HTTP, you must enable it. For Splunk Enterprise, enable HEC through the *Global Settings* dialog box.

1. Click *Settings > Data Inputs*.
2. Click *HTTP Event Collector*.
3. Click *Global Settings*.
4. In the **All Tokens** toggle button, select **Enabled**.
5. (Optional) Choose a **Default Source Type** for all HEC tokens. You can also type in the name of the source type in the text field above the drop-down before choosing the source type.
6. (Optional) Choose a **Default Index** for all HEC tokens.
7. (Optional) Choose a **Default Output Group** for all HEC tokens.
8. (Optional) To use a deployment server to handle configurations for HEC tokens, click the **Use Deployment Server** check box.
9. (Optional) To have HEC listen and communicate over HTTPS rather than HTTP, click the **Enable SSL** checkbox.
10. (Optional) Enter a number in the **HTTP Port Number** field for HEC to listen on. **Note:** Confirm that no firewall blocks the port number that you specified in the **HTTP Port Number** field, either on the clients or the Splunk instance that hosts HEC.
11. Click **Save**.

**Create an Event Collector token**

To use HEC, you must configure at least one token.

1. Click **Settings > Add Data**.
2. Click **monitor**.
3. Click **HTTP Event Collector**.
4. In the **Name** field, enter a name for the token.
5. (Optional) In the **Source name override** field, enter a source name for events that this input generates.
6. (Optional) In the **Description** field, enter a description for the input.
7. (Optional) In the **Output Group** field, select an existing forwarder output group.
8. (Optional) If you want to enable indexer acknowledgment for this token, click the **Enable indexer acknowledgment** checkbox.
9. Click **Next**.
10. (Optional) Confirm the source type and the index for HEC events.
11. Click **Review**.
12. Confirm that all settings for the endpoint are what you want.
13. If all settings are what you want, click **Submit**. Otherwise, click < to make changes.
14. (Optional) Copy the token value that Splunk Web displays and paste it into another document for reference later.

For information about HEC tokens, see [About Event Collector tokens](#).

For information about defining forwarding output groups, see [Configure forwarders with outputs.conf](#). You can also set up forwarding in Splunk Web, which generates a default output group called `default-autolb-group`.

For information on indexer acknowledgement, see [HTTP Event Collector indexer acknowledgement](#). Indexer acknowledgement in HTTP Event Collector is not the same indexer acknowledgement capability described in indexer acknowledgement and indexer clusters.

**Modify an Event Collector token**

![Modify an Event Collector token image]
You can make changes to an HEC token after you have created it.

1. Click **Settings > Data Inputs**.
2. Click **HTTP Event Collector**.
3. Locate the token that you want to change in the list.
4. In the **Actions** column for that token, click **Edit**. You can also click the link to the token name.
5. (Optional) Edit the description of the token by entering updated text in the **Description** field.
6. (Optional) Update the source value of the token by entering text in the **Source** field.
7. (Optional) Choose a different source type by selecting it in the **Source Type** drop-down.
   1. Choose a category.
   2. Select a source type in the pop-up menu that appears.
   3. (Optional) You can also type in the name of the source type in the text box at the top of the drop-down.
8. (Optional) Choose a different index by selecting it in the Available Indexes pane of the Select Allowed Indexes control.
9. (Optional) Choose a different output group from the Output Group drop-down.
10. (Optional) Choose whether you want indexer acknowledgment enabled for the token.
11. Click Save.

**Delete an Event Collector token**

You can delete an HEC token. Deleting an HEC token does not affect other HEC tokens, nor does it disable HEC.

You cannot undo this action. Clients that use this token to send data to your Splunk deployment can no longer authenticate with the token. You must generate a new token and change the client configuration to use the new token.

1. Click Settings > Data Inputs.
2. Click HTTP Event Collector.
3. Locate the token that you want to delete in the list.
4. In the Actions column for that token, click Delete.
5. In the Delete Token dialog box, click Delete.

**Enable and disable Event Collector tokens**

You can enable or disable a single HEC token from within the HEC management page. Changing the status of one token does not change the status of other tokens. To enable or disable all tokens, use the Global Settings dialog. See Enable the HTTP Event Collector.

To toggle the active status of an HEC token:

1. Click Settings > Data Inputs.
2. Click HTTP Event Collector.
3. In the Actions column for that token, click the Enable link, if the token is not active, or the Disable link, if the token is active. The token status toggles immediately and the link changes to Enable or Disable based on the changed token status.

**Configure HTTP Event Collector on self-service Splunk Cloud**
Enable HTTP Event Collector

1. Click Settings > Data Inputs.
2. Click HTTP Event Collector.
3. Click Global Settings.

4. In the All Tokens toggle button, select Enabled.
5. (Optional) Choose a Default Source Type for all HEC tokens. You can also type in the name of the source type in the text field above the drop-down before choosing the source type.
6. (Optional) Choose a Default Index for all HEC tokens.
7. Click Save.

Create an Event Collector token

To use HEC, you must configure at least one token.

1. Click Settings > Add Data.
2. Click monitor.
3. Click HTTP Event Collector.
4. In the Name field, enter a name for the token.
5. (Optional) In the Source name override field, enter a name for a source to be assigned to events that this endpoint generates.
6. (Optional) In the Description field, enter a description for the input.
7. (Optional) If you want to enable indexer acknowledgment for this token, click the Enable indexer acknowledgment checkbox.
8. Click Next.
9. (Optional) Make edits to source type and confirm the index where you want HEC events to be stored. See Modify input settings.
10. Click Review.
11. Confirm that all settings for the endpoint are what you want.
12. If all settings are what you want, click **Submit**. Otherwise, click < to make changes.

13. (Optional) Copy the token value that Splunk Web displays and paste it into another document for reference later.

For information about HEC tokens, see About Event Collector tokens.

For information on indexer acknowledgement, see HTTP Event Collector indexer acknowledgment. Indexer acknowledgement in HTTP Event Collector is not the same indexer acknowledgement capability described in indexer acknowledgement and indexer clusters.

**Modify an Event Collector token**

![Edit Token: Test2](image)

You can make changes to an HEC token after you have created it.

1. Click **Settings > Data Inputs**.
2. Click **HTTP Event Collector**.
3. Locate the token that you want to change in the list.
4. In the **Actions** column for that token, click **Edit**. You can also click the link to the token name.
5. (Optional) Edit the description of the token by entering updated text in the **Description** field.
6. (Optional) Update the source value of the token by entering text in the **Source** field.
7. (Optional) Choose a different source type by selecting it in the **Source Type** drop-down.
1. Choose a category.
2. Select a source type in the pop-up menu that appears.
3. (Optional) You can also type in the name of the source type in the text box at the top of the drop-down.

8. (Optional) Choose a different index by selecting it in the Available Indexes pane of the Select Allowed Indexes control.
9. (Optional) Choose whether you want indexer acknowledgment enabled for the token.
10. Click **Save.**

**Delete an Event Collector token**

You can delete an HEC token. Deleting an HEC token does not affect other HEC tokens, nor does it disable the HEC endpoint.

You cannot undo this action. Clients that use this token to send data to your Splunk deployment can no longer authenticate with the token. You must generate a new token and change the client configuration to use the token.

1. Click **Settings > Data Inputs.**
2. Click **HTTP Event Collector.**
3. Locate the token that you want to delete in the list.
4. In the **Actions** column for that token, click **Delete.**
5. In the Delete Token dialog, click **Delete.**
Enable and disable Event Collector tokens

You can enable or disable an HEC token from within the HEC management page. Changing the status of one token does not change the status of other tokens. To enable or disable all tokens, use the Global Settings dialog. See Enable the HTTP Event Collector.

1. Click **Settings > Data Inputs**.
2. Click **HTTP Event Collector**.
3. In the **Actions** column for that token, click the **Enable** link, if the token is not active, or the **Disable** link, if the token is active. The token status toggles and the link changes to **Enable** or **Disable** based on the changed token status.

Configure HTTP Event Collector on managed Splunk Cloud

Enable HTTP Event Collector

Call Splunk Support and open a ticket to enable HEC.

Create an Event Collector token

To use HEC, you must configure at least one token. In managed Splunk Cloud instances, the token is distributed across the deployment. The token is not ready for use until distribution has completed.

1. Click **Settings > Add Data**.
2. Click **monitor**.
3. Click **HTTP Event Collector**.
4. In the **Name** field, enter a name for the token.
5. (Optional) In the **Source name override** field, enter a name for a source to be assigned to events that this endpoint generates.
6. (Optional) In the **Description** field, enter a description for the input.
7. Click **Next**.
8. (Optional) Make edits to source type and confirm the index where you want HEC events to be stored. See **Modify input settings**.
9. Click **Review**.
10. Confirm that all settings for the endpoint are what you want.
11. If all settings are what you want, click **Submit**. Otherwise, click < to make changes.
12. (Optional) Copy the token value that Splunk Web displays and paste it into another document for reference later.
13. (Optional) Click **Track deployment progress** to see progress on how the token has been deployed to the rest of the Splunk Cloud deployment. When you see a status of "Done", you can then use the token to send data to HEC.

For information about HEC tokens, see **About Event Collector tokens**.

For information on indexer acknowledgement, see **Enable indexer acknowledgement**. Indexer acknowledgement in HTTP Event Collector is not the same indexer acknowledgement capability in Splunk Enterprise.

**Check Event Collector token distribution status**

You can check the distribution status of an HEC token from the HEC token page. When a distribution is in progress, the page displays "Operation in progress" and a progress bar. Otherwise, the page displays "Last deployment status."

1. Click **Settings > Data Inputs**.
2. Click **HTTP Event Collector**.
3. Click **Operation in progress** or **Last deployment status**.
4. View the status of the token distribution.
5. Click **Close**.

**Modify an Event Collector token**

You can make changes to an HEC token after it has been created.

1. Click **Settings > Data Inputs**.
2. Click **HTTP Event Collector**.
3. Locate the token that you want to change in the list.
4. In the **Actions** column for that token, click **Edit**. You can also click the link to the token name.
5. (Optional) Edit the description of the token by entering updated text in the **Description** field.
6. (Optional) Update the source value of the token by entering text in the **Source** field.
7. (Optional) Choose a different source type by selecting it in the **Source Type** drop-down.
   1. Choose a category.
   2. Select a source type in the pop-up menu that appears.
   3. (Optional) You can also type in the name of the source type in the text box at the top of the drop-down.
8. (Optional) Choose a different index by selecting it in the **Available Indexes** pane of the **Select Allowed Indexes** control.
9. (Optional) Choose whether you want indexer acknowledgment enabled for the token.
10. Click **Save**.

**Delete an Event Collector token**

You can delete an HEC token. Deleting an HEC token does not affect other HEC tokens, nor does it disable the HEC endpoint.

You cannot undo this action. Clients that use this token to send data to your Splunk deployment can no longer authenticate with the token. You must generate a new token and change the client configuration to use the new value.

1. Click **Settings > Data Inputs**.
2. Click **HTTP Event Collector**.
3. Locate the token that you want to delete in the list.
4. In the **Actions** column for that token, click **Delete**.
5. In the Delete Token dialog, click **Delete**.

**Enable and disable Event Collector tokens**

You can enable or disable a token from within the HEC management page. Changing the active status of one token does not change the status of other tokens.

1. Click **Settings > Data Inputs**.
2. Click **HTTP Event Collector**.
3. In the **Actions** column for a token, click the **Enable** link, if the token is not active, or the **Disable** link, if the token is active. The token status toggles and the link changes to **Enable** or **Disable** based on the changed token status.
Send data to HTTP Event Collector

You must satisfy all of the following conditions when you send data to HEC:

- HEC must be enabled
- You must have at least one active HEC token available
- You must use an active token to authenticate into HEC
- You must format the data that goes to HEC in a certain way. See Format events for HTTP Event Collector

There are several options for sending data to HTTP Event Collector:

- You can make an HTTP request using your favorite HTTP client and send your JSON-encoded events.
- As a developer, you can use the Java, JavaScript (node.js), and .NET logging libraries in your application to send data to HEC. These libraries are compatible with popular logging frameworks. See Java, JavaScript (Node.js), and .NET on the Splunk Dev Portal.

Send data to HTTP Event Collector on Splunk Enterprise

You send data to a specific Uniform Resource Indicator (URI) for HEC.

The standard form for the HEC URI in Splunk Enterprise is as follows:

\[
\text{<protocol>://<host>:<port>/<endpoint>}
\]

Where:

- \text{<protocol>} is either \text{http} or \text{https}
- \text{<host>} is the Splunk instance that runs HEC
- \text{<port>} is the HEC port number, which is 8088 by default, but you can change in the HEC Global Settings
- \text{<endpoint>} is the HEC endpoint you want to use. In many cases, you use the \text{/services/collector} endpoint for JavaScript Object Notation (JSON)-formatted events or the \text{services/collector/raw} endpoint for raw events

Send data to HTTP Event Collector on Splunk Cloud instances

Depending on the type of Splunk Cloud that you use, you must send data using a specific URI for HEC.
The standard form for the HEC URI in self-service Splunk Cloud is as follows:

<protocol>://input-<host>:<port>/<endpoint>

The standard form for the HEC URI in managed Splunk Cloud is as follows:

<protocol>://http-inputs-<host>:<port>/<endpoint>

Where:

- <protocol> is either http or https
- <host> is the Splunk instance that runs HEC
- <port> is the HEC port number
  - 8088 on self-service Splunk Cloud instances
  - 443 on managed Splunk Cloud instances
- <endpoint> is the HEC endpoint you want to use. In many cases, you use the /services/collector endpoint for JavaScript Object Notation (JSON)-formatted events or the services/collector/raw endpoint for raw events
- For self-service Splunk Cloud plans, you must pre-pend the hostname with input-
- For managed Splunk Cloud plans, pre-pend the hostname with http-inputs-

If you do not include these prefixes before your Splunk Cloud hostname when you send data, the data cannot reach HEC.

**Example of sending data to HEC with an HTTP request**

The following example makes a HTTP POST request to the HEC on port 8088 and uses HTTPS for transport. This example uses the `curl` command to generate the request, but you can use a command line or other tool that better suits your needs.

You can configure the network port and HTTP protocol settings independently of settings for other instances of HEC in your Splunk Enterprise or self-service Splunk Cloud deployment.

The following cURL command uses an example HTTP Event Collector token (B5A79AAD-D822-46CC-80D1-819F80D7BFB0), and uses https://hec.example.com as the hostname. Replace these values with your own before running this command.
**JSON request and response**

When you make a JSON request to send data to HEC, you must specify the "event" key in the command.

```bash
curl -k https://hec.example.com:8088/services/collector/event -H "Authorization: Splunk B5A79AAD-D822-46CC-80D1-819F80D7BFB0" -d '{"event": "hello world")'
```

```json
{
"text": "Success", "code": 0
}
```

**More information on HEC for developers**

For developer content on using HTTP Event Collector, see HTTP Event Collector Examples, as well as Introduction to Splunk HTTP Event Collector in the Splunk Developer Portal.

**Set up and use HTTP Event Collector with configuration files**

HTTP Event Collector stores its settings on your Splunk Enterprise instance in two .conf files: inputs.conf and outputs.conf. These files are not accessible on Splunk Cloud instances, and configurations must be managed through Splunk Web.

Configuring HEC inputs with a configuration file is a slightly different process than configuring other data inputs. In many cases, you would edit `inputs.conf` in the `$SPLUNK_HOME/etc/system/local` directory. For HEC, you should edit `$SPLUNK_HOME/etc/apps/splunk_httpinput/local/inputs.conf`. No matter how many `inputs.conf` files a Splunk Enterprise instance has and where they reside, Splunk Enterprise combines all their settings, using the rules of location precedence, as described in configuration file precedence.

1. In the `$SPLUNK_HOME/etc/apps/splunk_httpinput` directory, create a directory local, if it does not exist.
2. Change to the `$SPLUNK_HOME/etc/apps/splunk_httpinput/local` directory.
3. Create an `inputs.conf` file if it does not exist.
4. Open `inputs.conf` for editing.
5. Specify global and token settings as described later in this topic.
   - The HEC token must be a **unique** GUID.
6. Save the file and close it.
7. Restart Splunk Enterprise for the changes to take effect.
Token-related settings

HEC stores settings related to token management in the `inputs.conf` file.

You can specify whether settings apply globally (to all tokens) or to specific tokens:

- The `[http]` stanza contains global settings that apply to all tokens.
- The `[token_name]` stanzas, where `token_name` indicates the token name as assigned by the user, apply to individual tokens. Settings specified here override settings specified within the `[http]` stanza.

The `inputs.conf` file contains basic explanatory information about each setting.

Global settings

The `[http]` stanza contains global settings that apply to all tokens.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dedicatedIoThreads</td>
<td>The number of dispatcher threads on the HTTP Event Collector server. The default value is 2. This setting should not be altered unless you have been requested to do so by Splunk Support. The value of this parameter should never be more than the number of physical CPU cores on your Splunk Enterprise server.</td>
</tr>
<tr>
<td>disabled</td>
<td>Whether tokens are disabled. 1 indicates true; 0 indicates false. The default value is 1. When set to 1 in the <code>[http]</code> stanza, this parameter disables all tokens.</td>
</tr>
<tr>
<td>enableSSL</td>
<td>Whether the HTTP Event Collector server protocol is HTTP or HTTPS. 1 indicates HTTPS is enabled; 0 indicates HTTP. The default value is 1. HTTP Event Collector shares SSL settings with the Splunk management server and cannot have enableSSL settings that differ from the settings on the Splunk management server.</td>
</tr>
<tr>
<td>index</td>
<td>The global default index. This parameter can be overridden when set in an individual token’s stanza, or by event data whose header contains an index parameter set to a different value. You can limit the set of allowed values for this parameter on a per-token basis by using the indexes parameter.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>maxSockets</td>
<td>The number of HTTP Event Collector connections (expressed as an integer) that Splunk Enterprise accepts simultaneously. You can limit this number to constrain resource usage. When set to 0, Splunk Enterprise automatically sets it to one third of the maximum allowable open files on the host. If this number is less than 50, it will be set to 50. If this number is greater than 400000, it will be set to 400000. If set to a negative number, no limit will be enforced. Defaults to 0.</td>
</tr>
<tr>
<td>maxThreads</td>
<td>The number of threads (expressed as an integer) that can be used by active HTTP transactions. You can limit this number to constrain resource usage. When set to 0, Splunk Enterprise automatically sets the limit to one third of the maximum allowable threads on the host. If this number is less than 20, it will be set to 20. If this number is greater than 150000, it will be set to 150000. If maxSockets is not negative and maxThreads is greater than maxSockets, then Splunk Enterprise sets maxThreads to be equal to maxSockets. If set to a negative number, no limit will be enforced. Defaults to 0.</td>
</tr>
<tr>
<td>outputGroup</td>
<td>The global default output group. An output group is a group of indexers set up by the Splunk Enterprise administrator to index the data. If there is no output group specified, event data will go to the local indexer. If the given output group is invalid, the data will be dropped and an error message will be logged to splunkd.log. For more information about specifying output groups, see Output group-related settings later in this topic.</td>
</tr>
<tr>
<td>port</td>
<td>The HTTP Event Collector server port. The default value is 8088. This port number must not already be in use.</td>
</tr>
<tr>
<td>sourcetype</td>
<td>The global default sourcetype. This parameter can be overridden when set in an individual token's stanza, or by event data whose header contains a sourcetype parameter set to a different value.</td>
</tr>
<tr>
<td>useDeploymentServer</td>
<td>Whether to use Deployment Server. When set to true (1), writes to the location specified by repositoryLocation property in serverclass.conf. Defaults to 0 and writes to $SPLUNK_HOME/etc/apps.</td>
</tr>
</tbody>
</table>
### Per-token settings

The `<token_name>` stanzas, where `<token_name>` indicates the token name as assigned by the user, apply to individual tokens. Settings specified here override settings specified within the [http] stanza.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| connection_host   | The type of default host for the token. This parameter can be set to any of the following literal values:  
  - `dns` indicates the host value is the reverse DNS entry for the IP address of the system sending the data.  
  - `ip` indicates the host value is the IP address of the system sending the data.  
  - `none` sets the host value to the connection host specified in the HTTP host header. This is typically the Splunk server's hostname. |
<p>| disabled          | Whether the token is disabled. 1 indicates true; 0 indicates false. The default value is 0.                                                |
| index             | The token's default index. This parameter can be overridden by event data whose header contains an <code>index</code> parameter set to a different value. |
| indexes           | A list of allowable indexes to which the data can be indexed.                                                                                     |
| persistentQueueSize | The maximum size of the persistent queue. The value of this parameter is in the form <code>&lt;integer&gt;[KB|MB|GB]</code>. The default value is 0, which indicates there is no persistent queue. Persistent queues can help prevent loss of transient data by saving data in an input queue to disk. When set, the value of the <code>persistentQueueSize</code> parameter must be more than the value of the <code>queueSize</code> parameter. For more information about persistent queues, see Use persistent queues to help prevent data loss. |
| source            | The token's default source. This parameter can be overridden by event data whose header contains a <code>source</code> parameter set to a different value. |
| queueSize         | The maximum size of the input queue in memory. The value of this parameter is in the form |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>&lt;integer&gt;[KB</td>
</tr>
<tr>
<td>sourcetype</td>
<td>The token's default sourcetype. This parameter can be overridden by event data whose header contains a sourcetype parameter set to a different value.</td>
</tr>
<tr>
<td>token</td>
<td>The HTTP Event Collector token. The token must be a unique GUID.</td>
</tr>
</tbody>
</table>

Output group-related settings

Settings that apply to forwarding and load balancing are stored in outputs.conf, including settings for specifying HTTP Event Collector output groups. These settings are the same ones that Splunk Enterprise admins use to manage forwarding and load balancing among indexers.

- The [tcpout] stanza defines the output groups to which the data will be forwarded.
- The [tcpout:<target_group>] stanza defines the configuration of the target output group indicated by <target_group>. You can have as many target groups as you want. If more than one target group is specified, the forwarder clones the data to each target group.

The outputs.conf file contains basic explanatory information about each setting. For more information, see about forwarding and receiving and configure forwarders with outputs.conf

Global settings

The [tcpout] stanza defines the output groups to which the data will be forwarded.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>defaultGroup</td>
<td>A comma-separated list of one or more target output group names in the form &lt;target_group&gt;, &lt;target_group&gt;, ... The names of the output groups are specified later in the outputs.conf file in the [tcpout:&lt;target_group&gt;] stanzas. Data will be sent to the specified groups.</td>
</tr>
</tbody>
</table>
Per-output group settings

The `[tcpout:target_group]` stanza defines the configuration of the target output group indicated by `<target_group>`. You can have as many target groups as you want. If more than one target group is specified, the forwarder clones the data to each target group.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>blockWarnThreshold</td>
<td>The output pipeline’s send failure count threshold. The default value is 100. After the threshold is met, a failure message will be displayed as a banner in the Splunk Enterprise UI. To effectively disable UI warnings, set this to a very large value (for example, 2000000).</td>
</tr>
<tr>
<td>server</td>
<td>Required. A comma-separated list of one or more Splunk servers to which to send data. The list is in the form `[&lt;ip&gt;</td>
</tr>
<tr>
<td></td>
<td>• IP or server name</td>
</tr>
<tr>
<td></td>
<td>• port number</td>
</tr>
</tbody>
</table>

Set up and use HTTP Event Collector from the CLI

You can use the `http-event-collector` parameter of the Splunk CLI and its options to administer a HTTP Event Collector instance on a Splunk Enterprise server. This topic lists the available HEC options.

For more information about the CLI, see the following:

• About the CLI
• Use the CLI to administer a remote Splunk server

CLI syntax

There are two syntaxes to use when you administer HEC via the CLI:

• The syntax for all other HEC actions (such as creating, deleting, and showing tokens, and so on)
• The syntax for sending data to HEC

Use the following syntax for all actions except sending data to HEC:
All HTTP Event Collector commands (except for send) assume that the first option following the command name is the name of the token. In addition, the create command assumes that the second option is a description of the token in quotation marks.

Use the following syntax for when you want to send data to HEC:

```
splunk http-event-collector send -uri <uri_value> -name <token-name> <data>
```

If you want to apply the CLI commands to the global configuration, do not use the `-name <token-name>` argument. For example, the following enables HTTP Event Collector:

```
splunk http-event-collector enable -uri <uri_value> <data>
```

**Supported CLI commands**

The following HTTP Event Collector-specific CLI commands are supported:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>create</td>
<td>Create a new token.</td>
</tr>
<tr>
<td>delete</td>
<td>Remove a token.</td>
</tr>
<tr>
<td>list</td>
<td>Show all available tokens.</td>
</tr>
<tr>
<td>update</td>
<td>Change token properties.</td>
</tr>
<tr>
<td>enable</td>
<td>Enable a token.</td>
</tr>
<tr>
<td>disable</td>
<td>Disable a token.</td>
</tr>
<tr>
<td>help</td>
<td>Show help.</td>
</tr>
<tr>
<td>send</td>
<td>Send data to an endpoint.</td>
</tr>
</tbody>
</table>

**Supported CLI parameters**

HEC supports the following CLI parameters. You must immediately follow a CLI parameter with its value. You must wrap any values that contain spaces in quotation marks.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>

225
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-uri</td>
<td>The Uniform Resource Identifier (URI) of the Splunk server in the form: scheme://host:port. As an alternative to setting this parameter, you can set the <code>$SPLUNK_URI</code> environment variable instead. The port number to use should be the management port of your Splunk server (by default, 8089), and not the HTTP Event Collector port (by default, 8088).</td>
</tr>
<tr>
<td>-auth</td>
<td>Splunk server user authentication in the form: username:password. If this parameter is missing, you are prompted for a username and password.</td>
</tr>
<tr>
<td>-name</td>
<td>The name of the token.</td>
</tr>
<tr>
<td>-disabled</td>
<td>Whether to disable the token. 1 indicates true; 0 indicates false.</td>
</tr>
<tr>
<td>-description</td>
<td>A description of the token.</td>
</tr>
<tr>
<td>-indexes</td>
<td>A list of indexes accepted by the token.</td>
</tr>
<tr>
<td>-index</td>
<td>The token default index. Splunk Enterprise assigns this value to data that doesn't already have an index value set.</td>
</tr>
<tr>
<td>-source</td>
<td>The token default source value. Splunk Enterprise assigns this value to data that doesn't already have a source value set.</td>
</tr>
<tr>
<td>-sourcetype</td>
<td>The token default sourcetype value. Splunk Enterprise assigns this value to data that doesn't already have a sourcetype value set.</td>
</tr>
<tr>
<td>-outputgroup</td>
<td>The token default outputgroup value. An output group is a group of indexers set up by the Splunk software administrator to index the data. Splunk Enterprise assigns this value to data that doesn't already have an outputgroup value set.</td>
</tr>
<tr>
<td>-port</td>
<td>The HTTP Event Collector server port. The default value is 8088, but you can change it using this parameter.</td>
</tr>
<tr>
<td>-enable-ssl</td>
<td>Whether the HTTP Event Collector server protocol is HTTP or HTTPS. 1 indicates HTTPS; 0 indicates HTTP.</td>
</tr>
</tbody>
</table>
| -dedicated-io-threads | The number of dispatcher threads on the HTTP Event Collector server. The default value is 2. This setting should not be altered unless you have been requested to
do so by Splunk Support. The value of this parameter should never be more than the number of physical CPU cores on your Splunk Enterprise server.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-output-format</td>
<td>The output format. txt indicates text; json indicates JSON. The default value is txt.</td>
</tr>
</tbody>
</table>

(end table)

**Example CLI syntax**

The following example CLI entry creates a token called "new-token," assigns it the given URI, gives it a description (in quotation marks), sets it to disabled, and indicates HTTP Event Collector data should be saved to the "log" index.

```
splunk http-event-collector create new-token -uri https://localhost:8089 -description "this is a new token" -disabled 1 -index log
```

The following example CLI entry enables the token called "myapp," assigns it the given URI, and sets the user authentication as shown:

```
splunk http-event-collector enable -name myapp -uri https://localhost:8089 -auth admin:changeme
```

The following example CLI entry sends data (“this is some data”) to HTTP Event Collector using the given token and URI.

```
splunk http-event-collector send -uri https://localhost:8089 -token new-token {"this is some data"}
```

**Use cURL to manage HTTP Event Collector tokens, events, and services**

**Manage HTTP Event Collector tokens with cURL**

All HTTP Event Collector token operations are available via the token management endpoint using cURL. The tokens are stored at the following REST API endpoint, assuming your Splunk server management address is https://localhost:8089:

https://localhost:8089/servicesNS/admin/splunk_httpinput/data/inputs/http/
List the existing HTTP Event Collector tokens using cURL

You can list the existing tokens using cURL. For example, the following example cURL command lists the tokens that exist on the Splunk server at https://localhost:8089 via the user "admin:"

curl -k
-u admin:changeme https://localhost:8089/servicesNS/admin/splunk_httpinput/data/inputs/http

Create an HTTP Event Collector token using cURL

To create a token using cURL, use the name property. For example, the following example CLI command creates a token called "mytoken," on the Splunk server at https://localhost:8089 via the user "admin:"

curl -k -u admin:changeme https://localhost:8089/servicesNS/admin/splunk_httpinput/data/inputs/http -d name=mytoken

Edit an HTTP Event Collector token using cURL

You can update any token property (except its name or value) using cURL. For example, the following example cURL command updates the description of the "mytoken" token on the Splunk server at https://localhost:8089 via the user "admin:"

curl -k
-u admin:changeme https://localhost:8089/servicesNS/admin/splunk_httpinput/data/inputs/http/mytoken -d description=abc

You can update any of the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disabled</td>
<td>Whether to disable the token. 1 indicates true; 0 indicates false.</td>
</tr>
<tr>
<td>description</td>
<td>A description of the token.</td>
</tr>
<tr>
<td>indexes</td>
<td>A list of indexes accepted by the token.</td>
</tr>
<tr>
<td>index</td>
<td>The token's default index. Splunk Enterprise assigns this value to data that doesn't already have an index value set.</td>
</tr>
<tr>
<td>source</td>
<td>The token's default source value. Splunk Enterprise assigns this value to data that doesn't already have a</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>source</td>
<td>The token's default <em>sourcetype</em> value. Splunk Enterprise assigns this value to data that doesn't already have a <em>sourcetype</em> value set.</td>
</tr>
<tr>
<td>sourcetype</td>
<td>The token's default <em>outputgroup</em> value. An output group is a group of indexers set up by the Splunk software administrator to index the data. Splunk Enterprise assigns this value to data that doesn't already have an <em>outputgroup</em> value set.</td>
</tr>
<tr>
<td>outputgroup</td>
<td>The HTTP Event Collector server port. The default value is 8088, but you can change it using this parameter.</td>
</tr>
<tr>
<td>port</td>
<td>Whether the HTTP Event Collector server's protocol is HTTP or HTTPS. 1 indicates HTTPS; 0 indicates HTTP.</td>
</tr>
<tr>
<td>enableSSL</td>
<td>The number of dispatcher threads on the HTTP Event Collector server. The default value is 2. This setting should not be altered unless you have been requested to do so by Splunk Support. The value of this parameter should never be more than the number of physical CPU cores on your Splunk Enterprise server.</td>
</tr>
<tr>
<td>dedicatedIoThreads</td>
<td>Returns an acknowledgment when events are indexed. Set to 1 to enable.</td>
</tr>
</tbody>
</table>

### Enable or disable an HTTP Event Collector token using cURL

You can enable or disable a token using cURL. Changing the status of one token does not change the status of other tokens. To enable or disable a token, use the POST command, the token name, and the enable or disable endpoint, respectively. For example, the following example disables the token called "mytoken" on the Splunk server at https://localhost:8089 via the user "admin:"

```
curl -k -X "POST" -u admin:changeme https://localhost:8089/servicesNS/admin/splunk_httpinput/data/inputs/http/mytoken/disable
```

Similarly, the following example enables the token called "mytoken" on the Splunk server at https://localhost:8089 via the user "admin:"

```
curl -k -X "POST" -u admin:changeme https://localhost:8089/servicesNS/admin/splunk_httpinput/data/inputs/http/mytoken/enable
```
Enable or disable HTTP Event Collector using cURL

You can enable or disable HTTP Event Collector itself by making a bulk change to all tokens using cURL. Simply leave out a token name when using the enable or disable endpoint. To enable or disable HTTP Event Collector, use the POST command and the enable or disable endpoint, respectively. For example, the following example disables HTTP Event Collector on the Splunk server at https://localhost:8089 via the user "admin:"

curl -k -X "POST" -u admin:changeme
https://localhost:8089/servicesNS/admin/splunk_httpinput/data/inputs/http/http/disable

Delete an HTTP Event Collector token using cURL

To delete a token using cURL, use the DELETE command and the token name. For example, the following example cURL command deletes the token called "mytoken" from the Splunk server at https://localhost:8089 via the user "admin:"

curl -k -X "DELETE" -u admin:changeme
https://localhost:8089/servicesNS/admin/splunk_httpinput/data/inputs/http/mytoken

Manage HEC events and services with cURL

The following commands show you how you can send events to and manage HEC services. They are not an all-inclusive list but give you an idea of the things that can be accomplished with HEC.

Send an event to HEC

The following example demonstrates basic HEC usage. It includes the Splunk server address with port and endpoint, the authentication token, and event data and metadata formatted according to the HEC event data format specification.

curl -k "https://mysplunkserver.example.com:8088/services/collector" \
   -H "Authorization: Splunk CF179AE4-3C99-45F5-A7CC-3284AA91CF67" \
   -d '{"event": "Hello, world!", "sourcetype": "manual"}''

Send an event to HEC using basic authentication

This example demonstrates basic authentication, which is an alternative to the HTTP Authentication. To use basic authentication, submit a colon-separated user/password pair in the request as the -u argument, using any string as the username and the token as the <password>: <user>:<password>.
Basic auth

curl -k -u "x:CF179AE4-3C99-45F5-A7CC-3284AA91CF67"
"https://mysplunkserver.example.com:8088/services/collector/event" \
-d '{"sourcetype": "mysourcetype", "event": "Hello, world!"}'

Send multiple events to HEC in one request

The following example demonstrates sending multiple events in one request. Though you can send multiple events in a single request, you cannot split one event across multiple requests.

curl -k "https://mysplunkserver.example.com:8088/services/collector" \
-H "Authorization: Splunk CF179AE4-3C99-45F5-A7CC-3284AA91CF67" \
-d '{"event": "Pony 1 has left the barn"}{"event": "Pony 2 has left the barn"}{"event": "Pony 3 has left the barn", "nested": {"key1": "value1"}}'

Send raw text to HEC

The following example demonstrates sending raw text to HEC. Note the use of the raw endpoint, plus the channel identifier and sourcetype specification, both of which are done using URL query parameters.

curl -k 
"https://mysplunkserver.example.com:8088/services/collector/raw?channel=00872DC6-AC83-4EDE-8AFE-8413C3825C4C&sourcetype=mydata" \
-H "Authorization: Splunk CF179AE4-3C99-45F5-A7CC-3284AA91CF67" -d '1, 2, 3... Hello, world!'

Send raw batched events to HEC

The following example demonstrates how to send raw, batched events to HEC. In this case, the command sends splunkd access logs. It indicates that the indexer should assign these events the sourcetype of splunkd_access, and specified that they should be sent to the main index.

# HEC Raw batching

curl -k  
"https://mysplunkserver.example.com:8088/services/collector/raw?channel=00872DC6-AC83-4EDE-8AFE-8413C3825C4C&sourcetype=splunkd_access&index=main" \
-H "Authorization: Splunk CF179AE4-3C99-45F5-A7CC-3284AA91CF67" \
-d '127.0.0.1 - admin [28/Sep/2016:09:05:26.875 -0700] "GET/servicesNS/admin/launcher/data/ui/views?count=-1 HTTP/1.0" 200 126721 -- -- 6ms  
127.0.0.1 - admin [28/Sep/2016:09:05:26.917 -0700] "GET/servicesNS/admin/launcher/data/ui/nav/default HTTP/1.0" 200 4367 -- -- 6ms  
127.0.0.1 - admin [28/Sep/2016:09:05:26.941 -0700] "GET
Send events to HEC with indexer acknowledgement enabled

The following example demonstrates how to send events to HEC with indexer acknowledgement enabled. Note that the sole difference between this example and the basic example is the inclusion of a channel identifier. Indexer acknowledgement also works with raw data.

```
# Indexer ack
curl -k "https://mysplunkserver.example.com:8088/services/collector?channel=00872DC6-AC83-4EDE-8AFE-8413C3825C4C"
  -H "Authorization: Splunk CF179AE4-3C99-45F5-A7CC-3284AA91CF67"
  -d '{"event": "Hello, world!", "sourcetype": "manual"}'
```

Check HEC indexer acknowledgement status

The following example demonstrates how to check the indexing status of a prior HEC request. It sends the request to the `ack` endpoint, and includes the `acks` key, which is set to the three acknowledgement identifiers (ackIDs) whose status is queried.

```
# Check ack status
curl -k "https://mysplunkserver.example.com:8088/services/collector/ack?channel=00872DC6-AC83-4EDE-8AFE-8413C3825C4C"
  -H "Authorization: Splunk CF179AE4-3C99-45F5-A7CC-3284AA91CF67"
  -d '{"acks": [1,3,4]}'
```

Extract JSON fields from events sent to HEC

The following example demonstrates how to instruct Splunk Enterprise or Splunk Cloud to extract JSON fields from the events sent to HEC.

```
# Extracting JSON fields
curl -k "https://mysplunkserver.example.com:8088/services/collector"
  -H "Authorization: Splunk CF179AE4-3C99-45F5-A7CC-3284AA91CF67"
  -d '{"sourcetype": "_json", "event": {"a": "value1", "b": ["value1_1", "value1_2"]}}'
```


**Extract Explicit JSON fields from events sent to HEC**

The following example is similar to the previous example, but it explicitly specifies the JSON fields.

```bash
# Explicit JSON fields
curl -k "https://mysplunkserver.example.com:8088/services/collector/event" \\
-H "Authorization: Splunk CF179AE4-3C99-45F5-A7CC-3284AA91CF67"
-d '{"event": "Hello, world!", "sourcetype": "cool-fields",
"device": "macbook", "users": ["joe", "bob"]}'
```

**About HTTP Event Collector Indexer Acknowledgment**

The indexer acknowledgment feature in HTTP Event Collector is available in Splunk Enterprise 6.4.0 and later, Splunk Light 6.4.0 and later, and the current releases of both Splunk Cloud (self-service and trial) and Splunk Light Cloud. Indexer acknowledgment is not yet supported in managed Splunk Cloud.

HTTP Event Collector (HEC) supports indexer acknowledgement. While similar in purpose and identical in name, indexer acknowledgment in HEC is not the same as the indexer acknowledgment capability described in protect against loss of in-flight data in the *Splunk Enterprise Forwarding Data* manual.

**Why indexer acknowledgment**

By default, when HEC receives an event successfully, it immediately sends an HTTP Status 200 to the sender. However, this simply means that the event data appears valid, and the status message is sent before the event data enters the processing pipeline. During processing, there are several places where, due to an outage or a system failure, events could be lost before they are indexed. While HEC has precautions in place to prevent data loss, it’s impossible to completely prevent such an occurrence, especially in the event of a hardware crash. This is where indexer acknowledgment comes in.

**How indexer acknowledgment works**

In current versions of Splunk software, you can enable indexer acknowledgment on a per-token basis. The indexer acknowledgment process is similar to a package tracking scenario:
A tracking number is issued upon shipment of a package, the package’s status is updated for the tracking number once it’s delivered, and then at your convenience, you check whether the package arrived successfully by using the tracking number to retrieve the status.

The following diagram illustrates the indexer acknowledgment process in order from top to bottom. Each step is referred to by number in the paragraphs that follow:

Each time a request is sent from a client to the HEC endpoint using a token with indexer acknowledgment enabled (1), the server returns an acknowledgment identifier to the client (2). The response body is simply a JSON object with the acknowledgment identifier, such as the following:

```json
{"ackID":"2"}
```

The client can then query the Splunk server with the identifier to verify whether all the events sent in the request that corresponds to that identifier have been indexed (3). The query is sent to a special endpoint (`/services/collector/ack`), and contains JSON-formatted data like the following, where the only key, "acks", is set to an array of the ackIDs whose status you are querying:

```json
{"acks":[0,1,2]}
```

Next, the server responds with the status information to the client (4). The body of the reply contains the status of each of the requests that were queried. A true status indicates that the event that corresponds to that ackID was replicated at the desired replication factor. A true status does not guarantee that the event was indexed, because the parsing pipeline might drop unparsable events. A
false status indicates that there is no status information for that ackID, or that the corresponding event has not been indexed. The corresponding event might not have been indexed yet, the ackID might not have been found, or some other problem may have occurred. For example:

{"acks": {"0": true, "1": false, "2": true}}

Because a false status could indicate any number of issues, only query an ackID during the timeframe in which the request could reasonably be expected to be in transit. Once a true status for an ackID has been retrieved, the server deletes that ackID's status information. If you query the same ackID again, the Splunk server will always return false for that ackID because its status information can no longer be found. For that reason, once you query an ackID and its status returns as true, avoid querying it again.

Enable indexer acknowledgement

You can enable indexer acknowledgement in Splunk Web or in inputs.conf.

In Splunk Web

When you create a new HEC token in Splunk Web, select the checkbox on the first screen labeled Enable indexer acknowledgement. Then continue with the token creation process.

Enable indexer acknowledgement

In inputs.conf

You can enable indexer acknowledgement for existing tokens by editing the HEC inputs.conf file.

1. Open the inputs.conf file, which is at the following path:
   - In *nix:
     `$SPLUNK_HOME/etc/apps/splunk_httpinput/local/inputs.conf`
   - In Windows:
     `%SPLUNK_HOME%\etc\apps\splunk_httpinput\local\inputs.conf`
2. Within the stanza that corresponds to the token for which you want to enable indexer acknowledgement, add the following line: `useACK=true`
3. Save and close the file.
About channels and sending data

Sending events with indexer acknowledgment enabled is similar to sending them without the setting enabled. However, there is one crucial difference: specifying a channel.

The concept of a channel was introduced in HEC primarily to prevent a fast client from impeding the performance of a slow client. When you assign one channel per client, because channels are treated equally on the Splunk server, one client can't affect another.

You must include a matching channel identifier both when sending data to HEC in an HTTP request and when requesting acknowledgement that events contained in the request have been indexed. If you don't, you will receive the error message, "Data channel is missing." Each request that includes a token for which indexer acknowledgement has been enabled must include a channel identifier, as shown in the following example cURL statement for Splunk Cloud, where <customer> indicates the account-specific portion of a Splunk Cloud URL, and <data> represents the event data portion of the request:

```
curl https://http-inputs-<customer>.splunkcloud.com/services/collector
-H "X-Splunk-Request-Channel: FE0ECFAD-13D5-401B-847D-77833BD77131" -H
"Authorization: Splunk BD274822-96AA-4DA6-90EC-18940FB2414C" -d
'<data>' -v
```

Alternatively, the X-Splunk-Request-Channel header field can be sent as a URL query parameter, as shown here:

```
curl
-H "Authorization: Splunk BD274822-96AA-4DA6-90EC-18940FB2414C" -d
'<data>' -v
```

Indexer acknowledgement also works with raw JSON data. In that case, the endpoint to use in requests is /services/collector/raw. For more information, see Format events for HTTP Event Collector.

Channels are designed so that you assign a unique channel to each client that sends data to HEC. Each channel has a channel identifier (ID), which must be a GUID but can be randomly generated. You assign channel IDs simply by including them in requests as shown in the examples above. When the Splunk server sees a new channel identifier, it creates a new channel.
Query for indexing status

Once you enable indexer acknowledgement for a token, every request sent to
HEC using that token will return the following acknowledgement identifier (ackID)
contained in a simple JSON object to the sender, where \(<\text{int}>\) represents a
unique integer identifier that corresponds to the request:

\[
\{\text{"ackID": "<int>"}\}
\]

To verify that the indexer has indexed the event(s) contained in the request,
query the following endpoint, where \(<\text{host}>\) and \(<\text{port}>\) represent the hostname
and port number of your Splunk server, respectively:

https://<host>:<port>/services/collector/ack

The query must contain JSON-formatted data like the following, where the only
key, "acks", is set to an array of the ackIDs whose status you are querying:

\[
\{\text{"acks": [0,1,2]}\}
\]

Following is an example cURL statement that queries the Splunk server for the
indexing status of the events contained in the requests with the identifiers "0", "1", "2", and "3":

```
curl -k https://<host>:<port>/services/collector/ack?channel=FE0ECFAD-13D5-401B-847D-77833BD77131
```

Both the data channel ID (\(?channel=FE0ECFAD-13D5-401B-847D-77833BD77131\))
and the auth header ("Authorization: Splunk
BD274822-96AA-4DA6-90EC-18940FB2414C") are required in this query. For more
information, see the previous section About channels and sending data.

The body of the reply contains the status of each of the request(s) for whose
status you queried. The following example response indicates that the requests
with the ackIDs "0" and "2" were successfully indexed, but the requests with the
ackIDs "1" and "3" were not successfully indexed:

\[
\{\text{"acks": {"0": true, "1": false, "2": true, "3": false}}\}
\]

237
Channel limits and indexing status expiration

Acknowledgement IDs and their corresponding status information are cached in memory. To prevent the server from running out of memory, and to prevent malicious or misbehaved clients, several new limit settings have been introduced. To prevent channels from being overloaded, and to prevent an excessive number of channels from being created, several new settings have been introduced to the http_input stanza in the limits.conf file:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value type</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_number_of_acked_requests_pending_query_per_ack_channel</td>
<td>int</td>
<td>10000000</td>
<td>Specifies the maximum number of ackIDs and their corresponding status information that are waiting to be queried in each channel. If a client makes many requests with indexer acknowledgement enabled, this setting prevents the client’s channel from becoming full of ackIDs and status information, preventing the client from receiving a server busy error.</td>
</tr>
<tr>
<td>max_number_of_ack_channel</td>
<td>int</td>
<td>1000000</td>
<td>Specifies the maximum number of channels that clients can acquire for this Splunk server instance. If a single client tries to acquire more than this number of channels, the client receives a server busy error.</td>
</tr>
</tbody>
</table>
of channels, the request will fail with server busy error. This is used to prevent a client from acquiring too many channels.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value type</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_number_of_acked_requests_pending_query</td>
<td>int</td>
<td>10000000</td>
<td>Specifies the maximum number of ackIDs and their corresponding status information in all channels.</td>
</tr>
</tbody>
</table>

To prevent the likelihood of the limits being reached, the Splunk server can clean up channels that are idle for a period of time and release the memory for those channels. You do this using the following settings, which are set at the global ([http] stanza) level in the inputs.conf file:

- In *nix: $SPLUNK_HOME/etc/apps/splunk_httpinput/local/inputs.conf
- In Windows: %SPLUNK_HOME%\etc\apps\splunk_httpinput\local\inputs.conf

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value type</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ackIdleCleanup</td>
<td>bool</td>
<td>false</td>
<td>When set to true, causes the server to remove channels that are idle for the number of seconds set in the maxIdleTime setting.</td>
</tr>
<tr>
<td>maxIdleTime</td>
<td>int</td>
<td>600</td>
<td>Specifies the maximum number of seconds that channels can be idle before they are removed.</td>
</tr>
</tbody>
</table>

Indexer acknowledgement client behavior

This section provides best practice information about how to set up a client for indexer acknowledgment. Follow these guidelines to ensure that the client
doesn't exhibit any malicious behavior or ends up hitting the limits described previously. An indexer acknowledgment client should:

- Create its own GUID to use as its channel identifier.
- Send requests using only that channel.
- Save each acknowledgement identifier (ackID) that is returned from requests to HEC.
- Continually poll the /services/collector/ack endpoint at an interval (for instance, every 10 seconds) to ensure that acknowledgement status is retrieved in a timely manner. Because status information is deleted from the Splunk server after it is retrieved by clients, this releases memory on the server.
- Resend any event data for which an acknowledgment hasn't been received within a certain amount of time (for instance, 5 minutes). It is safe to assume that, by that time, the event data has been lost. When you resend the event data, a good practice is to add some additional data in the resent event that indicates it may be duplicate data. It's possible the event was previously indexed but the status expired due to the cleanup of the channel, or the Splunk server may have been restarted, thus clearing the cache of statuses.

**Scale HTTP Event Collector with distributed deployments**

Use the HTTP event collector (HEC) as part of a distributed Splunk platform deployment. The Splunk software processes HEC data in the same way as any other input.

To use HEC on a managed Splunk Cloud deployment, contact Splunk Support, and open a ticket to enable HEC. Cloudworks stacks have HEC enabled by default.

You should be familiar with distributed Splunk Enterprise deployment before proceeding. For more information about distributed deployments, see Distributed Splunk Enterprise overview in the *Distributed Deployment Manual*, and Components of a Splunk Enterprise deployment in the *Capacity Planning Manual*. For more information about deployment server, see About deployment server and forwarder management in the *Updating Splunk Enterprise Instances manual*. 
Where to place the HEC

The HEC can be placed on heavy forwarders or indexers. The HEC, like any data input configuration, must reside on the component when the data enters the system. As a best practice, that is a heavy forwarder. HEC is not supported on universal forwarders. As with any data input configuration, you can also place the HEC directly on a clustered or non-clustered indexer, if necessary.

The deployment server will distribute any app that contains your HEC configurations. This configuration information includes:

- HTTP Event Collector default values (port, SSL, source type, index)
- SSL settings
- HTTP Event Collector tokens

Each HEC input entry must contain a valid UUID for the token. The HEC stores its configurations in the $SPLUNK_HOME/etc/apps/splunk_httpinput/ directory (%SPLUNK_HOME%/etc/apps/splunk_httpinput\ on Windows).

Place and distribute the HEC on heavy forwarders

Use HEC in a distributed Splunk platform deployment that uses forwarders. Use a deployment server to distribute HEC configurations to the heavy forwarders in your Splunk platform deployment.

If you plan to distribute HEC configurations through the deployment server, set the useDeploymentServer option in the [http] stanza of inputs.conf on the deployment server to 1. When this option is set to 1 and you make UI-based HEC changes on the deployment server, those changes are placed directly in the $SPLUNK_HOME/etc/deployment-apps/splunk_httpinput/ folder, rather than in $SPLUNK_HOME/etc/apps/splunk_httpinput/. See the inputs.conf spec file for further information.

See Deploy a heavy forwarder in the Forwarding Data manual to learn more.

Place and distribute the HEC on non-clustered indexers

Use HEC with non-clustered indexers in one of two ways:

- Place HEC on heavy forwarders that forward to the indexers, by using a deployment server to distribute configurations to the HEC.
• Place HECs directly on the indexers, by using a deployment server to distribute configurations to the HEC.

See About deployment server and forwarder management in the Updating Splunk Enterprise Instances manual to learn more.

For more information about configuring deployment clients, see Configure deployment clients in the Splunk Enterprise manual.

**Place and distribute the HEC on indexer cluster peer nodes**

Use HEC in a distributed Splunk platform deployment that uses indexer clustering.

Use the configuration bundle method to distribute HEC configurations to the peer nodes. Using your HEC's port number, preferred protocol (HTTP or HTTPS), SSL settings, and HTTP Event Collector tokens, connect the HECs on your deployment's forwarders with the peer nodes of your deployment's indexer cluster. Tokens are managed centrally on the Splunk Enterprise instance running the cluster master.

See the Use forwarders to get data into the indexer cluster section in the Managing Indexers and Clusters of Indexers manual.

See the Update common peer configurations and apps topic in the Managing Indexers and Clustered Indexers manual.

**Example serverclass.conf file**

A server class is a group of deployment clients that you can manage as a single unit. You assign the deployment clients you want to use in your HTTP Event Collector deployment to one common server class. Later, when you distribute HTTP Event Collector settings to the deployment clients, only members of that server class will receive the configuration settings. The following example serverclass.conf file defines a server class "FWD2Local" for HTTP Event Collector.

```
[globals]
whitelist.0=*  
restartSplunkd=true
stateOnClient = enabled

[serverClass:FWD2Local]
whitelist.0=*  
```
For the purposes of deploying HTTP Event Collector settings, you can think of HEC as an app called "splunk_httpinput." Within the stanzas, you can set client filtering attributes and several non-filtering attributes.

For more information about available client filtering attributes, see the section Define filters through serverclass.conf in the topic Set up client filters, in the Updating Splunk Enterprise Instances Manual. To learn more about available non-filtering attributes, see the section what you can configure for a server class in the Use serverclass.conf to define server classes topic in the Updating Splunk Enterprise Instances Manual.

See also

For more information about distributed deployment, see Distributed Splunk Enterprise overview in the Distributed Deployment Manual, and Components of a Splunk Enterprise deployment in the Capacity Planning Manual.

For more information about distributed deployment, including advanced configuration options and general examples, see the Updating Splunk Instances Manual.

Format events for HTTP Event Collector

HTTP Event Collector (HEC) receives events from clients in a series of HTTP requests. Each request can contain an HEC token, a channel identifier header, event metadata, or event data depending on whether your events are raw or JSON.

Support for parsing raw event text is available in Splunk Enterprise 6.4.0 and later, Splunk Light 6.4.0 and later, and in the current releases of Splunk Cloud and Splunk Light Cloud.

HEC token

Before HTTP Event Collector will accept your data for indexing, you must authenticate to the Splunk server on which it's running. You do this using the token you generate when you create a new HEC input. When you use the token management endpoint on the Splunk server to generate a token, it generates the token in the form of a GUID. This guarantees that the token is unique.
You have several ways to authenticate to the server:

**HTTP Authentication**

Place the token in the authorization header of each HTTP request as follows:

"Authorization: Splunk <hec_token>"

In context:

```
curl -k -H "Authorization: Splunk 12345678-1234-1234-1234-1234567890AB"
https://mysplunkserver.example.com:8088/services/collector/event -d
'{"sourcetype": "my_sample_data", "event": "http auth ftw!"}"
```

**Basic authentication**

Include a colon-separated user/password pair in the request after `-u`, inserting the HEC token as the `<password>`: "<user>:<password>". The `<user>` can be any string.

For example:

```
-u "x:<hec_token>"
```

In context:

```
curl -k -u "x:12345678-1234-1234-1234-1234567890AB"
https://mysplunkserver.example.com:8088/services/collector/event -d
'{"sourcetype": "my_sample_data", "event": "basic auth ftw!"}"
```

**Query string (Splunk Cloud only)**

Specify the HEC token as a query string in the URL that you specify in your queries to HEC. For example:

```
?token=<hec_token>
```

In context:

```
curl -k
https://mysplunkserver.example.com:8088/services/collector/event?token=12345678-1234-1234-1234-1234567890AB
-d '{"sourcetype": "my_sample_data", "event": "query string ftw!"}''
```

You must also enable query string authentication on a per-token basis. On your Splunk server, request Splunk Support to edit the file at 244
Within the stanza for each token you want to enable query string authentication, add the following setting (or change the existing setting, if applicable):

allowQueryStringAuth = true

Save and close the inputs.conf file.

For Splunk Cloud, you must open a Splunk Support ticket to set allowQueryStringAuth to true. Support for a toggle in Splunk Web for this setting is planned for a future release.

Channel identifier header

If your request includes raw events, you must include an X-Splunk-Request-Channel header field in the event, and it must be set to a unique channel identifier (a GUID). Following is an example of a cURL statement that constitutes a valid request: curl https://http-inputs-<customer>.splunkcloud.com/services/collector/raw -H "X-Splunk-Request-Channel: FE0ECFAD-13D5-401B-847D-77833BD77131" -H "Authorization: Splunk BD274822-96AA-4DA6-90EC-18940FB2414C" -d '<raw data string>' -v Alternatively, the X-Splunk-Request-Channel header field can be sent as a URL query parameter, as shown here:


If the token with which you are authenticating to HTTP Event Collector has indexer acknowledgement enabled, you must also include the channel identifier with your indexer status query. For more information, see Enable indexer acknowledgement.

Event metadata

This section describes the keys that can be included in event metadata. These keys are all optional. Any key-value pairs that are not included in the event will be set to values defined for the token on the Splunk server.

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

245
| **time** | The event time. The default time format is epoch time format, in the format `<sec>.<ms>`. For example, `1433188255.500` indicates 1433188255 seconds and 500 milliseconds after epoch, or Monday, June 1, 2015, at 7:50:55 PM GMT. |
| **host** | The **host** value to assign to the event data. This is typically the hostname of the client from which you're sending data. |
| **source** | The **source** value to assign to the event data. For example, if you're sending data from an app you're developing, you could set this key to the name of the app. |
| **sourcetype** | The **sourcetype** value to assign to the event data. |
| **index** | The name of the index by which the event data is to be indexed. The index you specify here must be within the list of allowed indexes if the token has the indexes parameter set. |
| **fields** | (Not applicable to raw data.) Specifies a JSON object that contains explicit custom fields to be defined at index time. Requests containing the "fields" property must be sent to the `/collector/event` endpoint, or they will not be indexed. For more information, see Indexed field extractions. |

With raw events, you can configure metadata at the global level (all tokens), at the token level, and at the request level using the query string. Metadata specified within a request will apply to all events that are extracted from the request.

### Event data

Event data can be assigned to the `event` key within the JSON object in the HTTP request, or it can be raw text. The `event` key is at the same level within the JSON event packet as the metadata keys. Within the `event` key value's curly brackets, the data can be in whatever format you want—a string, a number, another JSON object, and so on.

You can batch multiple events in one event packet by combining them within the request. By doing this, you are specifying that any event metadata within the request is to apply to all of the events contained in the request. Batching events can significantly speed performance when you need to index large quantities of data.
Examples

Following is an example of properly-formatted event metadata and event data (a string) contained within a JSON object:

```json
{
   "time": 1426279439, // epoch time
   "host": "localhost",
   "source": "random-data-generator",
   "sourcetype": "my_sample_data",
   "index": "main",
   "event": "Hello world!"
}
```

Here's an example of including a JSON object as the event data within a properly-formatted event:

```json
{
   "time": 1437522387,
   "host": "dataserver992.example.com",
   "source": "testapp",
   "event": {
      "message": "Something happened",
      "severity": "INFO"
   }
}
```

Here is an example of batched data. The batch protocol for HTTP Event Collector is simply event objects stacked one after the other as shown here, and not in a JSON array. Note that these events, though they only contain the "event" and "time" keys, are still valid:

```json
{
   "event":"event 1",
   "time": 1447828325
}

{
   "event":"event 2",
   "time": 1447828326
}
```

The following example is a simple "Hello, World!" cURL statement that includes the auth header, a destination endpoint, and very simple event data. Note that the request is going to the /services/collector/event endpoint, which is where all JSON-formatted event requests must go:
The following example cURL statement demonstrates sending raw event data. Note the addition of the channel ID, which is required when sending raw event data. Also, the request is going to the /services/collector/raw endpoint, which is where all raw event requests should go:

```bash
```

Alternately, this example cURL statement passes the channel ID as a URL parameter:

```bash
curl -k
http://localhost:8088/services/collector/raw?channel=18654C68-B28B-4450-9CF0-6E7645CA60CA
-H 'Authorization: Splunk B5A79AAD-D822-46CC-80D1-819F80D7BF0' -d 'hello world'
```

For additional examples on how to format and send event data using cURL, see Use cURL to manage HTTP Event Collector tokens, events, and services.

**Event parsing**

The HTTP Event Collector endpoint extracts the events from the HTTP request and parses them before sending them to indexers. Because the event data formats, as described in this topic, are pre-determined, Splunk Enterprise is able to parse your data quickly, and then sends it to be indexed. This results in improved data throughput and reduced event processing time compared to other methods of getting data in.

You can configure extraction rules in the `props.conf` file. To learn more, see Configure rule-based source type recognition in the Splunk Enterprise Getting Data In manual.

**Raw event parsing**

Available in Splunk Enterprise 6.4.0 and later, Splunk Light 6.4.0 and later, and the current releases of Splunk Cloud and Splunk Light Cloud.

HTTP Event Collector can parse raw text and extract one or more events. HEC expects that the HTTP request contains one or more events with line-breaking
rules in effect. Once HEC accepts the request, it passes its events into the pipeline, which extracts fields such as timestamps. HEC uses a line-breaking strategy that is based on the timestamp, but you can override it by setting a sourcetype in the props.conf file.

Events must be contained within a single HTTP request. They cannot span multiple requests.

To accommodate raw events, use the services/collector/raw endpoint.

This endpoint requires an additional X-Splunk-Request-Channel header field, which you must set to a unique channel identifier (a GUID). You must include a channel identifier with each HTTP request that contains raw events. The following is an example of a cURL statement that constitutes a valid request:

```
curl https://http-inputs-<customer>.splunkcloud.com/services/collector/raw
-H "X-Splunk-Request-Channel: FE0ECFAD-13D5-401B-847D-77833BD77131" -H "Authorization: Splunk BD274822-96AA-4DA6-90EC-18940FB2414C" -d '<raw data string>' -v
```

Alternatively, the X-Splunk-Request-Channel header field can be sent as a URL query parameter, as shown here:

```
-H "Authorization: Splunk BD274822-96AA-4DA6-90EC-18940FB2414C" -d '<raw data string>' -v
```

If the token with which you are authenticating to HTTP Event Collector has indexer acknowledgment enabled, you must also include the channel identifier with your indexer status query. For more information, see Enable indexer acknowledgement.

With raw events, you can configure metadata at the global level (all tokens), at the token level, and at the request level using the query string. Metadata specified within a request will apply to all events that are extracted from the request.

Timestamp extraction rules are enabled at the sourcetype level to extract timestamps. Most common timestamp formats are recognized—for example, the "current-time" key—but if no timestamp is able to be extracted, one is assigned based on the current time. For other metadata, you can configure extraction rules in the props.conf file.
For more examples of cURL requests to `services/collector/raw`, see Input endpoint examples in the *Splunk Enterprise REST API Reference Manual*.

For more information about channels, see the *About channels and sending data* in the *Enable indexer acknowledgement* topic.

**Automate indexed field extractions with HTTP Event Collector**

When Splunk software indexes data, it parses the data stream into a series of events. As part of this process, it adds a number of fields to the event data. These fields include default fields that it adds automatically and any custom fields that you specify. The process of adding fields to events is known as field extraction. There are two types of field extraction, search-time field extraction and indexed field extraction. Indexed fields are incorporated into the index at index time and become part of the event data.

The indexed field extractions feature in HTTP Event Collector is available in Splunk Enterprise 6.5.0 and later, Splunk Light 6.5.0 and later, and the current releases of both Splunk Cloud and Splunk Light Cloud.

Previously, setting up custom fields created at index time required significant configuration steps, as described in *Create custom fields at index time*, that involve editing the props.conf, transforms.conf, and fields.conf files to add regular expression extractions. Now, you can use HTTP Event Collector to automate this process, using built-in support for indexed field extractions.

Indexed field extraction does not work with data sent to the REST endpoint.

**Form HEC requests to trigger indexed field extractions**

You can trigger indexed extractions of JavaScript Object Notation (JSON) fields in two ways--as part of the main event data or separate from the event data but still associated with the event.

**Use nested JSON inside the "event" property**

Assign the `event` property (at the top level of the JSON being sent to HEC) to a JSON object that contains the custom fields to be indexed, as key-value pairs. For example, the following "event" property, from within an HTTP request sent to the Splunk server, specifies two custom fields: "club" and "wins":

```json
{ "event": { "club": "football", "wins": "3" } }
```
In this example, the `wins` property has been set to a multi-value JSON array. The `wins` field will be assigned both the values in the array.

At the same level as the `event` property, you must also include a `sourcetype` property, and set it to a sourcetype that has indexed extraction enabled. You can use any sourcetype that has INDEXED_EXTRACTIONS set to JSON in the `props.conf` file, including built-in sourcetypes such as `_json`. For example:

```
"sourcetype": "_json"
```

Following is an example cURL command that sends an event to HEC on a Splunk server. In this case, the event data contains two custom fields that will be extracted at index time:

```
# Extracting JSON fields
```

**Add a "fields" property at the top JSON level**

Include the `fields` property at the top level of the JSON being sent to HEC; that is, at the same level as the `event` property. This specifies explicit custom fields that are separate from the main `event` data. This method is useful if you don't want to include the custom fields with the event data, but you want to be able to annotate the data with some extra information, such as where it came from. Using this method is also typically faster than the nested JSON method.

Be aware that you must send HEC requests containing the `fields` property to the `/collector/event` endpoint. Otherwise, they will not be indexed.

Assign the `fields` property to a JSON object that contains the custom fields to be indexed, as key-value pairs. For example, the following `fields` property, from within an HTTP request sent to the Splunk server, specifies two custom fields: `club` and `wins`:

```
"fields": {"club": "glee", "wins": ["regionals", "nationals"]}
```

In this example, the `wins` property has been set to a multi-value JSON array. The `wins` field will be assigned both the values in the array.
At the same level as the `event` and `fields` properties, you must also include a `sourcetype` property, and set it to a sourcetype that has indexed extractions enabled. You can use any sourcetype that has `INDEXED_EXTRACTIONS` set to `JSON` in the props.conf file, including built-in sourcetypes such as `_json`. For example:

```
"sourcetype":"_json"
```

Following is an example cURL command that sends an event to HEC on a Splunk server. In this case, the event data contains two custom fields that will be extracted at index time:

```bash
# Explicit JSON fields
curl -k https://mysplunksserver.example.com:8088/services/collector/event
-H "Authorization: Splunk 12345678-1234-1234-1234-1234567890AB" -d
'{"event": "Hello, McKinley High!", "sourcetype": "_json", "fields":
{"club":"glee", "wins":['"regionals","nationals"]}}'
```

Only strings can be used as field values.

**Search for index-extracted fields**

After the data is indexed, you can search for this event using indexed extraction ("double-colon") notation, as shown here:

```
sourcetype=_json club::glee
```

For more information about using extracted fields to retrieve events, see Use fields to retrieve events in the Splunk Enterprise Search Manual.

**Send metrics to a metrics index**

If you gather metrics data, you can send it directly to a metrics index using HEC.

**Send metrics in JSON format**

Use the HEC `/collector` REST API endpoint to send metrics data over HTTP or HTTPS in JSON format from a client that is not natively supported to a metrics index.

Create a HEC data input for a Metrics index. Then, use the `/collector` REST API endpoint to send data to the metrics index using the following format:
You need to provide metrics event data and the following values:

- The Splunk host machine (an IP address, host name, or load balancer name)
- An HTTP port number
- An HEC token value

**Examples** The following example shows a command that sends a metric measurement to a metrics index, with the following values:

- Splunk host machine: "localhost"
- HTTP port number: "8088"
- HEC token value: "b0221cd8-c4b4-465a-9a3c-273e3a75aa29"

```
curl -k https://localhost:8088/services/collector \
-H "Authorization: Splunk b0221cd8-c4b4-465a-9a3c-273e3a75aa29" \
-d '{"time":1486683865.000,"source":"disk","host":"host_99","fields":{"region":"us-west-1","datacenter":"us-west-1a","rack":"63","os":"14.04","version":"0","service.environment":"test","path":"/dev/sda1","fstype":"ext3","_value":1099511627776,"metric_name":"total"}}'
```

This command sends two metrics measurements:

```
curl -k http://<IP address or host name or load balancer name>:8088/services/collector \
-H "Authorization: Splunk 98a1e071-bc35-410b-8642-78ce7d829083" \
-d '{"time":1505501013.000,"source":"disk","host":"host_99","fields":{"region":"us-west-1","datacenter":"us-west-1a","rack":"63","os":"14.04","version":"0","service_environment":"test","path":"/dev/sda1","fstype":"ext3","_value":999311222774,"metric_name":"total"}}' \
```

```
curl -k http://<IP address or host name or load balancer name>:8088/services/collector \
-H "Authorization: Splunk 98a1e071-bc35-410b-8642-78ce7d829083" \
-d '{"time":1505511013.000,"source":"disk","host":"host_99","fields":{"region":"us-west-1","datacenter":"us-west-1a","rack":"63","os":"14.04","version":"0","service_environment":"test","path":"/dev/sda1","fstype":"ext3","_value":1099511627776,"metric_name":"total"}}'
```

**Send metrics in collectd format**

Use the HEC /collector/raw REST API endpoint to send metrics data over HTTP or HTTPS in collectd JSON format to a metrics index.

Create a HEC data input for a Metrics index and select the Metrics > collectd_http pre-trained source type. Then, use the /collector/raw REST API endpoint to send data directly to a metrics index using the following format:

```
http://<Splunk_host>:<HTTP_port>/services/collector/raw?sourcetype=collectd_http \
```
You need to provide metrics data in collectd format and the following values:

- The Splunk host machine (an IP address, host name, or load balancer name)
- An HTTP port number
- An HEC token value

Because collectd is supported natively, you can configure collectd to send data to a metrics index. For more, see Get metrics in from collectd in the Metrics manual.

Example

The following example shows a command that sends a metric measurement to a metrics index, with the following values:

- Splunk host machine: "localhost"
- HTTP port number: "8088"
- HEC token value: "b0221cd8-c4b4-465a-9a3c-273e3a75aa29"

curl -k
https://localhost:8088/services/collector/raw?sourceType=collectd_http
   \n   -H "Authorization: Splunk b0221cd8-c4b4-465a-9a3c-273e3a75aa29"
   -d
   '[["values": [164.9196798931339196], "dstypes": ["derive"], "dsnames": ["value"], "time": 1505356687.894, "interval": 10.0, "host": "collectd", "plugin": "protocols", "plugin_instance": "IpExt", "type": "protocol_counter", "type_instance": "InOctets"]]'
<table>
<thead>
<tr>
<th>REST API endpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data/inputs/http</td>
<td>Access or update HTTP Event Collector global configuration tokens and application tokens.</td>
</tr>
<tr>
<td>data/inputs/http/{name}</td>
<td>Manage the {name} HTTP Event Collector token. HTTP, as in data/inputs/http/http, indicates global configuration.</td>
</tr>
<tr>
<td>data/inputs/http/{name}/disable</td>
<td>Disable the {name} HTTP Event Collector token.</td>
</tr>
<tr>
<td>data/inputs/http/{name}/enable</td>
<td>Enable the {name} HTTP Event Collector token.</td>
</tr>
<tr>
<td>services/collector</td>
<td>Send events to HTTP Event Collector using the Splunk platform JSON event protocol.</td>
</tr>
<tr>
<td>services/collector/ack</td>
<td>Query event indexing status.</td>
</tr>
<tr>
<td>services/collector/event</td>
<td>Sends timestamped events to HTTP Event Collector using the Splunk platform JSON event protocol when <code>auto_extract_timestamp</code> is set to &quot;true&quot; in the /event URL.</td>
</tr>
<tr>
<td></td>
<td>• If there is no timestamp in the event's JSON envelope, extraction is performed by leverage pipeline.</td>
</tr>
<tr>
<td></td>
<td>• If there is a timestamp, Splunk honors it.</td>
</tr>
<tr>
<td></td>
<td>• If &quot;time=xxx&quot; is used in the /event URL then <code>auto_extract_timestamp</code> is disabled.</td>
</tr>
<tr>
<td>services/collector/event/1.0</td>
<td>This endpoint works identically to services/collector/event but introduces a protocol version for future scalability.</td>
</tr>
<tr>
<td>services/collector/health</td>
<td>Checks the health of HTTP Event Collector. This endpoint is supported in versions 6.6.0+ of Splunk Enterprise.</td>
</tr>
<tr>
<td>services/collector/mint</td>
<td>Post MINT formatted data to the HTTP Event Collector.</td>
</tr>
<tr>
<td>services/collector/mint/1.0</td>
<td>This endpoint works identically to receivers/token/mint but introduces a protocol version.</td>
</tr>
<tr>
<td>REST API endpoint</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>services/collector/raw</td>
<td>Send raw data directly to the HTTP Event Collector.</td>
</tr>
<tr>
<td>services/collector/raw/1.0</td>
<td>This endpoint works identically to services/collector/raw but introduces a protocol version for future scalability.</td>
</tr>
</tbody>
</table>

### HTTP Event Collector Examples

#### Basic example

This example demonstrates basic HEC usage. It includes the Splunk server address including port and endpoint, the authentication token, and event data and metadata formatted according to the HEC event data format specification.

```
curl -k "https://mysplunkserver.example.com:8088/services/collector" 
   -H "Authorization: Splunk CF179AE4-3C99-45F5-A7CC-3284AA91CF67" 
   -d '{"event": "Hello, world!", "sourcetype": "manual"}"
```

#### Batching example

This example demonstrates sending multiple events in one request. Though you can send multiple events in a single request, you can't split one event across multiple requests.

```
curl -k "https://mysplunkserver.example.com:8088/services/collector" 
   -H "Authorization: Splunk CF179AE4-3C99-45F5-A7CC-3284AA91CF67" 
   -d '{"event": "Pony 1 has left the barn"}"event": "Pony 2 has left the barn"}"event": "Pony 3 has left the barn", "nested": {"key1": "value1"}'}
```

#### Raw example

This example demonstrates sending raw text to HEC. Note the use of the raw endpoint, plus the channel identifier and sourcetype specification, both of which are done using URL query parameters.

```
curl -k "https://mysplunkserver.example.com:8088/services/collector/raw?channel=00872DC6-AC83-4EDE-8AFE-8413C3825C4C&sourcetype=mydata"
   -H "Authorization: Splunk CF179AE4-3C99-45F5-A7CC-3284AA91CF67" 
   -d '1, 2, 3... Hello, world!'"
Raw example with batching

This example shows how to send raw, batched events to HEC. In this case, we’re sending splunkd access logs. We’ve indicated that the indexer should assign these events the sourcetype of splunkd_access, and specified that they should be indexed by main.

curl -k
"https://mysplunkserver.example.com:8088/services/collector/raw?channel=00872DC6-AC83-4EDE-8AFE-8413C3825C4C&sourcetype=splunkd_access&index=main"
- - 6ms
127.0.0.1 - admin [28/Sep/2016:09:05:26.917 -0700] "GET
/servicesNS/admin/launcher/data/ui/nav/default HTTP/1.0" 200 4367 -- -- 6ms
127.0.0.1 - admin [28/Sep/2016:09:05:26.941 -0700] "GET
/services/apps/local?search=disabled%3Dfalse&count=-1 HTTP/1.0" 200 31930 -- -- 3ms
127.0.0.1 - admin [28/Sep/2016:09:05:26.968 -0700] "GET
/servicesNS/admin/launcher/data/ui/views?digest=1&count=-1 HTTP/1.0" 200 58672 -- -- 5ms'

Indexer acknowledgement example

This example demonstrates how to send events to HEC with indexer acknowledgement enabled. Note that the sole difference between this example and the basic example is the inclusion of a channel identifier. Indexer acknowledgement also works with raw data.

curl -k
"https://mysplunkserver.example.com:8088/services/collector?channel=00872DC6-AC83-4EDE-8AFE-8413C3825C4C"
- - 6ms
127.0.0.1 - admin [28/Sep/2016:09:05:26.941 -0700] "GET
/services/apps/local?search=disabled%3Dfalse&count=-1 HTTP/1.0" 200 31930 -- -- 4ms
127.0.0.1 - admin [28/Sep/2016:09:05:26.954 -0700] "GET
/services/apps/local?search=disabled%3Dfalse&count=-1 HTTP/1.0" 200 31930 -- -- 3ms
127.0.0.1 - admin [28/Sep/2016:09:05:26.968 -0700] "GET
/servicesNS/admin/launcher/data/ui/views?digest=1&count=-1 HTTP/1.0" 200 58672 -- -- 5ms'

Check acknowledgement status example

This example demonstrates how to check the indexing status of a prior HEC request. Note that we’re sending the request to the ack endpoint, and we’re including "acks" key, which is set equal to the three acknowledgement identifiers (ackIDs) whose status we want to know.
curl -k
"https://mysplunkserver.example.com:8088/services/collector/ack?channel=00872DC6-AC83-4EDE-8AFE-8413C3825C4C"
  -H "Authorization: Splunk CF179AE4-3C99-45F5-A7CC-3284AA91CF67"
  -d '{"acks": [1,3,4]}'

Extract JSON fields example

This example demonstrates how to instruct Splunk Enterprise or Splunk Cloud to extract JSON fields from the events sent to HEC.

curl -k "https://mysplunkserver.example.com:8088/services/collector"
  -H "Authorization: Splunk CF179AE4-3C99-45F5-A7CC-3284AA91CF67"
  -d '{"sourcetype": "_json", "event": {"a": "value1", "b": ["value1_1", "value1_2"]}}'

Explicit JSON fields example

This example is similar to the previous example, but it explicitly specifies the JSON fields.

curl -k
"https://mysplunkserver.example.com:8088/services/collector/event"
  -H "Authorization: Splunk CF179AE4-3C99-45F5-A7CC-3284AA91CF67"
  -d '{"event": "Hello, world!", "sourcetype": "cool-fields", "fields": {"device": "macbook", "users": ["joe", "bob"]}}'

Basic authentication example

This example demonstrates basic authentication, which is an alternative to the HTTP Authentication that has been demonstrated in all of the previous examples. To use basic auth, place a colon-separated user/password pair in the request after -u as shown here, inserting the token as the <password> and any string (we've used x) as the <user>: "<user>:<password>".

curl -k -u "x:CF179AE4-3C99-45F5-A7CC-3284AA91CF67"
"https://mysplunkserver.example.com:8088/services/collector/event"
  -d '{"sourcetype": "mysourcetype", "event": "Hello, world!"}"

Troubleshoot HTTP Event Collector

Logging

HTTP Event Collector saves usage data about itself to log files. You can query these usage metrics using Splunk Enterprise or Splunk Cloud to explore usage
trends system-wide, per token, per sourcetype, and more, as well as to evaluate HTTP Event Collector performance. Metrics are logged whenever HTTP Event Collector is enabled. HTTP Event Collector is disabled by default, so it will not log data until you enable it.

You can also view HTTP Event Collector error logs in `splunkd.log`. See enable debug logging for how to enable debugging on your Splunk instance.

**Log file location and management**

HTTP Event Collector metrics are written to the `http_event_collector_metrics.log` file located at the following path:

```
$SPLUNK_HOME/var/log/introspection/splunk/
```

A new `http_event_collector_metrics.log` file is created when you start your Splunk Enterprise instance (or log off of and then onto Splunk Cloud). Any existing file with that name is renamed by giving it next higher available numeric extension. For example, if you restart Splunk Enterprise or log off of and onto Splunk Cloud and there exists `http_event_collector_metrics.log`, `http_event_collector_metrics.1`, and `http_event_collector_metrics.2` files, the `http_event_collector_metrics.log` file will be renamed `http_event_collector_metrics.3` and HTTP Event Collector will begin logging to a new `http_event_collector_metrics.log` file.

You configure the logging frequency of HTTP Event Collector metrics in the `limits.conf` file. 60 seconds is the default frequency. HTTP Event Collector continues logging system-level metrics even when there is no data input activity. When there is no activity, you can expect about 200 kilobytes (KB) of metrics log data to be produced every 24 hours. The maximum size of a metrics log file is 25 megabytes (MB). If a log file reaches that limit, the log file is renamed as described in the previous paragraph and a new one is created. Up to five metrics log files can be stored at a time. The `props.conf` file defines parameters for reading and indexing the metrics log file.

**Querying HTTP Event Collector metrics data**

HTTP Event Collector metrics data is indexed to the "_introspection" index. To query the accumulated HTTP Event Collector metrics using Splunk, use the following command:

```
index="_introspection" token
```
**Metrics log data format**

HTTP Event Collector metrics data is recorded to the log in JSON format. This means that the log is both easily human-readable and consistent with other Splunk Enterprise or Splunk Cloud log formats. A single entry consists of both input summary metrics (series = http_event_collector) and per-token metrics (series = http_event_collector_token), as shown in the following example:

```json
{
    "datetime":"09-01-2016 19:21:19.014 -0700",
    "log_level":"INFO",
    "component":"HttpEventCollector",
    "data":{
        "series":"http_event_collector",
        "transport":"http",
        "format":"json",
        "total_bytes_received":0,
        "total_bytes_indexed":0,
        "num_of_requests":0,
        "num_of_events":0,
        "num_of_errors":0,
        "num_of_parser_errors":0,
        "num_of_auth_failures":0,
        "num_of_requests_to_disabled_token":0,
        "num_of_requests_to_incorrect_url":0,
        "num_of_requests_in_mint_format":0,
        "num_of_ack_requests":0,
        "num_of_requests_acked":0,
        "num_of_requests_waiting_ack":0
    }
}
```

```json
{
    "datetime":"08-22-2016 12:38:04.854 -0700",
    "log_level":"INFO",
    "component":"HttpEventCollector",
    "data":{
        "token_name":"test",
        "series":"http_event_collector_token",
        "transport":"http",
        "format":"json",
        "total_bytes_received":57000,
        "total_bytes_indexed":44000,
        "num_of_requests":1000,
        "num_of_events":1000,
        "num_of_errors":0,
        "num_of_parser_errors":0,
        "num_of_requests_to_disabled_token":0,
        "num_of_requests_in_mint_format":0
    }
}
```
**HEC summary metrics**

System-wide summary metrics are always accumulated even if there is no input activity. These metrics are identified by "series":"http_event_collector".

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>component</td>
<td>HTTP Event Collector metrics data identifier.</td>
<td>HttpEventCollector</td>
</tr>
<tr>
<td>data:format</td>
<td>HTTP Event Collector data format.</td>
<td>json</td>
</tr>
<tr>
<td>data:num_of_auth_failures</td>
<td>Total number of authentication failures due to invalid token.</td>
<td>unsigned integer</td>
</tr>
<tr>
<td>data:num_of_errors</td>
<td>Total number of per-token errors, which include:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• bad data format</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• no authorization</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• bad authorization</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• connectivity problems</td>
<td></td>
</tr>
<tr>
<td>data:num_of_events</td>
<td>Total number of per-token events received by the HTTP Event Collector endpoint.</td>
<td>unsigned integer</td>
</tr>
<tr>
<td>data:num_of_parser_errors</td>
<td>Total number of per-token parser errors due to</td>
<td>unsigned integer</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Value</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>data:num_of_requests</td>
<td>incorrectly formatted event data.</td>
<td></td>
</tr>
<tr>
<td>data:num_of_requests_to_incorrect_url</td>
<td>Total number of valid per-token individual HTTP(S) requests received by an HTTP Event Collector endpoint. Each request can have one or more data events.</td>
<td>unsigned integer</td>
</tr>
<tr>
<td>data:num_of_requests_in_mint_format</td>
<td>Total number of requests from Splunk MINT.</td>
<td>unsigned integer</td>
</tr>
<tr>
<td>data:num_of_requests_to_disabled_token</td>
<td>Total number of per-token requests to disable token.</td>
<td>unsigned integer</td>
</tr>
<tr>
<td>data:series</td>
<td>Metrics data type.</td>
<td>http_eventCollector</td>
</tr>
<tr>
<td>data:total_bytes_indexed</td>
<td>Total amount of per-token data sent to the indexer.</td>
<td>unsigned integer</td>
</tr>
<tr>
<td>data:total_bytes_received</td>
<td>Total amount of per-token data received by calling the /receive/token endpoint.</td>
<td>unsigned integer</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Value</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>data:transport</td>
<td>Data transport protocol for HTTP Event Collector data.</td>
<td>http</td>
</tr>
<tr>
<td>datetime</td>
<td>Date and time associated with the data. Format: MM-DD-YYYY HH:MM:SS.SSS +/-GMTDELTA</td>
<td>string</td>
</tr>
<tr>
<td>log_level</td>
<td>Log severity level.</td>
<td>INFO</td>
</tr>
</tbody>
</table>

**Per-token metrics**

In contrast to the system-wide summary metrics, per-token metrics are accumulated only when HTTP Event Collector is enabled. These metrics are identified by "series":"http_event_collector_token".

The [http_input] stanza in the limits.conf file defines the logging interval and maximum number of tokens logged for these metrics.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>component</td>
<td>HTTP Event Collector metrics data identifier.</td>
<td>HttpEventCollector</td>
</tr>
<tr>
<td>data:format</td>
<td>HTTP Event Collector data format. (Always json for metrics logging.)</td>
<td>json</td>
</tr>
</tbody>
</table>
| data:num_of_errors | Number of errors, which include:  
  • bad data format  
  • no authorization | unsigned integer |
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>data:num_of_events</td>
<td>Number of events received by the HTTP Event Collector endpoint.</td>
<td>unsigned integer</td>
</tr>
<tr>
<td>data:num_of_parser_errors</td>
<td>Number of parser errors due to incorrectly formatted event data.</td>
<td>unsigned integer</td>
</tr>
<tr>
<td>data:num_of_requests</td>
<td>Number of valid individual HTTP(S) requests received by an HTTP Event Collector endpoint. Each request can have one or more data events.</td>
<td>unsigned integer</td>
</tr>
<tr>
<td>data:num_of_requests_in_mint_format</td>
<td>Total number of requests from Splunk MINT.</td>
<td>unsigned integer</td>
</tr>
<tr>
<td>data:num_of_requests_to_disabled_token</td>
<td>Number of requests to a disabled token.</td>
<td>unsigned integer</td>
</tr>
<tr>
<td>data:series</td>
<td>Metrics data type.</td>
<td>http_event_collector_token</td>
</tr>
<tr>
<td>data:token_name</td>
<td>Token name.</td>
<td>string</td>
</tr>
<tr>
<td>data:total_bytes_indexed</td>
<td>Total amount of data sent to the indexer.</td>
<td>unsigned integer</td>
</tr>
</tbody>
</table>

- bad authorization
- connectivity problems
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>data:total_bytes_received</td>
<td>Total amount of data received by calling the /receive/token endpoint.</td>
<td>unsigned integer</td>
</tr>
<tr>
<td>data:transport</td>
<td>Data transport protocol for HTTP Event Collector data.</td>
<td>http</td>
</tr>
<tr>
<td>datetime</td>
<td>Date and time associated with the data. Format: MM-DD-YYYY HH:MM:SS.SSS +/-GMTDELTA</td>
<td>string</td>
</tr>
<tr>
<td>log_level</td>
<td>Log severity level.</td>
<td>INFO</td>
</tr>
</tbody>
</table>

### Setting up logging with configuration files

**Configuration**

The limits.conf and props.conf files control metrics data logging and indexing behavior.

**limits.conf**

The [http_input] stanza in the $SPLUNK_HOME/etc/system/default/limits.conf file controls HTTP Event Collector metrics data logging.

For information about all HTTP Event Collector-related parameters, including those not related to metrics, see the [http_input] stanza documentation on limits.conf in the Splunk Enterprise Admin Manual.

### Example

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
</table>
max_number_of_tokens | 10000 | An unsigned integer that represents the maximum number of tokens reported by HTTP Event Collector metrics.

metrics_report_interval | 60 | An unsigned integer that represents the number of seconds in an HTTP Event Collector metrics report interval.

**props.conf**

The [http_event_collector_metrics] stanza in the $SPLUNK_HOME/etc/system/default/props.conf file controls reading and indexing the HTTP Event Collector log files.

**Example**

```sh
[source:.../http_event_collector_metrics.log(.*d+)?] sourcetype = http_event_collector_metrics
...

[http_event_collector_metrics]
SHOULD_LINEMERGE = false
TIMESTAMP_FIELDS = datetime
TIME_FORMAT = %m-%d-%Y %H:%M:%S.%l %z
INDEXED_EXTRACTIONS = json
KV_MODE = none
JSON_TRIM_BRACES_IN_ARRAY_NAMES = true
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHOULD_LINEMERGE</td>
<td>false</td>
<td>Specifies layout of events per line:</td>
</tr>
</tbody>
</table>

- true
  - Allow multiple events in the same

266
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TIMESTAMP_FIELDS</strong></td>
<td>datetime</td>
<td>Log entry time field name.</td>
</tr>
<tr>
<td><strong>TIME_FORMAT</strong></td>
<td>%m-%d-%Y %H:%M:%S.%L %z</td>
<td>Log entry time field format.</td>
</tr>
<tr>
<td><strong>INDEXED_EXTRCTIONS</strong></td>
<td>json</td>
<td>Metrics log format. (Always json for metrics logging.)</td>
</tr>
<tr>
<td><strong>KV_MODE</strong></td>
<td>none</td>
<td>Key-value data indicator:</td>
</tr>
<tr>
<td><strong>JSON_TRIM_BRACES_IN_ARRAY_NAMES</strong></td>
<td>true</td>
<td>Whether to trim brace characters from JSON array names</td>
</tr>
</tbody>
</table>
Possible error codes

The following status codes have particular meaning for all HTTP Event Collector endpoints:

<table>
<thead>
<tr>
<th>Status Code</th>
<th>HTTP status code ID</th>
<th>HTTP status code</th>
<th>Status message</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>200</td>
<td>OK</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>403</td>
<td>Forbidden</td>
<td>Token disabled</td>
</tr>
<tr>
<td>2</td>
<td>401</td>
<td>Unauthorized</td>
<td>Token is required</td>
</tr>
<tr>
<td>3</td>
<td>401</td>
<td>Unauthorized</td>
<td>Invalid authorization</td>
</tr>
<tr>
<td>4</td>
<td>403</td>
<td>Forbidden</td>
<td>Invalid token</td>
</tr>
<tr>
<td>5</td>
<td>400</td>
<td>Bad Request</td>
<td>No data</td>
</tr>
<tr>
<td>6</td>
<td>400</td>
<td>Bad Request</td>
<td>Invalid data format</td>
</tr>
<tr>
<td>7</td>
<td>400</td>
<td>Bad Request</td>
<td>Incorrect index</td>
</tr>
<tr>
<td>8</td>
<td>500</td>
<td>Internal Error</td>
<td>Internal server error</td>
</tr>
<tr>
<td>9</td>
<td>503</td>
<td>Service Unavailable</td>
<td>Server is busy</td>
</tr>
<tr>
<td>10</td>
<td>400</td>
<td>Bad Request</td>
<td>Data channel is missing</td>
</tr>
<tr>
<td>11</td>
<td>400</td>
<td>Bad Request</td>
<td>Invalid data channel</td>
</tr>
<tr>
<td>12</td>
<td>400</td>
<td>Bad Request</td>
<td>Event field is required</td>
</tr>
<tr>
<td>13</td>
<td>400</td>
<td>Bad Request</td>
<td>Event field cannot be blank</td>
</tr>
<tr>
<td>14</td>
<td>400</td>
<td>Bad Request</td>
<td>ACK is disabled</td>
</tr>
<tr>
<td>15</td>
<td>400</td>
<td>Bad Request</td>
<td>Error in handling indexed fields</td>
</tr>
<tr>
<td>16</td>
<td>400</td>
<td>Bad Request</td>
<td>Query string authorization is not enabled</td>
</tr>
</tbody>
</table>

Monitoring Console

The **Monitoring Console** provides prebuilt dashboards for HTTP event collector that you can use to investigate your instance’s performance.

- For Splunk Cloud, see Monitor Splunk Cloud deployment health.
- For Splunk Enterprise, see About the Monitoring Console.
The Monitoring Console provides a pre-built dashboard to monitor HTTP Event Collector. See HTTP Event Collector Dashboards in the Monitoring Splunk Enterprise manual.

**Detecting scaling problems**

If you are experiencing performance slowdowns, or want to speed up your HTTP Event Collector deployment, the following factors can affect performance:

- **HTTP vs. HTTPS**: There is a significant performance improvement when sending data over HTTP versus sending data over HTTPS.
- **Batching**: If you batch multiple events into single requests, it can speed up data transmission. Because a request's metadata applies to all events in the request, less data is sent overall. For more information about how event data is packaged, see format events for HTTP Event Collector.
- **HTTP Keep-alive**: Setting keepalive on your connection can increase performance. As long as the client sending the data supports HTTP 1.1 and is set up to support a persistent connection, you're taking advantage of keep-alive.
- **Persistent queues**: Persistent queuing slows down performance by storing data in an input queue to disk. For more information, see use persistent queues to help prevent data loss.
- **Use index-time field extraction**: For more information about index-time field extraction, see Create custom fields at index time.
Get other kinds of data in

Monitor First In, First Out (FIFO) queues

This topic describes how to configure a First In, First Out (FIFO) input by editing the inputs.conf file on a Splunk Enterprise instance (Splunk Web does not currently support the definition of FIFO inputs.) If you have Splunk Cloud, use a heavy forwarder to read FIFO queues.

Note: Data that you send over FIFO queues does not remain in computer memory and can be an unreliable method for data sources. To ensure data integrity, use the monitor input 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 configuration files before, read About Configuration Files in the Admin manual before you begin.

This input stanza configures Splunk Enterprise to read from a FIFO queue at the specified path.

[fifo://<path>]
<attribute1> = <val1>
<attribute2> = <val2>
...

You can use the following attributes with FIFO stanzas:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>host = &lt;string&gt;</td>
<td>The host key/field to a static value for this stanza. The &lt;string&gt; is prepended with 'host::'. Sets the host key's initial value. This key is used during parsing and indexing</td>
<td>The IP address or fully qualified domain name of the host where the data originated</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>index = &lt;string&gt;</td>
<td>The index where events from this input will be stored. The &lt;string&gt; is prepended with 'index::'.</td>
<td>main, or whatever you have set as your default index.</td>
</tr>
</tbody>
</table>
| sourcetype = <string> | The sourcetype key/field for events from this input. Explicitly declares the source type for this data, as opposed to letting it be determined automatically. This is important both for searchability and for applying the relevant formatting for this type of data during parsing and indexing. Sets the sourcetype key’s initial value. This value is used during parsing and indexing to set the source type field. It is also the source type field used at search time.  
  - The <string> is prepended with 'sourcetype::'.  
  - For more information about source types, see Why source types matter in this manual.             | Splunk software picks a source type based on various aspects of the data. There is no hard-coded default. |
| source = <string>  | Sets the source key/field for events from this input. The <string> is prepended with 'source::'.                                                                    | The input file path.                                                   |
Do not override the source field unless absolutely necessary. The input layer provides a more accurate string to aid in problem analysis and investigation, accurately recording the file from which the data was retrieved. Consider use of source types, tagging, and search wildcards before overriding this value.

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.

Monitor changes to your file system

This feature has been deprecated.

This feature has been deprecated as of Splunk Enterprise version 5.0. This means that although it continues to function in version 6.x of Splunk software, it might be removed in a future version. As an alternative, you can:

- Learn how to monitor file system changes on Windows systems.
- Use the auditd daemon on *nix systems and monitor output from the daemon.

For a list of all deprecated features, see the topic Deprecated features in the Release Notes.

The Splunk Enterprise file system change monitor tracks changes in your file system. The monitor watches a directory you specify and generates an event when that directory undergoes a 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 configurations change.

To monitor file system changes on Windows, see Monitor file system changes in
this manual to learn how with Microsoft 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:

- whitelist using regular expressions
  - specify files that will be checked, no matter what
- blacklist 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 Enterprise

By default, the file system change monitor generates audit events whenever the
contents of $SPLUNK_HOME/etc/ are changed, deleted, or added to. When you
start Splunk Enterprise for the first time, it generates an audit event for each file
in the $SPLUNK_HOME/etc/ directory and all subdirectories. Afterward, any change
in configuration (regardless of origin) generates an audit event for the affected
file. If you have configured signedaudit=true, Splunk Enterprise indexes the file
system change into the audit index (index=_audit). If signedaudit is not turned
on, by default, Splunk Enterprise writes the events to the main index unless you
specify another index.
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.

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

### Configure the file system change monitor

Configure the file system change monitor in inputs.conf. There is no support for configuring the file system change monitor in Splunk Web. You must restart Splunk Enterprise any time you make changes to the `[fschange]` stanza.

1. Open `inputs.conf`.

2. Add `[fschange:<directory>]` stanzas to specify files or directories that Splunk Enterprise should monitor for changes.

3. Save the inputs.conf file and close it.

4. Restart Splunk Enterprise. File system change monitoring begins immediately.

If you want to use this feature with **forwarding**, follow these guidelines:

- To send the events to a remote indexer, use a **heavy forwarder**.
- If you cannot use a heavy forwarder, then follow the configuration instructions at Use with a universal forwarder.

To use the file system change monitor to watch any directory, add or edit an `[fschange]` stanza to `inputs.conf` in `$SPLUNK_HOME/etc/system/local/` or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For information on configuration files in general, see About configuration files in the Admin manual.

**Syntax**

Here is the syntax for the `[fschange]` stanza:

```
[fschange:<directory or file to monitor>]
<attribute1> = <val1>
<attribute2> = <val2>
...
```
Note the following:

- Splunk Enterprise monitors all adds/updates/deletes to the directory and its subdirectories.
- Any change generates an event that Splunk indexes.
- `<directory or file to monitor>` defaults to `$SPLUNK_HOME/etc/`.

**Attributes**

All attributes are optional. Here is the list of available attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>index=&lt;indexname&gt;</td>
<td>The index where all generated events should be stored.</td>
<td>main (unless you have turned on audit event signing).</td>
</tr>
<tr>
<td>recurse=&lt;true</td>
<td>false&gt;</td>
<td>Whether or not to recurse all directories within the directory specified in <code>&lt;code[fschange]</code>. Set to true to recurse all subdirectories and false to specify only the current directory.</td>
</tr>
<tr>
<td>followLinks=&lt;true</td>
<td>false&gt;</td>
<td>Whether or not the file system change monitor follows symbolic links. Set to true to follow symbolic links and false not to follow symbolic links.</td>
</tr>
<tr>
<td>pollPeriod=N</td>
<td>Check this directory for changes every N seconds. If you make a change, the file system audit events</td>
<td>3600 seconds</td>
</tr>
</tbody>
</table>
could take anywhere between 1 and 3600 seconds to be generated and become available in audit search.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>hashMaxSize=N</td>
<td>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.</td>
<td>-1 (no hashing used for change detection).</td>
</tr>
<tr>
<td>signedaudit=&lt;true</td>
<td>Send cryptographically signed add/update/delete events. Set to true to generate events in the _audit index. Set to false if you're setting the index attribute. <strong>Note:</strong> When setting signedaudit to true, make sure auditing is enabled in audit.conf.</td>
<td>false</td>
</tr>
<tr>
<td>fullEvent=&lt;true</td>
<td>* Send the full event if an add or update change is detected.</td>
<td>false</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>sendEventMaxSize=N</td>
<td>* 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.</td>
<td>-1 (unlimited.)</td>
</tr>
<tr>
<td>sourcetype = &lt;string&gt;</td>
<td>Set the source type for events from this input. &quot;sourcetype:::&quot; is prepended to &lt;string&gt;.</td>
<td>audittrail (if signedaudit=true) or fs_notification (if signedaudit=false)</td>
</tr>
<tr>
<td>filesPerDelay = &lt;integer&gt;</td>
<td>Injects a delay specified by delayInMills after processing &lt;integer&gt; files. This throttles file system monitoring so it does not consume as much CPU.</td>
<td>n/a</td>
</tr>
<tr>
<td>delayInMills = &lt;integer&gt;</td>
<td>The delay in milliseconds to use after processing every &lt;integer&gt; files as specified in filesPerDelay. This is used to throttle file system</td>
<td></td>
</tr>
</tbody>
</table>
To define a filter to use with the filters attribute, add a [filter...] stanza as follows:

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

The following list describes how Splunk Enterprise handles fschange whitelist and blacklist logic:

- 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, then Splunk Enterprise indexes the event.
- If the first filter to match an event is a blacklist, the filter prevents the event from getting indexed.
- If an event reaches the end of the chain with no matches, then Splunk Enterprise indexes the event. This means that there is an implicit "all pass" filter built in.
To default to a situation where Splunk Enterprise does not index events if they don't match a whitelist explicitly, end the chain with a blacklist that matches all remaining events.

For example:

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

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

If you blacklist a directory including a terminal blacklist at the end of a series of whitelists, then Splunk Enterprise blacklists all its subfolders and files, as they do not pass any whitelist. To accommodate this, whitelist all desired folders and subfolders explicitly ahead of the blacklist items in your filters.

**Example of explicit whitelisting and terminal blacklisting**

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

In this example, a directory could be blacklisted. If this is the case, Splunk Enterprise blacklists all of its subfolders and files 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
```
Use with a universal forwarder

To forward file system change monitor events from a universal forwarder, you must set `signedaudit = false` and `index=_audit`.

```
[fchange:<directory or file to monitor>]
signedaudit = false
index=_audit
```

With this workaround, Splunk Enterprise indexes file system change monitor events into the `_audit` index with `sourcetype` set to `fs_notification` and `source` set to `fschangemonitor`, instead of the default value of `audittrail` for both `sourcetype` and `source`.

Get data from APIs and other remote data interfaces through scripted inputs

Splunk Enterprise can accept events from scripts that you provide. Scripted input is useful in conjunction with some Windows and *nix command-line tools, such as `ipconfig`, `iostat`, `netstat`, `top`, and so on. You can use scripted input to get data from application program interfaces (APIs) and other remote data interfaces and message queues. You can then use commands like `vmstat` and `iostat` on that data to generate metrics and status data. On Windows platforms, you can enable text-based scripts, such as those in Perl and Python, with an intermediary Windows batch (.bat) or PowerShell (.ps1) file.

This topic describes how to add scripted inputs that you have already written. To learn how to write scripted inputs, see Build scripted inputs in the Developing Views and Apps for Splunk Web manual.

You can configure scripted inputs from the Settings menu or by editing `inputs.conf`.

When a scripted input launches a script, that script inherits the Splunk Enterprise environment. Clear any environment variables that can affect the operation of a script. The only environment variable that could cause problems is the library path (most commonly known as `LD_LIBRARY_PATH` on Linux, Solaris, and FreeBSD).

Splunk Enterprise logs any messages that scripted inputs send to the `stderr` I/O channel to `splunkd.log`. 
Add a scripted input in Splunk Web

Go to the Add New page

You can get there by two routes.

- Splunk Home
- Splunk Settings

By Splunk Settings:

1. Click Settings in the upper right corner of Splunk Web.
2. Click Data Inputs.
3. Click Scripts.
4. Click New to add an input.

By Splunk Home:

1. Click the Add Data link in Splunk Home.
2. Click Monitor to monitor a script on the local machine, or Forward to forward data from a script on a remote machine. Splunk Web displays the "Add Data - Select Source" page.
3. In the left pane, locate and select Scripts.

Note: Forwarding data from scripted inputs requires additional setup.

Select the input source

1. In the Script Path drop down, select the path where the script resides. Splunk Web updates the page to include a new drop down list, "Script Name."
2. In the Script Name drop-down, select the script that you want to run. Splunk Web updates the page to populate the "Command" field with the script name.
3. In the Command field, add any arguments needed to invoke the script.
4. In the Interval field, enter the amount of time (in seconds) that Splunk Enterprise should wait before invoking the script.
5. Optionally, In the Source Name Override field, enter a new source name to override the default source value, if necessary.
6. Click Next.
**Specify input settings**

The **Input Settings** page lets you specify application context, default host value, and index. All of these parameters are optional. Learn more about setting the host value in "About hosts".

When you set the **Host** on this page, this only sets the **host** field in the resulting events. It does not direct Splunk Enterprise to look on a specific host on your network.

1. Select the source type for the script. You can choose **Select** to pick from the list of available source types on the local machine, or "Manual" to enter the name of a source type.
2. Select the appropriate **Application context** for this input.
3. Set the **Host** name value. You have several choices for this setting.
4. Set the **Index** that Splunk Enterprise should send data to. 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 Enterprise has a number of utility indexes, which also appear in this drop down box.
5. Click **Review**.

**Review your choices**

After specifying all your input settings, review your selections. Splunk Web lists all options you selected, including the type of monitor, the source, the source type, the application context, and the index.

1. Review the settings.
2. If they do not match what you want, click < to go back to the previous step in the wizard. Otherwise, click **Submit**.

Splunk Web displays the "Success" page.

**Add a scripted input with inputs.conf**

You add a scripted input in **inputs.conf** by adding a **[script]** stanza.

**Syntax**

The syntax for the **[script]** stanza follows:

```
[script://$SCRIPT]
```
$SCRIPT is the full path to the location of the script.
$SCRIPT can also be a file path that ends in the .path suffix. This special suffix lets you use the stanza to point to another command or script that exists anywhere on the host filesystem. See Use the .path suffix to reference external scripts. The file that you reference in the stanza must heed the location restrictions that are described in "Where to place the scripts for scripted inputs" in this topic.

Where to place the scripts for scripted inputs

The script that you reference in $SCRIPT can only reside in one of the following places on the host file system:

- $SPLUNK_HOME/etc/system/bin
- $SPLUNK_HOME/etc/apps/<your_app>/bin
- $SPLUNK_HOME/bin/scripts

As a best practice, put your script in the bin/ directory that is nearest the inputs.conf that calls your script on the host filesystem. For example, if you configure $SPLUNK_HOME/etc/system/local/inputs.conf, place your script in $SPLUNK_HOME/etc/system/bin/. If you work on an application in $SPLUNK_HOME/etc/apps/$APPLICATION/, put your script in $SPLUNK_HOME/etc/apps/$APPLICATION/bin/.

Attributes

All attributes are optional. Here is the list of available attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>interval = &lt;number&gt;</td>
<td>How often to execute the specified command. Specify either an integer value representing seconds or a valid cron schedule.</td>
<td>60 seconds</td>
</tr>
<tr>
<td></td>
<td>When you specify a cron schedule, the script does not execute on start up, but rather at the times that the cron schedule defines.</td>
<td></td>
</tr>
</tbody>
</table>
Splunk Enterprise keeps one invocation of a script per instance. Intervals are based on when the script completes. If you configure a script 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 interval). For one-shot data streams, enter -1. Setting `interval` to -1 causes the script to run each time at start-up.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>index = &lt;string&gt;</td>
<td>The index where events from this input should be stored. Splunk Enterprise prepends the <code>&lt;string&gt;</code> with <code>index::</code>. For more information about the index field, see &quot;How indexing works&quot; in the Managing Indexers and Clusters manual.</td>
<td>main, or whatever you have set as your default index.</td>
</tr>
<tr>
<td>sourcetype = &lt;string&gt;</td>
<td>Sets the sourcetype key/field for events from this input. * The <code>&lt;string&gt;</code> is prepended with 'sourcetype::'. Explicitly declares the source type for this data, as opposed to letting it be determined automatically. This is important both for searchability and for applying the relevant formatting for this type of data during parsing and indexing. Sets the sourcetype key initial value. Splunk Enterprise picks a source type based on various aspects of the data. There is no hard-coded default.</td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>indexing</td>
<td>It also uses the source type field at search time.</td>
<td></td>
</tr>
<tr>
<td>source = &lt;string&gt;</td>
<td>* Sets the source key/field for events from this input.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>Note:</strong> Do not override the source key unless absolutely necessary. Typically, the input</td>
<td></td>
</tr>
<tr>
<td></td>
<td>layer will provide a more accurate string to aid in problem analysis and investigation, accurately</td>
<td></td>
</tr>
<tr>
<td></td>
<td>recording the file from which the data was retrieved. Consider use of source types, tagging, and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>search wildcards before overriding this value.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Splunk Enterprise prepends &lt;string&gt; with source::..</td>
<td></td>
</tr>
<tr>
<td>disabled = &lt;true</td>
<td>Whether or not the input should run. Set to true if you want to disable the input.</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>false&gt;</td>
<td></td>
</tr>
</tbody>
</table>

### Run scripts continuously

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 Enterprise keeps track of scripts it has spawned and shuts them down on exit.

### Use a wrapper script

It is best practice to write a wrapper script for scripted inputs that use commands with arguments. In some cases, the command can contain special characters that the scripted input escapes when it validates text that you have entered in Splunk Web. This causes updates to a previously configured input to fail to save.

Splunk Enterprise escapes characters that should not be in paths, such as the equals sign (\=) and semicolon (\;), when it validates text. For example, the following scripted input is not correctly saved when you edit it in Splunk Web because the scripted input escapes the equals (\=) sign in the parameter to the
myUtil.py utility:

[script://$SPLUNK_HOME/etc/apps/myApp/bin/myUtil.py file=my_data.csv]
disabled = false
To avoid this problem, write a wrapper script that contains the scripted input, or use the special .path argument for the scripted input stanza name. For information on writing wrapper scripts, see Scripted inputs overview in the Developing Views and Apps for Splunk Web manual.

When you update scripted Inputs by editing inputs.conf directly, this validation does not occur.

Use the .path suffix to reference external scripts

As an alternative to writing a wrapper script, you can configure the scripted input to reference a script or executable that is anywhere on the host file system.

The script that you reference can have a single line that calls the script or executable that you want. You can use this file to call a runtime environment that is outside of the Splunk Enterprise environment. For example, if you have both Splunk Enterprise, which comes with Python, and a second installation of Python on the same host, you can use the .path method to reference the second Python installation.

1. Use Splunk Web or edit inputs.conf and specify a scripted input stanza with a script name that ends in .path.

   [script://myfile.path]
   disabled = 0

2. Place the file that you reference in the stanza in the appropriate directory, as described in Where to place the scripts for scripted inputs.

3. Edit the file to specify the script or executable you want.

   /path/to/myscript -arg1 arg -arg2 arg

Examples of scripted inputs with inputs.conf

Unix top command

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

1. Create a new application directory. This example uses scripts/.

   $ mkdir $SPLUNK_HOME/etc/apps/scripts

286
2. All scripts should be run out of a `bin/` directory inside your application directory.

   $ mkdir $SPLUNK_HOME/etc/apps/scripts/bin

3. This example uses a small shell script `top.sh`.

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

4. Make the script executable.

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

5. Test that the script works by running it via the shell.

   $SPLUNK_HOME/etc/apps/scripts/bin/top.sh
   The script should send one `top` output.

6. Add the script entry to `inputs.conf` in

   $SPLUNK_HOME/etc/apps/scripts/local/.

   [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

   **Note:** You might need to modify `props.conf`:

   ♦ By default Splunk Enterprise breaks the single `top` entry into multiple events.
   ♦ The easiest way to fix this problem is to tell the 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]
   BREAKONLYBEFORE = <stuff>
Since there is no timestamp in the top output, you must tell Splunk Enterprise to use the current time. Use props.conf and set the following:

```
DATETIME_CONFIG = CURRENT
```

**Reference an external script with the .path stanza**

The following example uses the special .path stanza setting to reference an external build of Python to run a script on your host.

1. Edit `inputs.conf`.

   ```
   [script://loglogs.path]
   disabled = 0
   ```

2. Place or create `loglogs.path` in `$SPLUNK_HOME/etc/system/bin`.

3. Edit `loglogs.path` to reference the external version of Python.

   ```
   /usr/bin/python logit.py --source /opt/files/my_files --target /opt/files/my_files/processed --logfile /opt/src/my_sources/logfiles
   ```

**Set interval attribute to cron schedule**

In the above example, you can also set the `interval` attribute to a "cron" schedule by specifying strings like the following:

- `0 * * * *`: Means run once an hour, at the top of the hour.

- `*/15 9-17 * * 1-5`: Means run every 15 minutes from 9 am until 5 pm, on Monday to Friday.

- `15,35,55 0-6,20-23 1 */2 *`: Means run at 15, 35, and 55 minutes after the hour, between midnight and 7 am and again between 8 pm and midnight, on the first of every even month (February, April, June and so on).

For more information about setting cron schedules, read CRONTAB(5) on the Crontab website.
Configure event processing

Overview of event processing

Events are records of activity that reside in log files and machine data. They are primarily what Splunk software indexes. Events provide information about the systems that produce the machine data. The term event data refers to the contents of a Splunk index.

Here is a sample event:

```
172.26.34.223 - - [01/Jul/2017:12:05:27 -0700] "GET /trade/app?action=logout HTTP/1.1" 200 2953
```

When Splunk software 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.
- Segments events.
- Dynamically assigns metadata to events, if specified.
- Anonymizes data, if specified.

For an overview of the indexing process, see the Indexing overview chapter of the Managing Indexers and Clusters manual.

Configure character set encoding

You can configure character set encoding for your data sources. Splunk software has built-in character set specifications to support internationalization of your deployment. Splunk software supports many languages (including some that do not use Universal Coded Character Set Transformation Format - 8-bit (UTF-8) encoding).

Splunk software attempts to apply UTF-8 encoding to your sources by default. If a source does not use UTF-8 encoding or is a non-ASCII file, Splunk software tries 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.
You can retrieve a list of the valid character encoding specifications by using the `iconv -l` command on most *nix systems. A port for `iconv` on Windows is available.

**Supported character sets**

Splunk software supports an extremely wide range of character sets, including such key ones as:

- UTF-8
- UTF-16LE
- Latin-1
- BIG5
- SHIFT-JIS

See "Comprehensive list of supported character sets" at the end of this topic for the exhaustive list.

Here is a short list of the main supported character sets and the languages they correspond to.

<table>
<thead>
<tr>
<th>Language</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>CP1256</td>
</tr>
<tr>
<td>Arabic</td>
<td>ISO-8859-6</td>
</tr>
<tr>
<td>Armenian</td>
<td>ARMSCII-8</td>
</tr>
<tr>
<td>Belarus</td>
<td>CP1251</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>ISO-8859-5</td>
</tr>
<tr>
<td>Czech</td>
<td>ISO-8859-2</td>
</tr>
<tr>
<td>Georgian</td>
<td>Georgian-Academy</td>
</tr>
<tr>
<td>Greek</td>
<td>ISO-8859-7</td>
</tr>
<tr>
<td>Hebrew</td>
<td>ISO-8859-8</td>
</tr>
<tr>
<td>Japanese</td>
<td>EUC-JP</td>
</tr>
<tr>
<td>Japanese</td>
<td>SHIFT-JIS</td>
</tr>
<tr>
<td>Korean</td>
<td>EUC-KR</td>
</tr>
<tr>
<td>Russian</td>
<td>CP1251</td>
</tr>
<tr>
<td>Russian</td>
<td>ISO-8859-5</td>
</tr>
</tbody>
</table>
Manually specify a character set

To manually specify a character set to apply to an input, set the CHARSET key in props.conf:

```
[spec]
CHARSET=<string>
```

For example, if you have a host that generates 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 software parses only 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 software can automatically detect languages and proper character sets using its sophisticated character set encoding algorithm.

To configure Splunk software to automatically detect the proper language and character set encoding for a particular input, set CHARSET=AUTO for the input in props.conf. For example, 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
```

Train Splunk software to recognize a character set

If you want to use a character set encoding that Splunk software does not recognize, train it to recognize the character set by adding a sample file to the
following path and restarting Splunk Enterprise:

$SPLUNK_HOME/etc/ngram-models/_<language>-<encoding>.txt

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

After the sample file is added to the specified path, Splunk software recognizes sources that use the new character set, and automatically converts them to UTF-8 format at index time.

If you have Splunk Cloud and want to add a character set encoding to your Splunk deployment, file a Support ticket.

Comprehensive list of supported character sets

The common character sets described earlier are a small subset of what the CHARSET attribute can support. Splunk software also supports a long list of character sets and aliases, identical to the list supported by the *nix iconv utility.

Note: Splunk software ignores punctuation and case when matching CHARSET, so, for example, "utf-8", "UTF-8", and "utf8" are all considered identical.

Here is the full list, with aliases indicated in parantheses:

- utf-16le (aka, UCS-2LE, UNICODELITTLE)
- utf-16be (aka, ISO-10646-UCS-2, UCS-2, CSUNICODE, UCS-2BE, UNICODE-1-1, UNICODEBIG, CSUNICODE11, UTF-16)
- utf-32le (aka, UCS-4LE)
- utf-32be (aka, ISO-10646-UCS-4, UCS-4, CSUCS4, UCS-4BE, UTF-32)
- utf-7 (aka, UNICODE-1-1-UTF-7, CSUNICODE11UTF7)
- c99 (aka, java)
- utf-ebcdic
- latin-1 (aka, CP819, IBM819, ISO-8859-1, ISO-IR-100, ISO_8859-1:1987, L1, CSISOLATIN1)
<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>latin-7 (aka, ISO-8859-13, ISO-IR-179, L7)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ISO-8859-11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>roman-8 (aka, HP-ROMAN8, R8, CSHPROMAN8)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>KOI8-R (aka, CSKOI8R)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>KOI8-U</td>
<td></td>
</tr>
<tr>
<td></td>
<td>KOI8-T</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GEORGIAN-ACADEMY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GEORGIAN-PS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARMSCEII-8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MACINTOSH (aka, MAC, MACROMAN, CSMACINTOSH) [Note: these MAC* charsets are for MacOS 9; OS/X uses unicode]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MACGREEK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MACCYRILLIC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MACUKRAINE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MACCENTRALEUROPE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MACTURKISH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MACCROATIAN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MACICELAND</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MACROMANIA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MACHEBREW</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MACTHAI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NEXTSTEP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CP850 (aka, 850, IBM850, CSPC850MULTILINGUAL)</td>
<td></td>
</tr>
</tbody>
</table>
• CP862 (aka, 862, IBM862, CSPC862LATINHEBREW)
• CP866 (aka, 866, IBM866, CSIBM866)
• CP874 (aka, WINDOWS-874)
• CP932
• CP936 (aka, MS936, WINDOWS-936)
• CP949 (aka, UHC)
• CP950
• CP1250 (aka, MS-EE, WINDOWS-1250)
• CP1251 (aka, MS-CYRL, WINDOWS-1251)
• CP1252 (aka, MS-ANSI, WINDOWS-1252)
• CP1253 (aka, MS-GREEK, WINDOWS-1253)
• CP1254 (aka, MS-TURK, WINDOWS-1254)
• CP1255 (aka, MS-HEBR, WINDOWS-1255)
• CP1256 (aka, MS-ARAB, WINDOWS-1256)
• CP1257 (aka, WINBALTRIM, WINDOWS-1257)
• CP1258 (aka, WINDOWS-1258)
• CP1361 (aka, JOHAB)
• BIG-5 (aka, BIG-FIVE, CN-BIG5, CSBIG5)
• BIG5-HKSCS(aka, BIG5-HKSCS:2001)
• CN-GB (aka, EUC-CN, EUCCN, GB2312, CSGB2312)
• EUC-JP (aka,
  EXTENDED_UNIX_CODE_PACKED_FORMAT_FOR_JAPANESE, 
  CSEUCPKDFMTJAPANESE)
• EUC-KR (aka, CSEUCKR)
• EUC-TW (aka, CSEUCTW)
• GB18030
• GBK
• GB_1988-80 (aka, ISO-IR-57, ISO646-CN, CSISO57GB1988, CN)
• HZ (aka, HZ-GBK-2312)
• GB_2312-80 (aka, CHINESE, ISO-IR-58, CSISO58GB231280)
• SHIFT-JIS (aka, MS_KANJI, SJIS, CSSHIFTJIS)
• ISO-IR-87 (aka, JIS0208 JIS_C6226-1983, JIS_X0208 JIS_X0208-1983, 
  JIS_X0208-1990, X0208, CSISO87JISX0208, ISO-IR-159, JIS_X0212, 
  JIS_X0212-1990, JIS_X0212.1990-0, X0212, CSISO159JISX02121990)
• ISO-IR-14 (aka, ISO646-JP, JIS_C6220-1969-RO, JP, 
  CSISO14JISC6220RO)
• JISX0201-1976 (aka, JIS_X0201, X0201, CSHALFWIDTHKATAKANA)
• ISO-IR-149 (aka, KOREAN, KSC_5601, KS_C_5601-1987, 
  KS_C_5601-1989, CSKSC56011987)
• VISCII (aka, VISCII1.1-1, CSVISCII)
• ISO-IR-166 (aka, TIS-620, TIS620-0, TIS620.2529-1, TIS620.2533-0, 
  TIS620.2533-1)

294
Configure event line breaking

Some events consist of more than one line. Splunk software handles most multiline events correctly by default. If you have multiline events that Splunk software doesn’t handle properly, you can configure the software to change its line breaking behavior.

How Splunk software determines event boundaries

Splunk software determines event boundaries in two steps:

1. Line breaking, which uses the LINE_BREAKER attribute regular expression value to split the incoming stream of bytes into separate lines. By default, the LINE_BREAKER is any sequence of newlines and carriage returns (that is, (`\r\n`)+).

2. Line merging, which only occurs when you configure the SHOULD_LINEMERGE setting to "true" (the default). This step uses all the other line merging settings (for example, BREAK_ONLY_BEFORE, BREAK_ONLY_BEFORE_DATE, MUST_BREAK_AFTER, etc.) to merge the previously separated lines into events.

If the second step does not run (because you set the SHOULD_LINEMERGE attribute to "false"), then the events are the individual lines that the LINE_BREAKER attribute determines. The first step is relatively efficient, while the second is relatively slow. Appropriate use of the LINE_BREAKER regular expression can produce the results you want in the first step. This is valuable if a significant amount of your data consists of multiline events.

How to configure event boundaries

Many event logs have a strict one-line-per-event format, but some do not. Splunk software can often recognize the event boundaries, but if event boundary recognition does not work properly, you can set custom rules in props.conf to establish event boundaries.

Edit props.conf to configure multiline events

1. Examine the event format.
2. Determine a pattern in the events to set as the start or end of an event.
3. Edit $SPLUNK_HOME/etc/system/local/props.conf.
4. In the props.conf file, set the necessary attributes to configure your data.
5. Save the file and close it.
6. Restart Splunk Enterprise to commit the changes.

There are two ways to handle multiline events:

**Break and reassemble the data stream into events**

This method usually simplifies the configuration process, as it gives you access to several attributes that you can use to define line-merging rules.

1. Specify a stanza in `props.conf` that represents the stream of data you want to break and reassemble into events.
2. In that stanza, use the `LINE_BREAKER` attribute to break the data stream into multiple lines.
3. Set the `SHOULD_LINEMERGE` attribute to true.
4. Set your line-merging attributes (`BREAK_ONLY_BEFORE`, etc.) to specify how to reassemble the lines into events.

If your data conforms well to the default `LINE_BREAKER` setting (any number of newlines and carriage returns), you don’t need to alter the `LINE_BREAKER` setting. Instead, set `SHOULD_LINEMERGE=true` and use the line-merging attributes to reassemble it.

**Break the data stream directly into real events with the LINE_BREAKER setting**

Using the `LINE_BREAKER` setting to define event boundaries might increase your indexing speed, but is somewhat more difficult to work with. If you find that indexing is slow and a significant amount of your data consists of multiline events, this method can provide significant improvement.

1. Specify a stanza in `props.conf` that represents the stream of data you want to break directly into events.
2. Configure the `LINE_BREAKER` setting with a regular expression that matches the boundary that you want to use to break up the raw data stream into events.
3. Set the `SHOULD_LINEMERGE` setting to false.

**Line breaking general attributes**

The following tables list the settings in the `props.conf` file that affect line breaking.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

296
TRUNCATE = <non-negative integer>

Change the default maximum line length (in bytes). Although this attribute is a byte measurement, Splunk rounds down line length when this attribute would otherwise land mid-character for multibyte characters.

Set to 0 if you never want truncation (very long lines are, however, often a sign of garbage data).

10000 bytes

LINE_BREAKER = <regular expression>

A regular expression that determines how the raw text stream gets broken into initial events, before any line merging takes place (if specified by the SHOULD_LINEMERGE attribute, described below).

The expression must contain a capturing group (a pair of parentheses that defines an identified subcomponent of the match.)

Wherever the expression matches, Splunk software considers the start of the first capturing group to be the end of the previous event, and considers the end of the first capturing group to be the start of the next event.

Splunk software discards the contents of the first capturing group. This content will not be present in any event, as Splunk software considers this text to come between lines.

(\r\n)+ (Splunk software breaks data into an event for each line, delimited by any number of carriage return (\r) or newline (\n) characters.)
You can realize a significant boost to processing speed when you use\LINE_BREAKER\ to delimit multiline events (as opposed to using\SHOULD_LINEMERGE\ to reassemble individual lines into multiline events). Consider using this method if a significant portion of your data consists of multiline events.

See the props.conf specification file for information on how to use\LINE_BREAKER\ with branched expressions and additional information.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>\LINE_BREAKER_LOOKBEHIND\</td>
<td>When there is leftover data from a previous raw chunk, \LINE_BREAKER_LOOKBEHIND\ indicates the number of characters before the end of the raw chunk (with the next chunk concatenated) that Splunk software applies the \LINE_BREAKER\ regular expression. You might want to increase this value from its default if you are dealing with especially large or multiline events.</td>
<td>100 characters</td>
</tr>
<tr>
<td>\SHOULD_LINEMERGE\</td>
<td>When set to true, Splunk software combines several input lines into a single event, with configuration based on the attributes described in the next section.</td>
<td>true</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>\BREAK_ONLY_BEFORE_DATE\ = [true</td>
<td>false]</td>
<td>When set to true, Splunk software creates a new event if it encounters a new line with a date.</td>
</tr>
<tr>
<td>\BREAK_ONLY_BEFORE\ = &lt;regular expression&gt;</td>
<td>When set, Splunk software creates a new event if it encounters a new line that matches the regular expression.</td>
<td>empty string</td>
</tr>
<tr>
<td>\MUST_BREAK_AFTER\ = &lt;regular expression&gt;</td>
<td>When set and the regular expression matches the current line, Splunk software always creates a new event for the next input line. Splunk software might still break before the current line if another rule matches.</td>
<td>empty string</td>
</tr>
<tr>
<td>\MUST_NOT_BREAK_AFTER\ = &lt;regular expression&gt;</td>
<td>When set and the current line matches the regular expression, Splunk software does not break on any subsequent lines until the \MUST_BREAK_AFTER\ expression matches.</td>
<td>empty string</td>
</tr>
<tr>
<td>\MUST_NOT_BREAK_BEFORE\ = &lt;regular expression&gt;</td>
<td>When set and the current line matches the regular expression, Splunk software does not break the last event before the current line.</td>
<td>empty string</td>
</tr>
<tr>
<td>\MAX_EVENTS\ = &lt;integer&gt;</td>
<td>Specifies the maximum number of input lines that Splunk software adds to any event. The software breaks the event after it reads the specified number of lines.</td>
<td>256 lines</td>
</tr>
</tbody>
</table>
### Examples of configuring event line breaking

#### Specify event breaks

```
[my_custom_sourcetype]
BREAK_ONLY_BEFORE = ^\d+\s*$
```

Assume that any line that consists of only digits is the start of a new event for any data whose source type is set to `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&Ticket=646A1DA4STF896EE&SessionTime=25368&ReturnUrl=http://www.clientabc.com,
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=p12345678&Acco
MANUser:
Version=1&Name=&Debit=&Credit=&AccessTime=&BillDay=&Status=&Language=&Country=&Email=&EmailNotify=&Pin=&PinPayment=&PinAmount=&PinPG=&PinPGRate=&PinMenu=,
""}}
```

To index this multiline 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_LINESMERGE = True
BREAK_ONLY_BEFORE = Path=
```

This code tells Splunk software to merge the lines of the event, and only break before the term `Path=`.
Multiline event line breaking and segmentation limitations

Splunk software applies line breaking and segmentation limitations to extremely large events:

- **Events over `MAX_EVENTS` lines.** If Splunk software encounters a multiline event that exceeds the number of lines that has been specified in `MAX_EVENTS`, it breaks the event at that limit, sets the `BREAK_ONLY_BEFORE_DATE` setting to false (if it is true), and then drops any `MUST_NOT_BREAK_BEFORE` or `MUST_NOT_BREAK_AFTER` rules. This can result in events not being line broken as you would expect. To work around the problem, you can raise the `MAX_EVENTS` setting, but you might get better results by changing the `SHOULD_LINEMERGE` setting to false and by specifying the event boundary with the `LINE_BREAKER` setting.

- **Lines over 10,000 bytes.** Splunk software 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. It still groups these lines into a single event.

- **Segmentation for events over 100,000 bytes.** In search results, Splunk Web displays the first 100,000 bytes of an event. Segments after those first 100,000 bytes of a very long line are still searchable, however.

- **Segmentation for events over 1,000 segments.** In search results, Splunk Web displays the first 1,000 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.

Answers

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

Configure event timestamps

This topic discusses how Splunk software handles timestamps.

Examine this sample event:

```
172.26.34.223 - - [01/Jul/2017:12:05:27 -0700] "GET /trade/app?action=logout HTTP/1.1" 200 2953
```

The time information in the event:
is a timestamp.

Splunk software 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 does not contain timestamp information, Splunk software attempts to assign a timestamp value to the event at index time.

In most cases, Splunk software extracts timestamps correctly, but there are situations where you might need to configure timestamp handling. For example, when dealing with some sources or with distributed deployments, you might need to reconfigure timestamp recognition and formatting.

See the "Configure timestamps" chapter of this manual for specific instructions on how to configure timestamps.

**Configure indexed field extraction**

This topic discusses what fields Splunk software extracts at index time and leads to the chapter on how to configure extraction.

Splunk software can extract the following fields at index time:

- Default fields
- Custom fields
- File header fields

Splunk software always extracts a set of default fields for each event. You can configure it to extract custom fields and, for some data, file header fields.

For more information on indexed field extraction, see the chapter Configure indexed field extraction in this manual.

**Anonymize data**

You might need to anonymize, or mask, sensitive personal information from the data that you index into Splunk Enterprise, such as credit card or Social Security numbers. You can anonymize parts of confidential fields in events to protect privacy while providing enough remaining data for use in event tracking. You can
configure Splunk Enterprise indexers or heavy forwarders to anonymize data as it arrives and before the software indexes it.

There are two ways to anonymize data with Splunk Enterprise:

- With a sed script. This method is easier to do, takes less time to configure, and is slightly faster, but has limits in how many times you can invoke it and what it can do. For instructions on this method, see Anonymize data with a sed script
- With a regular expression (regex) transform. This method takes longer to configure, but is easier to modify after the initial configuration and can be assigned to multiple data inputs more easily. For instructions on this method, see Anonymize data with a regular expression transform

To anonymize data with Splunk Cloud, you must configure a Splunk Enterprise instance as a heavy forwarder and anonymize the incoming data with that instance before sending it to Splunk Cloud. You can follow the instructions in this topic on the heavy forwarder.

**Key points to anonymizing data**

Before you can anonymize data, you must select a set of events to anonymize.

- You use props.conf to select the events to anonymize
- You then use props.conf to anonymize the events with a sed script
- Or, you use props.conf and transforms.conf to anonymize the events with a regular expression transform

**Select events to anonymize**

You can anonymize event data based on whether the data comes from a specific source or host, or is tagged with a specific source type. You must specify which method to use in props.conf. The stanza name that you specify in props.conf determines how Splunk Enterprise selects and processes events for anonymization.

- [host::<host>]] matches events that contain the specified host
- source::<source> matches events with the specified source
- <sourcetype> matches events with the specified source type. You must specify the source type in inputs.conf for this stanza type to work. This option is a Splunk best practice.
**Replace strings in events with SEDCMD**

You can use the SEDCMD method to replace strings or substitute characters. The syntax for a sed replace is:

```
SEDCMD-<class> = s/<regex>/<replacement>/flags
```

The SEDCMD command has the following components:

- **regex** is a Perl language regular expression
- **replacement** is a string to replace the regular expression match.
- **flags** can be either the letter g to replace all matches or a number to replace a specified match.

**Substitute characters in events with SEDCMD**

The syntax for a sed character substitution is:

```
SEDCMD-<class> = y/<string1>/<string2>
```

This substitutes each occurrence of the characters in string1 with the characters in string2.

**Use a regular expression transform with transforms.conf to anonymize events**

Each stanza in transforms.conf defines a transform class that you can reference from props.conf for a given source type, source, or host.

Transforms have several settings and variables that let you specify what changes and where, but the following are the most important:

- The **REGEX** setting specifies the regular expression that points to the string in the event that you want to anonymize
- The **FORMAT** setting specifies the masked values
- The **$1** variable represents the text of the event before the regular expression that represents the string in the event that you want to mask
- The **$2** variable represents the text of the event after the regular expression
- **DEST_KEY = _raw** says to write the value from FORMAT to the raw value in the log. This anonymizes the event.

The regular expression processor does not handle multiline events. In cases where events span multiple lines, specify that the event is multiline by placing
before the regular expression in transforms.conf.

**Anonymize data with a sed script**

You can anonymize data by using a `sed` script to replace or substitute strings in events.

*`Sed`* is a *nix utility that reads a file and modifies the input based on commands that you use within or arguments that you supply to the utility. Many *nix users use the utility for fast transformation of incoming because the utility is so versatile. Splunk Enterprise lets you use a *sed*-like syntax in `props.conf` to script the masking of your data.

**Prerequisites for anonymizing data with a sed script**

You must have the following to anonymize data with a `sed` script:

- Data that you want to anonymize
- An understanding of how regular expressions work. See regular-expressions.info for details on regular expressions
- An `inputs.conf` file with a configuration that tells Splunk Enterprise where this data is located
- A `props.conf` file that references the `sed` script that anonymizes the data

For example, if you have a log file called `accounts.log` that contains Social Security and credit card numbers:

```
...  
ss=123456789, cc=1234-5678-9012-3456
ss=123456790, cc=2234-5678-9012-3457
ss=123456791, cc=3234-5678-9012-3458
ss=123456792, cc=4234-5678-9012-3459
...  
```

And you want to mask the fields, so that they appear like this:

```
...  
ss=XXXXX6789, cc=XXXX-XXXX-XXXX-3456
ss=XXXXX6790, cc=XXXX-XXXX-XXXX-3457
ss=XXXXX6791, cc=XXXX-XXXX-XXXX-3458
ss=XXXXX6792, cc=XXXX-XXXX-XXXX-3459
...  
```

You can use `inputs.conf` and `props.conf` to change the data that comes in from `accounts.log` as Splunk Enterprise accesses it. These configuration files reside
Configure inputs.conf to use a sed script

In this example, you create the source type SSN-CC-Anon, and assign it to the data input for accounts.log. The transform that you create uses this source type to know what data to transform. While there are other options available for using SEDCMD to transform incoming data from a log file, best practice is to create a source type, then assign the transform to that source type in props.conf.

1. On the machine that runs Splunk Enterprise, create an inputs.conf file in the $SPLUNK_HOME/etc/system/local directory. If the file already exists, proceed to the next step.
2. Open $SPLUNK_HOME/etc/system/local/inputs.conf with a text editor.
3. Add the following stanza to reference accounts.log and assign a source type to the MyAppServer.log data.

   [monitor:///opt/appserver/logs/accounts.log]
   sourcetype = SSN-CC-Anon

4. Save the file and close it.

Define the sed script in props.conf

In this example, props.conf uses the SEDCMD setting to perform the transformation directly.

The "-Anon" clause after the "SEDCMD" stem can be any string that helps you identify what the transformation script does. The clause must exist because it and the SEDCMD stem form the class name for the script. The text after the = is the regular expression that invokes the transformation.

1. On the machine that runs Splunk Enterprise, create a props.conf in the
   $SPLUNK_HOME/etc/system/local directory. If the file already exists, proceed to the next step.
2. Open $SPLUNK_HOME/etc/system/local/props.conf with a text editor.
3. Add the following stanza to reference the transform that you created in inputs.conf to do the masking transformation.

   [SSN-CC-Anon]
   SEDCMD-Anon = s/ss=\d{5}(\d{4})/ss=xxxxxx\1/g
   s/cc=(\d{4}-){3}(\d{4})/cc=xxxx-xxxx-xxxx-\2/g

4. Save the file and close it.
5. Restart Splunk Enterprise.
Anonymize data with a regular expression transform

You can mask data by creating a transform. Transforms take incoming data and change it based on configurations you supply. In this case, the transformation is the replacement of portions of the data with characters that obscure the real, sensitive data, while retaining the original data format.

Prerequisites for anonymizing data with a regular expression transform

To mask sensitive data, you need the following items:

- Data that you want to anonymize
- An understanding of how regular expressions work.
- An inputs.conf file, with a configuration that tells Splunk Enterprise where this data is located
- A transforms.conf file that does the data masking
- A props.conf file that references the transforms.conf file for the data that you want to mask

For example, if you have an application server log file called MyAppServer.log that contains events like the following:

"2006-09-21, 02:57:11.58", 122, 11, "Path=/LoginUser
Query=CrmId=ClientABC&
ContentItemID=TotalAccess&SessionId=3A1785URH117BEA&Ticket=646A1DA4STF896EE&
SessionTime=25368&ReturnUrl=http://www.clientabc.com,
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=clientabcUserId=p12345678&AccountId=&AgentHost=man&
AgentId=man, MANUser:
Version=1&Name=&Debit=&Credit=&AccessTime=&BillDay=&Status=
&Language=&Country=&Email=&EmailNotify=&Pin=&PinPayment=&PinAmount=&PinPG=
&PinPGRate=&PinMenu=&", ""

And you want to change the data so that the "sessionID" and "Ticket" fields are masked, and the events appear as follows:

"2006-09-21, 02:57:11.58", 122, 11, "Path=/LoginUser
Query=CrmId=ClientABC&
ContentItemID=TotalAccess&SessionId=###########7BEA&Ticket=############96EE&
SessionTime=25368&ReturnUrl=http://www.clientabc.com,
Use the inputs.conf, props.conf, and transforms.conf files to change the data that comes in from MyAppServer.log as Splunk Enterprise accesses it. All of these configuration files reside in the $SPLUNK_HOME/etc/system/local/ directory.

**Configure inputs.conf**

In this example, you create the MyAppServer-Anon source type. The transform you create uses this source type to know what data to transform. There are other options for selecting the data to transform, that will be explained later in this topic.

1. On the machine that runs Splunk Enterprise, create an inputs.conf file in the $SPLUNK_HOME/etc/system/local directory. If the file already exists, proceed to the next step.
2. Open $SPLUNK_HOME/etc/system/local/inputs.conf with a text editor.
3. Add the following stanza to reference MyAppServer.log and assign a source type to the MyAppServer.log data.

```
[monitor:///opt/MyAppServer/logs/MyAppServer.log]
sourcetype = MyAppServer-Anon
```
4. Save the file and close it.

**Configure transforms.conf**

Transforms.conf is the file that Splunk Enterprise uses to perform the transformation of the data.

1. On the machine that runs Splunk Enterprise, create a transforms.conf file in the $SPLUNK_HOME/etc/system/local directory. If the file already exists, proceed to the next step.
2. Open $SPLUNK_HOME/etc/system/local/transforms.conf with a text editor.
3. Add the following text to define the transform that anonymizes the sessionID field so that only the last four characters in the field are exposed:

```
[session-anonymizer]
REGEX = (?m)^(.*)SessionId=\w+\w{4}[&"].*$
FORMAT = $1SessionId=########$2
DEST_KEY = _raw
```
4. Add the following text directly underneath the session-anonymizer stanza to define the transform for the Ticket field, similar to the sessionID field:
Configure props.conf

Props.conf specifies the transforms to use to anonymize your data. It references one or more transform classes that you define in a transforms.conf file.

In this example, session-anonymizer and ticket-anonymizer are the transform class names that you defined in the transforms.conf file.

1. On the machine that runs Splunk Enterprise, create a props.conf file in the $SPLUNK_HOME/etc/system/local directory. If the file already exists, proceed to the next step.
2. Open $SPLUNK_HOME/etc/system/local/props.conf with a text editor.
3. Add the following stanza to reference the transforms that you created in transforms.conf to do the masking transformation.

   [MyAppServer-Anon]
   TRANSFORMS-anonymize = session-anonymizer, ticket-anonymizer

4. Save the file and close it.
5. Restart Splunk Enterprise.

Example

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 stanza and settings to your props.conf:

   [source:.../abc.log]
   SEDCMD-abc = y/abc/ABC/

Splunk Enterprise substituted "A" for each "a", "B" for each "b", and "C" for each "c". When you search for source="*/abc.log", the lowercase letters "a", "b", and "c" do not appear in your data.

Caveats for anonymizing data
Restrictions for using the sed script to anonymize data

If you use the SEDCMD method to anonymize the data, the following restrictions apply:

- The SEDCMD script applies only to the _raw field at index time. With the regular expression transform, you can apply changes to other fields.
- You cannot use more than one SEDCMD type transformation for the same host, source, or source type in a single props.conf file.

Restrictions for using the regular expression transform to anonymize data

If you use the regular expression transform to anonymize data, the following restrictions apply, include the LOOKAHEAD setting when you define the transform and set it to a number that is larger than the largest expected event. Otherwise, anonymization could fail.

Splunk indexers do not parse structured data

When you forward structured data to an indexer, the indexer does not parse it, even if you configured props.conf on that indexer with the INDEXED_EXTRACTIONS setting. Forwarded data skips the following queues on the indexer, which precludes data parsing:

- parsing
- aggregation
- typing

The forwarded data must arrive at the indexer already parsed. To achieve this, you must set up props.conf on the forwarder that sends the data. This includes configuring the INDEXED_EXTRACTIONS setting and any other parsing, filtering, anonymizing, and routing rules.

Universal forwarders can parse structured data only. See Forward data extracted from structured data files.
Configure timestamps

How timestamp assignment works

Timestamp processing is a key step in event processing. Splunk software uses timestamps to:

- Correlate events by time
- Create the timeline histogram in Splunk Web
- Set time ranges for searches

Splunk software adds timestamps to events at index time. It oftentimes assigns timestamp values automatically by using information that it finds in the raw event data. If there is no explicit timestamp in an event, Splunk software attempts to assign a timestamp value through other means. For some data, you might need to help it learn how to recognize the timestamps.

Splunk software stores timestamp values in the _time field, in Coordinated Universal Time (UTC) format.

For more information on event processing, see the chapter in this manual called Configure event processing.

How Splunk software assigns timestamps

Splunk software uses the following precedence rules to assign timestamps to events:

1. It looks for a time or date in the event itself using an explicit `TIME_FORMAT`, if provided. You configure the `TIME_FORMAT` attribute in props.conf.
2. If no `TIME_FORMAT` was configured for the data, Splunk software attempts to automatically identify a time or date in the event itself. It uses the source type of the event (which includes `TIME_FORMAT` information) to try to find the timestamp.
3. If an event has a time and date, but not a year, Splunk software determines the year, as described in How Splunk software determines timestamps with no year, and builds the timestamp from that.
4. If no events in a source have a date, Splunk software tries to find a date in the source name or file name. Time of day is not identified in filenames. (This requires that the events have a time, even though they don't have a date.)
5. For file sources, if no date can be identified in the file name, Splunk software uses the file modification time.
6. As a last resort, Splunk software sets the timestamp to the current system time when indexing each event.

Splunk software can extract only dates from a source, not times. If you need to extract a time from a source, use a transform. See Create custom fields at index time.

**How Splunk software determines timestamps with no year**

If Splunk software discovers a timestamp within an event that does not have a year element, it uses the following logic to determine the year:

1. It identifies the current date by using either the date of the event it last parsed or the current clock time.
2. It then uses the year from that date as a base and runs the year through several tests:
   1. If the date in the new event is December 31 and the current date is January 1, it decrements the base year.
   2. If the date in the new event is January 1 and the current date is December 31, it increments the base year.
   3. If the date in the new event is February 29, it determines if the current date year is a leap year.
   4. If the current date year is a leap year, it uses that year as the base year. If it is not, it uses the previous leap year.
3. If none of the previous tests results in a successful base year determination, the software uses the following procedure to determine the year:
   1. It determines the day of the year of the new event by calculating the number of days from January 1.
   2. If the date information of the previous event is available, and the day of the year of that event is more than the day of the year of the new event plus 4, then it increments the base year.
   3. If the date information of the previous event is not available, and the day of the year of the new event is greater than the current day of the year plus 2, then it decrements the base year.
4. The software then assigns the base year to the timestamp for the event. The timestamp must still pass the time range check for the timestamp to be valid.
**Example 1**

If Splunk software encounters **26 Jun** in a new event on May 26, 2017, and it was not able to determine the year in the previous events:

1. Since it was not able to determine the year in the previous event, it sets a base year of **2017** as that is the year of the current date.
2. The December 31 and January 1 tests fail, as the date is neither December 31 nor January 1. The base year remains **2017**.
3. The leap year test fails, as the date is not February 29. The base year remains **2017**.
4. Splunk software calculates the day of the year for June 26 as Day **177**.
5. Since it could not determine the year in the previous event, it adds two to this number to arrive at **179**.
6. It then compares **179** to the day of the year of the current date, May 26 (2017) which is Day **147**.
7. Since 179 is greater than 147, the software decrements the year from **2017** to **2016**.
8. The software then builds the new timestamp: **26 Jun 2016**.
9. If the new timestamp falls within the time range that has been set, the software adds the timestamp to the event.

**Example 2**

If Splunk software encounters **10 Apr** in a new event on May 26, 2017, and it determined the year **2017** in previous events:

1. Since it determined the year in the previous event, it sets that year as the base year: **2017**.
2. The December 31 and January 1 tests fail, as the date is neither December 31 nor January 1. The base year remains **2017**.
3. The leap year test fails, as the date is not February 29. The base year remains **2017**.
4. Splunk software calculates the day of the year for April 10 as Day **100**.
5. Since the year information in the previous event was available, it adds four to this number to arrive at **104**.
6. It then compares **104** to the day of the year of the current date, May 26 (2017) which is Day **147**.
7. Since 104 is less than 147, the software increments the year from **2017** to **2018**.
8. The software then builds the new timestamp: **10 Apr 2018**.
9. By default, this new timestamp is not legal, since it falls outside the default **MAX_DAYS_HENCE** setting which limits valid timestamps to 2 days into the...
future. The software uses the current date of 26 May 2017 as the timestamp, and applies that timestamp to the event.

Configure timestamps

Most events do not require special timestamp handling. Splunk software automatically recognizes and extracts their timestamps. For some sources and distributed deployments, you might need to configure how timestamps are extracted, so that they format properly.

There are two ways to configure timestamp extraction:

- Use the "Set Sourcetype" page in Splunk Web to interactively adjust timestamps on sample data. Once you are happy with the results, save the changes to a new source type and then apply that source type to your data inputs. See The "Set Sourcetypes" page.
- Edit props.conf directly. See Configure timestamp recognition.

You can also configure timestamp extraction to:

- Apply time zone offsets. See Apply time zone offsets.
- Extract the correct timestamp from events with more than one timestamp. See Configure timestamp assignment for events with multiple timestamps.
- Improve indexing performance. See Tune timestamp recognition for better indexing performance.

Considerations when adding data from new inputs

If you index data from a new input and then discover that you need to adjust the timestamp extraction process, you must reindex that data after you make the configuration changes.

Consider previewing your data to prevent the need to reindex. Alternatively, you can test new data inputs in a test Splunk deployment (or in a separate index on the production Splunk instance) before adding data to your production instance. That way, you can delete and reindex until you get the results you want.

Configure timestamp recognition
Most events do not require special timestamp handling. Splunk software recognizes and extracts their timestamps correctly. However, with some sources and distributed deployments, you might need to configure how timestamps are extracted, to ensure they are formatted properly.

There are two ways to configure timestamp extraction:

- Use the "Set Sourcetype" page in Splunk Web to interactively adjust timestamps on sample data. Once you are happy with the results, you can save the changes to a new source type and then apply that source type to your data inputs. See The Set Sourcetype page.

- Edit props.conf directly, as explained in this topic.

If you have Splunk Enterprise and need to modify timestamp extraction, perform the configuration on your indexer machines or, if you forward data, use heavy forwarders and perform the configuration on the machines where the heavy forwarders run. If you have Splunk Cloud and need to modify timestamp extraction, use heavy forwarders and perform the configuration on the machines where the heavy forwarders run.

The timestamp processor

The timestamp processor resides at $SPLUNK_HOME/etc/datetime.xml by default. You do not need to touch this file normally, unless you work with unusual, custom timestamps. If you need to configure timestamp recognition in some way, you can make the necessary changes by setting props.conf timestamp attributes, as described later in this topic.

If you have a custom timestamp that cannot be handled by configuring props.conf, substitute your own timestamp processor with the DATETIME_CONFIG attribute. This attribute specifies the file to be used by Splunk software for timestamp processing.

Edit timestamp properties in props.conf

To configure how Splunk software recognizes timestamps, edit props.conf. There are a number of attributes that pertain to timestamps. In particular, you can determine how Splunk software recognizes a timestamp by using the TIME_FORMAT attribute to specify a strptime() format for the timestamp.

You can also set other attributes that pertain to timestamps. This includes specifying where to look in an event for a timestamp, what time zone to use, or
how to deal with timestamps of varying currency.

Edit the props.conf file in $SPLUNK_HOME/etc/system/local/ or in your own custom application directory in $SPLUNK_HOME/etc/apps/. For information on configuration files in general, see About configuration files in the Admin manual.

To set timestamp recognition, configure one or more of the timestamp attributes in props.conf. Refer to the props.conf specification file for detailed information regarding these and other attributes.

**Syntax overview**

An overview of the syntax for the timestamp attributes follows:

```
[<spec>]
DATETIME_CONFIG = <filename relative to $SPLUNK_HOME>
TIME_PREFIX = <regular expression>
MAX_TIMESTAMP LOOKAHEAD = <integer>
TIME_FORMAT = <strptime-style format>
TZ = <POSIX time zone string>
MAX_DAYS_AGO = <integer>
MAX_DAYS_HENCE = <integer>
MAX_DIFF_SECS_AGO = <integer>
MAX_DIFF_SECS_HENCE = <integer>
```

In this syntax, `<spec>` can be:

- `<sourcetype>`, the source type 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.

If an event contains data that matches the value of `<spec>`, then the timestamp rules specified in the stanza apply to that event. You can have multiple stanzas, to handle different `<spec>` values.

**Timestamp validity attributes and their impact on events**

By default, all events are indexed, unless you specifically filter out events through other means.

When you set the `MAX_DAYS_AGO, MAX_DAYS_HENCE, MAX_DIFF_SECS_AGO, or MAX_DIFF_SECS_HENCE` attributes in a stanza, and event timestamps fall outside of the parameters of the attributes, then the software uses the following algorithm to determine the timestamp:
• The software uses the timestamp of the previous event to assign the
timestamp of the current event.
• If the timestamp of the previous event can't be determined, then the
software uses the current index time to assign a timestamp to the event.

Events are not dropped if they fall outside of the parameters of these attributes.

**Timestamp attributes**

You can set the following timestamp attributes with `props.conf`.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATETIME_CONFIG = &lt;filename relative to $SPLUNK_HOME&gt;</td>
<td>Specify a file to use to configure the Splunk timestamp processor. Under normal circumstances, you do not need to create your own timestamp processor file or modify the default datetime.xml file. The other props.conf attributes, described in this topic, can usually tweak the timestamp recognition capability to meet your needs. However, if your data has a custom timestamp format, you might need to substitute your own version of this file.</td>
<td>$SPLUNK_HOME/etc/datetime.xml</td>
</tr>
</tbody>
</table>

Set `DATETIME_CONFIG = NONE` to prevent the timestamp processor from running. When timestamp processing is off, Splunk software does not look at the text of the event for the timestamp—it instead uses the event’s "time of receipt"; in other words, the time the event is received via its input. For file-based inputs, the event timestamp is taken from the modification time of the input file.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TIME_PREFIX = &lt;regular expression&gt;</strong></td>
<td>When set, Splunk software uses the specified regular expression to looks for a match before attempting to extract a timestamp. The timestamp algorithm only looks for a timestamp in the event text that follows the end of the first regular expression match. You should use a regular expression that points exactly before your event's timestamp. For example, if the timestamp follows the phrase abc123 in your events, you should set TIME_PREFIX to abc123. If the TIME_PREFIX cannot be found in the event text, timestamp extraction does not take place.</td>
<td>empty string</td>
</tr>
<tr>
<td><strong>MAX_TIMESTAMP_LOOKAHEAD = &lt;integer&gt;</strong></td>
<td>Specify how far (how many characters) into an event Splunk software should look for a timestamp.</td>
<td>128 characters</td>
</tr>
</tbody>
</table>
This constraint is applied starting from the location positioned by TIME_PREFIX.

For example, if TIME_PREFIX positions a location 11 characters into the event, and MAX_TIMESTAMP_LOOKAHEAD is set to 10, timestamp extraction will be constrained to characters 11 through 20.

If set to 0 or -1, the length constraint for timestamp recognition is effectively disabled. This can have negative performance implications which scale with the length of input lines (or with event size when LINE_BREAKER is redefined for event splitting).

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME_FORMAT = <code>&lt;strptime-style format&gt;</code></td>
<td>Specifies a <code>strptime()</code> format string to extract the timestamp.</td>
<td>empty string</td>
</tr>
</tbody>
</table>

`strptime()` is a Unix standard for designating time formats. For more information, see the section Enhanced `strptime()` support, below.

TIME_FORMAT starts reading after the TIME_PREFIX (or directly at the start of the event, if there is no TIME_PREFIX attribute). If you use a TIME_PREFIX, it must match up to and including the character before the timestamp begins. If you don't set TIME_PREFIX but you do set TIME_FORMAT, the timestamp must appear at the very start of each event; otherwise, Splunk software
will not be able to process the formatting instructions, and every event will contain a warning about the inability to use strftime. (It's possible that you will still end up with a valid timestamp, based on how Splunk software attempts to recover from the problem.)

For best results, the `<strftime-style format>` should describe the day of the year and the time of day.

If `<strftime-style format>` contains an hour component, but no minute component, `TIME_FORMAT` ignores the hour component. It treats the format as an anomaly and considers the precision to be date-only.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>TZ = <code>&lt;timezone_identifier&gt;</code></td>
<td>The time zone of an event is determined as follows:</td>
<td>empty string</td>
</tr>
<tr>
<td></td>
<td>• If the event has a time zone in its raw text (such as UTC or -08:00), use that.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Otherwise, if TZ is set to a valid time zone string, use that. Specify a time zone setting using a value from the zoneinfo TZ database.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• If an event that arrives at an indexer has originated from a forwarder, and both indexer and forwarder are version 6.0 or later, then use the time zone that the forwarder provides.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Otherwise, use the time zone of the system that runs</td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>splunkd.</td>
<td>For more details and examples, see Specify time zones of timestamps in this manual.</td>
<td></td>
</tr>
<tr>
<td>TZ_ALIAS = &lt;key=value&gt;[,&lt;key=value&gt;]...</td>
<td>Provides admin-level control over how time zone strings extracted from events are interpreted. For example, EST can mean Eastern (US) Standard Time or Eastern (Australian) Standard Time. There are many other three letter time zone acronyms with multiple expansions.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>There is no requirement to use TZ_ALIAS if the traditional Splunk default mappings for these values work as expected. For example, EST maps to the Eastern US by default.</td>
<td>not set</td>
</tr>
<tr>
<td></td>
<td>Has no effect on the TZ value. It affects only time zone strings from event text, either from any configured TIME_FORMAT or from pattern-based guess fallback.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The setting is a list of key=value pairs, separated by commas.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The key is matched against the text of the time zone specifier of the event, and the value is the time zone specifier to use when mapping the timestamp to UTC/GMT.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The value is another TZ specifier that expresses the desired offset.</td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>MAX_DAYS_AGO = &lt;integer&gt;</td>
<td>Specifies the maximum number of days in the past, from the current date, that an extracted date can be valid. For example, if MAX_DAYS_AGO = 10, Splunk software ignores dates older than 10 days from the current date and instead either uses the timestamp of the previous event, or uses the current index time of the event if it cannot determine a timestamp in the previous event. The maximum settable number of days in the past is 10951.</td>
<td>2000 days</td>
</tr>
<tr>
<td>Note: If you have data that more than 2000 days old, increase this setting.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAX_DAYS_HENCE = &lt;integer&gt;</td>
<td>Specifies the maximum number of days in the future from the current date that an extracted date can be valid. For example, if MAX_DAYS_HENCE = 3, the software ignores dates that are more than 3 days in the future and instead either uses the timestamp of the previous event, or uses the current index time of the event if it cannot determine a timestamp from the previous event. <strong>Note:</strong> False positives are less likely with a tighter window. Change this attribute with caution.</td>
<td>2 days</td>
</tr>
<tr>
<td>Example: TZ_ALIAS = EST=GMT+10:00 (See the props.conf example file in the Configuration File Reference for more examples).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>MAX_DIFF_SECS_AGO</td>
<td>If the event timestamp is more than <code>&lt;integer&gt;</code> seconds before the previous timestamp, Splunk software accepts it only if it has the same time format as the majority of timestamps from the source.</td>
<td>3600 seconds (1 hour)</td>
</tr>
<tr>
<td>MAX_DIFF_SECS_HENCE</td>
<td>If the event's timestamp is more than <code>&lt;integer&gt;</code> seconds after the previous timestamp, Splunk only accepts it if it has the same time format as the majority of timestamps from the source.</td>
<td>604800 seconds (1 week)</td>
</tr>
</tbody>
</table>

**Enhanced strftime() support**

Use the `TIME_FORMAT` attribute in props.conf to configure timestamp parsing. This attribute takes a `strftime()` format string, which it uses to extract the timestamp.
Splunk software 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. The additional formats include:

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%N</td>
<td>For GNU date-time nanoseconds. Specify any sub-second parsing by providing the width: %3N = milliseconds, %6N = microseconds, %9N = nanoseconds.</td>
</tr>
<tr>
<td>%Q,%q</td>
<td>For milliseconds, microseconds for Apache Tomcat. %Q and %q can format any time resolution if the width is specified.</td>
</tr>
<tr>
<td>%I</td>
<td>For hours on a 12-hour clock format. If %I appears after %S or %s (like &quot;%H:%M:%S.%l&quot;), it takes on the log4cpp meaning of milliseconds.</td>
</tr>
<tr>
<td>%+</td>
<td>For standard Unix date format timestamps.</td>
</tr>
<tr>
<td>%v</td>
<td>For BSD and OSX standard date format.</td>
</tr>
<tr>
<td>%Z</td>
<td>The time zone abbreviation (nothing if there is no time zone information.)</td>
</tr>
<tr>
<td>%z, %:z, %::z</td>
<td>The time zone offset designator in International Organization for Standardization (ISO) 8601 format (for example, -0800 for PST, +0000 for GMT, or nothing if the time zone cannot be determined.) Use %:z if the timestamp offset contains hours and minutes (for example, -08:00) and %::z if the timestamp offset contains hours, minutes, and seconds (for example, -08:00:00.)</td>
</tr>
<tr>
<td>%o</td>
<td>For AIX timestamp support (%o used as an alias for %Y).</td>
</tr>
<tr>
<td>%p</td>
<td>The locale’s equivalent of AM or PM. (Note: there may be none.)</td>
</tr>
<tr>
<td>%s</td>
<td>Epoch (10 digits)</td>
</tr>
</tbody>
</table>

**Note:** A strptime() expression that ends with a literal dot and subsecond specifier such as %Q, %q, %N treats the terminal dot and conversion specifier as optional. If the .subseconds portion is absent from the text, the timestamp is still extracted.

**strptime() format expression examples**

Here are some sample date formats, with the strptime() expressions that handle them:

<table>
<thead>
<tr>
<th>Date Format</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>1998-12-31</td>
<td>%Y-%m-%d</td>
</tr>
<tr>
<td>98-12-31</td>
<td>%y-%m-%d</td>
</tr>
<tr>
<td>1998 years, 312 days</td>
<td>%Y years, %j days</td>
</tr>
</tbody>
</table>

323
<table>
<thead>
<tr>
<th>Jan 24, 2003</th>
<th>%b %d, %Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 24, 2003</td>
<td>%B %d, %Y</td>
</tr>
<tr>
<td>1397477611.862</td>
<td>%s.%3N</td>
</tr>
</tbody>
</table>

**Note:** Splunk software does not recognize non-English month names in timestamps. If you have an app that writes 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, such as:

...FOR: 04/24/07 PAGE 01...

To extract that timestamp, add this stanza in `props.conf`:

```
[host::foo]
TIME_PREFIX = FOR:
TIME_FORMAT = %m/%d/%y
```

Another example that includes time zone information:

```
?Valid_Until=Thu Dec 31 17:59:59 GMT-06:00 2020
```

To extract the timestamp, add this to `props.conf`:

```
[host::bar]
TIME_PREFIX = Valid_Until=
TIME_FORMAT = %a %b %d %H:%M:%S %Z%z %Y
```

Your data might contain other information that is parsed as timestamps, for example:

...1989/12/31 16:00:00 Wed May 23 15:40:21 2007...

Splunk software 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.

For more information on extracting the correct timestamp from events containing multiple timestamps, see Configure timestamp assignment for events with multiple timestamps.

Configure timestamps for specific needs

You can use the attributes described in this topic to configure the timestamp extraction processor for some specialized purposes, such as:

- To apply time zone offsets.
- To pull the correct timestamp from events with more than one timestamp.
- To improve indexing performance.

Configure how timestamps appear in search results

You can use your browser locale setting to configure how Splunk Web displays timestamps in search results. For information on setting the browser locale, see User language and locale.

Reconfigure how timestamps appear in raw data

Even though Splunk software uses the browser locale to configure how timestamps appear in search results, the raw data still remains in its original format. You might want to change this so that the data format is standardized in both raw data and search results. Do this with props.conf and transforms.conf. Here is an example:

Assume the timestamp data in the raw event looks like this:

06/07/2011 10:26:11 PM

but you want it to look like this (to correspond with how it appears in search results):

07/06/2011 10:26:11 PM

This example shows briefly how you can use props.conf and transforms.conf to transform the timestamp in the raw event.
In transforms.conf, add this stanza:

```
[resortdate]
REGEX = ^(/d(2))/(/d(2))/(/d(4))\$([^/]+)
FORMAT = $2/$1/$3 $4
DEST_KEY = _raw
```

In props.conf, add this stanza, where `<spec>` qualifies your data:

```
[<spec>]
TRANSFORMS-sortdate = resortdate
```

**Answers**

Have questions? Visit Splunk Answers and see what questions and answers the Splunk community has around timestamp recognition and configuration.

---

**Configure timestamp assignment for events with multiple timestamps**

If an event contains more than one timestamp, you can specify which timestamp is to be used for indexing. This is especially useful when indexing events that contain syslog host-chaining data.

Configure positional timestamp extraction by editing props.conf. For general information on editing `props.conf` for timestamps, see Configure timestamp recognition. If you have Splunk Enterprise and need to modify timestamp extraction, perform the configuration on your indexer machines or, if forwarding data, use heavy forwarders and perform the configuration on the machines where the heavy forwarders run. If you have Splunk Cloud and need to modify timestamp extraction, use heavy forwarder and perform the configuration on the machines where the heavy forwarders run.

**Configure positional timestamp extraction**

To specify the position of the timestamp you want extracted, you add `TIME_PREFIX` and `MAX_TIMESTAMP_LOOKAHEAD` attributes to a `props.conf` stanza. By setting a regular expression for `TIME_PREFIX`, you specify the pattern of characters that indicates the point to start looking for the timestamp. Set a value for `MAX_TIMESTAMP_LOOKAHEAD` to specify how far into an event (past the
TIME_PREFIX location) to look for the timestamp. By constraining lookahead, you can improve both accuracy and performance.

When TIME_PREFIX is set, Splunk software scans the event text for a match to its regular expression before it tries to extract a timestamp. The timestamping algorithm only looks for a timestamp in the text following the end of the first regular expression match. So if TIME_PREFIX is set to abc123, only the text following the first occurrence of abc123 is used for timestamp extraction.

TIME_PREFIX also sets the start point for MAX_TIMESTAMP_LOOKAHEAD; the lookahead starts after the matched portion of text in the TIME_PREFIX regular expression. For example, if TIME_PREFIX matches text through the first 11 characters of the event and the timestamp you want to extract is always within the next 30 characters, you can set MAX_TIMESTAMP_LOOKAHEAD=30. Timestamp extraction would be limited to text starting with character 12 and ending with character 41.

Example

Say you have an event that looks like this:

1989/12/31 16:00:00 Wed May 23 15:40:21 2007 ERROR UserManager -
Exception thrown
Ignoring unsupported search for eventtype: /doc
sourcetype="access_combined"
NOT eventtypetag=bot

To identify the timestamp as the second string of time information, May 23 15:40:21 2007, configure props.conf like this:

[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 = 21

This configuration instructs Splunk software to locate events that match the first timestamp construction, but ignore that timestamp in favor of a timestamp that occurs within the following 21 characters (a number it gets from the MAX_TIMESTAMP_LOOKAHEAD attribute). Splunk software will find the second timestamp because it always occurs within that 21-character limit.

Note: Optimize the speed of timestamp extraction by setting the value of MAX_TIMESTAMP_LOOKAHEAD to look only as far into an event as you need for the
timestamp you want to extract. In this example, MAX_TIMESTAMP_LOOKAHEAD is optimized to look just 21 characters into the event past the regular expression value.

Configure advanced timestamp recognition with datetime.xml

Datetime.xml is a file that Splunk Enterprise uses to extract dates and timestamps from events as it indexes them. The file contains regular expressions that describe how Splunk Enterprise performs those extractions from the raw event data.

In nearly all cases, you do not need to make modifications to datetime.xml. When you configure timestamp recognition with props.conf, Splunk Enterprise uses datetime.xml to configure its timestamp processor and extract timestamps out of the events for your source, source type, or host. If the software is not able to process the timestamps in your event data, you can train Splunk Enterprise to extract the timestamps by making a custom version of datetime.xml.

Structure of datetime.xml

The datetime.xml file has the following structure:

- Code blocks that define individual elements of a time stamp, such as year, month, day, hour, and minute. The following example code block defines the regular expression that Splunk Enterprise uses to extract a literal month element (for example, Jan, Mar) out of event data:

```xml
<define name="_litmonth" extract="litmonth">
  <text><![CDATA[(?<!\d)\](?<!\d)]]></text>
</define>
```

- Code blocks that use other elements that have been defined within the file. The following example code block defines the _time element which extracts hours, minutes, seconds, subseconds, period of day, and time zone:

```xml
<define name="_time" extract="hour, minute, second, subsecond, ampm, zone">
  <text><![CDATA[(?<!\d)]]></text>
</define>
```
Extraction pattern code blocks that define the order in which to attempt extracting times and dates from incoming event data. The timePatterns block defines the order in which Splunk Enterprise attempts to extract a timestamp, and the datePatterns block defines how to extract dates.

Each definition code block has one or more <text> definitions that contain a regular expression that Splunk Enterprise uses to extract the timestamp element.

Create or modify a custom datetime.xml

In nearly all cases, you do not need to modify datetime.xml. Instead, configure props.conf for timestamp extractions. See Edit timestamp properties in props.conf for instructions.

If Splunk Enterprise does not extract dates and times properly with props.conf, you might need to modify or substitute datetime.xml with a custom version. You can use the splunk train CLI command to take a sample of the timestamp data and generate code that you can use to create a custom datetime.xml that extracts your timestamp.

The splunk train CLI command is deprecated, but is still available to help you create patterns for datetime.xml based on your sample timestamp data.

After you create a pattern file with splunk train, you can make a copy of the default datetime.xml file and add your modifications to it, or you can create a new datetime.xml that only contains your custom timestamp definitions.

Never make modifications directly to $SPLUNK_HOME/etc/datetime.xml. Splunk Enterprise overwrites this file any time you upgrade, and any errors in the file that occur as the result of your changes can cause serious, lasting problems with data ingestion for both your custom source type and all other source types on the instance. If you want to make changes to the default file, save a copy to $SPLUNK_HOME/etc/system/local and make the changes there.
1. Create a sample timestamp pattern file
2. Run the splunk train CLI command against the file
3. Use the output to create a custom datetime.xml file
4. Reference the custom datetime.xml file in your timestamp configuration

**Create a sample timestamp pattern file**

1. From a prompt or PowerShell window, create a text file.
2. Paste in the sample of your timestamp data into this file.
3. Save the file and close it.
4. Change to the $SPLUNK_HOME/bin directory.

**Run the splunk train CLI command against the file**

1. Change to the $SPLUNK_HOME/bin directory:
   ```bash
   cd $SPLUNK_HOME/bin
   ```
2. Run the splunk train CLI command:
   ```bash
   ./splunk train dates
   ```
3. After the software asks the action you want to perform, type in `l`, `l`, or `learn` to perform the "learn" action.
4. Enter the path to the file that contains the timestamp sample.
5. Splunk Enterprise displays the first line of your sample and prompts you to enter values that represent 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: 10Jul200721:
    23:06 action: sell 3583 MNAG @ 42
```

Enter timestamp values as: month, day, year, hour, minute, second, ampm, timezone.
Enter values for month, day, year, hour, minute, second, period of day (am/pm), and time zone.

6. If the values are sufficient, Splunk software displays the following to show it has remembered the pattern:

Learned pattern.

If you are satisfied that the timestamps formats have been learned, hit control-c.

Hit Ctrl-C if Splunk Enterprise has correctly learned the timestamp formats.

7. Splunk software displays text similar to the following:

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/local/datetime.xml", and work with that file.
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="mycustom_date" extract="day,litmonth,year,">
<text><![CDATA[[:d+:s\w+s\s{\d+}\s\w+\s{\d+}]]]></text>
</define>

<define name="mycustom_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 >

Review the pattern definitions in the output. If the definition for your timestamp sample looks the way that you want, quit the splunk train session by typing in Q, q, or quit. Otherwise, type in L, l, or learn again to attempt the training operation again.
Use the output to create a custom datetime.xml file

After you successfully train Splunk Enterprise to understand your custom timestamp, you must add the definition that `splunk train` generated to a custom version of `datetime.xml`.

You have two options to create this file:

- **Add your timestamp definitions to an existing datetime.xml.** This is the preferred method.
- **Create a new datetime.xml** that contains only your customized timestamp definitions. This option is better when the source type for your data is in a very strict format and Splunk Enterprise was incorrectly choosing a broader default format.

See "Examples of custom datetime.xml configuration" later in this topic.

1. Make a copy of `datetime.xml` in the `$SPLUNK_HOME` directory.

   ```bash
   cd $SPLUNK_HOME/etc
   cp datetime.xml system/local/
   ```

   Never make edits to `$SPLUNK_HOME/etc/datetime.xml`. Always make a copy of this file and add your custom timestamp patterns to the copy.

2. Open `$SPLUNK_HOME/etc/system/local/datetime.xml` for editing.

3. Copy the block of code that `splunk train` generated and that begins with `define name`, on its own line, into the file. This code block can go anywhere between the `<datetime>` and `<timePatterns>` entries. See "Examples of custom datetime.xml configuration" later in this topic for examples.

4. Within the `<timePatterns>` block, add a line that references the definition line you added earlier in the `datetime.xml` file.

5. Within the `<datePatterns>` code block, add the same line you added in the previous step.

6. Save the custom `datetime.xml` file and close it.

Reference the custom datetime.xml file in your timestamp configuration

After you build your custom `datetime.xml` file, you can reference it in `props.conf` to extract your custom timestamps. You can set a custom timestamp extraction pattern for any host, source, or source type.

1. In `$SPLUNK_HOME/etc/system/local`, create `props.conf` if it does not
already exist.

2. Open props.conf in $SPLUNK_HOME/etc/system/local for editing.

3. Add a stanza for the host, source, or source type that requires the custom timestamp extraction, if it does not already exist.

4. Within this stanza, add a DATETIME_CONFIG setting that points to the custom datetime.xml, relative to the $SPLUNK_HOME directory. For example:

   [mysourcetype]
   DATETIME_CONFIG = /etc/system/local/datetime.xml
   MAX_TIMESTAMP_LOOKAHEAD = 128
   MAX_DAYS_AGO = 28

5. Repeat the previous steps as necessary for other hosts, sources, or source types that require the custom extraction.

6. Save props.conf and close it.

7. Restart Splunk software.

8. Confirm that timestamps are being extracted properly for the events that match the host, source, or source type that contains the custom timestamp extraction pattern.

Examples of custom datetime.xml configuration

The following blocks of code are examples of how to properly configure a custom datetime.xml file.

If splunk train generated code like the following:

   <define name="mycustom_date" extract="day,litmonth,year,">
   <text><![CDATA[:d+\s\w+\s(\d+)\s(\w+)\s(\d+)]]></text>
   </define>

   <define name="mycustom_time" extract="hour,minute,second,ampm,">
   <text><![CDATA[(\d+:\d+:\d+\s(\w+)

Example 1a: Modification of existing datetime.xml

Then you would add these definition blocks to an existing datetime.xml in $SPLUNK_HOME/etc/system/local that you copied previously:

   <datetime>

   <define name="mycustom_date" extract="day,litmonth,year,">
   <text><![CDATA[:d+\s\w+\s(\d+)\s(\w+)\s(\d+)]]></text>
   </define>

   <define name="mycustom_time" extract="hour,minute,second,ampm,">
Example 1b: New datetime.xml with only your timestamp configuration

Or, you would create a new datetime.xml in $SPLUNK_HOME/etc/system/local, as follows:

```
<datetime>
  <define name="mycustom_date" extract="day,litmonth,year,">
    <![CDATA[::<d+>\s+\w+\s+::<d+>\s+::<d+>\s+::<d+>]]>
  </define>

  <define name="mycustom_time" extract="hour,minute,second,ampm,"
    <![CDATA[<::<d+>:::<d+>:::<d+>\s+::<d+>]]>
  </define>

  <timePatterns>
    <use name="mycustom_date"/>
  </timePatterns>
</datetime>
```
Example 2: Reference of new datetime.xml in props.conf for your custom source type

You would then reference the custom datetime.xml file in the configuration for your source type in props.conf, as follows:

```
$SPLUNK_HOME/etc/system/local/props.conf

[my_custom_sourcetype]
DATETIME_CONFIG=/etc/system/local/datetime.xml
SHOULD_LINEMERGE=false
NO_BINARY_CHECK=true
```

Specify time zones for timestamps

If you index data from different time zones, you can use time zone offsets to ensure that they correlate correctly when you search. You can configure time zones based on the host, source, or source type of an event.

Configure time zones in props.conf. For general information on editing props.conf for timestamps, see Configure timestamp recognition. If you have Splunk Enterprise and need to modify timestamp extraction, perform the configuration on your indexer machines or, if forwarding data, use heavy forwarders and perform the configuration on the machines where the heavy forwarders run. If you have Splunk Cloud and need to modify timestamp extraction, use heavy forwarder and perform the configuration on the machines where the heavy forwarders run.

How Splunk software determines time zones

To determine the time zone to assign to a timestamp, Splunk software uses the following logic:

- Use the time zone specified in raw event data (for example, PST, -0800), if present.
- Use the TZ attribute set in props.conf, if the event matches the host,
source, or source type that the stanza specifies.

- If the forwarder and the receiving indexer are version 6.0 or later, use the time zone that the forwarder provides.
- Use the time zone of the host that indexes the event.

**Note:** If you have Splunk Enterprise and you change the time zone setting of the host machine, you must restart Splunk Enterprise for the software to detect the change.

### Specify time zones in props.conf

To configure time zone settings, edit props.conf in

```bash
$SPLUNK_HOME/etc/system/local/  or in your own custom application directory in
$SPLUNK_HOME/etc/apps/.
```

For information on configuration files in general, see About configuration files in the Admin manual.

Configure time zones by adding a `TZ` attribute to the appropriate stanza in props.conf. The `TZ` attribute recognizes zoneinfo TZ IDs. See all the time zone TZ IDs in the list of tz database time zones. Inside the stanza for a host, source, or source type, set the `TZ` attribute to the TZ ID for the desired time zone. This should be the time zone of the events coming from that host, source, or sourcetype.

You do not configure the time zone for the indexer in Splunk Enterprise, but in the underlying operating system. As long as the time is set correctly on the host system of the indexer, the offsets to event time zones will be calculated correctly.

### Examples of time zone specification in props.conf

Following are some examples of how to specify time zones in props.conf.

In the first example, events come into the indexer from New York City (in the US/Eastern time zone) 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 time zone to be specified as US/Eastern and US/Pacific, respectively.

The first example sets the time zone to US/Eastern for any events coming from hosts whose names match the regular expression `nyc.*`:
[host::nyc*]
TZ = US/Eastern
The second example sets the time zone to US/Pacific for any events coming from sources in the path /mnt/ca/...:

[source::/mnt/ca/...]
TZ = US/Pacific

**zoneinfo (TZ) database**

The zoneinfo database is a publicly maintained database of time zone values.

- UNIX versions of Splunk software rely on a TZ database included with the UNIX distribution you're running on. Most UNIX distributions store the database in the directory: /usr/share/zoneinfo.
- Solaris versions of Splunk software store TZ information in this directory: /usr/share/lib/zoneinfo.
- Windows versions of Splunk software ship with a copy of the TZ database.

Refer to the list of tz database time zones for all permissible TZ values.

**Map timezone strings extracted from event data**

Use the TZ_ALIAS attribute in props.conf to change how Splunk software interprets the timezone acronym string occurring in event data. For example, "EST" means Eastern (US) Standard Time by default, but your event data might be using that value instead to designate Eastern (Australian) Standard Time. To change the meaning of "EST" to the latter, set the attribute like this:

```
TZ_ALIAS = EST=GMT+10:00
```

Then, when Splunk software encounters "EST" in event data, it will interpret it as "GMT+10:00", rather than the default of "GMT- 5:00".

As this example shows, you can map a timezone string to an existing string plus offset value. You can also just map one TZ string directly to another.

When mapping timezone strings, be sure to handle both summer and winter versions of the time zones. If mapping EST, also map EDT, for example - depending on whatever your local pairs are. Test your software to see what timezone strings it produces.

You can specify multiple mappings. The syntax for TZ_ALIAS is:
TZ_ALIAS = <key=value>[,<key=value>]...
For more information, including examples, see the props.conf specification and example file in the Configuration File Reference.

Set the time zone for a user's search results

When you add or edit users using Splunk authentication, you can set a user time zone. Search results for that user will appear in the specified time zone. This setting, however, does not change the actual event data, whose time zone is determined at index time. For information on setting this value, see Configure users with Splunk Web in the Securing Splunk manual.

Tune timestamp recognition for better indexing performance

To speed up indexing, you can use props.conf to adjust how far ahead into events the Splunk timestamp processor looks, or even turn off the timestamp processor altogether.

For general information on editing props.conf for timestamps, see Configure timestamp recognition. If you have Splunk Enterprise and need to modify timestamp extraction, perform the configuration on your indexer machines or, if forwarding data, use heavy forwarders and perform the configuration on the machines where the heavy forwarders run. If you have Splunk Cloud and need to modify timestamp extraction, use heavy forwarder and perform the configuration on the machines where the heavy forwarders run.

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 the MAX_TIMESTAMP_LOOKAHEAD attribute.

The default number of characters that the timestamp processor looks into an event is 128. You can set MAX_TIMESTAMP_LOOKAHEAD to a lower value to speed up indexing. You should particularly do this if the timestamps always occur in the first part of the event.

Example:
Look for timestamps in the first 20 characters of events coming from source foo.

[source::foo]
MAX_TIMESTAMP_LOOKAHEAD = 20
...

**Disable timestamp processor**

You can turn off the timestamp processor entirely to improve indexing performance. Turn off timestamp processing for events matching a specified host, source, or sourcetype by setting the DATETIME_CONFIG attribute to NONE. When DATETIME_CONFIG=NONE, Splunk software does not look at the text of the event for the timestamp. Instead, it uses the event “time of receipt”; in other words, the time the event is received from its input. For file-based inputs (such as monitor) this means that the timestamp comes from the modification time of the input file.

You can also increase indexing performance by setting DATETIME_CONFIG to CURRENT, which assigns the current system time to each event at the time of indexing.

**Example:**

This example turns off timestamp extraction for events that come from the source foo.

[source::foo]
DATETIME_CONFIG = NONE
...

**Note:** Both CURRENT and NONE disable timestamp identification, so the default event boundary detection (BREAK_ONLY_BEFORE_DATE = true) might not work as you expect. When you use these settings, specify SHOULD_LINEMERGE or the BREAK_ONLY_* and MUST_BREAK_* settings to control event merging.
Configure indexed field extraction

About indexed field extraction

When Splunk software indexes data, it parses the data stream into a series of events. As part of this process, it adds a number of fields to the event data. These fields include default fields that it adds automatically and any custom fields that you specify.

The process of adding fields to events is known as field extraction. There are two types of field extraction:

- **Indexed field extraction**, which was described briefly at the start of this topic and which forms the basis for this chapter. These fields are stored in the index and become part of the event data.

- **Search-time field extraction**, which takes place when you search through data. Splunk software creates those fields when compiling search results and does not store them in the index. See About fields in the Knowledge Manager Manual for information about this type of field extraction.

There are two types of indexed fields:

- **Default fields**, which Splunk software automatically adds to each event. See About default fields in this chapter.
- **Custom fields**, which you specify. See Create custom fields at index time in this manual.

**Note:** When working with fields, consider that most machine data either does not have structure or has structure that changes constantly. For this type of data, use search-time field extraction for maximum flexibility. Search-time field extraction is easy to modify after you define it.

Other types of data might exhibit a more fixed structure, or the structure might already be defined within the data or events in the file. You can configure Splunk software to read the structure of these kinds of files (such as comma-separated value files (CSV), tab-separated value files (TSV), pipe-separated value files, and JavaScript Object Notation (JSON) data sources) and map fields at index time. To learn how this works, see Extract data from files with headers in this manual.
About default fields (host, source, sourcetype, and more)

When Splunk software indexes data, it tags each event with a number of fields. These fields become part of the index event data. The fields that are added automatically are known as default fields.

Default fields serve a number of purposes:

- The default field **index** identifies the index in which the event is located.
- The default field **linecount** describes the number of lines the event contains.
- The default field **timestamp** specifies the time at which the event occurred.

Splunk software uses the values in some of the fields, particularly sourcetype, when indexing the data, in order to create events properly. Once the data has been indexed, you can use the default fields in your searches.

The complete list of default fields follows:

<table>
<thead>
<tr>
<th>Type of field</th>
<th>List of fields</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal fields</td>
<td>_raw, _time, _indextime, _cd</td>
<td>These fields contain information that Splunk software uses for its internal processes.</td>
</tr>
<tr>
<td>Basic default fields</td>
<td>host, index, linecount, punct, source, sourcetype, splunk_server, timestamp</td>
<td>These fields provide basic information about an event, such as where it originated, what kind of data it contains, what index it's located in, how many lines it contains, and when it occurred.</td>
</tr>
<tr>
<td>Default datetime fields</td>
<td>date_hour, date_mday, date_minute, date_month, date_second, date_wday, date_year, date_zone</td>
<td>These fields provide additional searchable granularity to event timestamps.</td>
</tr>
</tbody>
</table>

**Note:** Only events that have timestamp information in them as generated by their respective systems will have date_* fields. If an event has a date_* field, it represents the value of time/date directly from the event itself. If you have specified any timezone conversions or changed the value of the
<table>
<thead>
<tr>
<th>Type of field</th>
<th>List of fields</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time/date at indexing or input time (for example, by setting the timestamp to be the time at index or input time), these fields will not represent that.</td>
<td></td>
</tr>
</tbody>
</table>

For information about default fields from the search perspective, see Use default fields in the Knowledge Manager Manual.

You can also specify additional, custom fields for inclusion in the index. See Create custom fields at index-time in this chapter.

This topic focuses on three key default fields:

- **host**
- **source**
- **sourcetype**

**Defining host, source, and sourcetype**

The host, source, and sourcetype fields are defined as follows:

- **host** - An event 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 lets you locate data originating from a specific device. For more information on hosts, see About hosts.

- **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. The source type determines how your data is to be formatted. For more information on source types, see Why source types matter.

**Source vs sourcetype**

Source and source type are both default fields, but they are entirely different otherwise, and can be easily confused.
• The source is the name of the file, stream, or other input from which a particular event originates.

• The sourcetype determines how Splunk software processes the incoming data stream into individual events according to the nature of the data.

Events with the same source type can come from different sources, for example, if you monitor source=/var/log/messages and receive direct syslog input from udp:514. If you search sourcetype=linux_syslog, events from both of those sources are returned.

**Under what conditions should you override host and sourcetype assignment?**

Much of the time, Splunk software 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.

**Override host assignment**

You might want to change your default host assignment when:

- You load archive data in bulk that was originally generated from a different host and you want those events to have that host value.
- You forward data from a different host. (The forwarder assigns its host name 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 if it originated elsewhere.

For detailed information about hosts, see the chapter Configure host values.

**Override sourcetype assignment**

You might want to change your default sourcetype assignment when:

- Splunk software cannot automatically format the data properly, resulting in problems such as wrong timestamping or event linebreaking.
- 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 software automatically recognizes, or to simply rename source types.

**Assign default fields dynamically**

This feature lets you dynamically assign default fields, also known as "metadata", to events as they are being consumed by Splunk software. Use this feature to specify source type, host, or source dynamically for incoming data. This feature is useful mainly with scripted data -- either a **scripted input** or an existing file processed by a script.

Do not use dynamic metadata assignment with file monitoring (tail) inputs. For more information about file inputs, see *Monitor files and directories* in this manual.

**Note:** The modular inputs feature has superseded this ***SPLUNK*** header feature. If you need dynamically-generated values for host, source and sourcetype, consider writing a modular input.

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 available metadata fields are sourcetype, host, and source.

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>
```

1. Set `<metadata field>=<string>` to a valid metadata/value pair. You can specify multiple pairs. For example, `sourcetype=log4j host=swan`.

2. 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.
3. 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.

Create custom fields at index time

In general, you should try to extract your fields at search time. However, there are times when you might find reason to add custom indexed fields. For example, you might have a situation 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`.

As well, you might 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 might want to add `foo` to the list of fields extracted by Splunk at index time.

For more information see About fields in the Knowledge Manager manual.

If you have Splunk Cloud and want to define index-time field extractions, open a support ticket.

**Caution:** Do not add custom fields to the set of default fields that Splunk software automatically extracts and indexes at index time unless absolutely necessary. This includes fields 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 Managing Indexers and Clusters 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 in 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.

Where to put the configuration changes in a distributed environment

If you have a distributed search deployment, processing is split between search peers (indexers) and a search head. You must deploy the changes as follows:

- Deploy the props.conf and transforms.conf changes to each of the search peers.
- Deploy the fields.conf changes to the search head.

If you are employing heavy forwarders in front of your search peers, the props and transforms processing takes place on the forwarders, not the search peers. Therefore, you must deploy the props and transforms changes to the forwarders, not the search peers.

For details on Splunk Enterprise distributed components, read Scale your deployment with Splunk Enterprise components in the Distributed Deployment Manual.

For details on where you need to put configuration settings, read Configuration parameters and the data pipeline in the Admin Manual.

Field name syntax restrictions

You can assign field names as follows:

- Valid characters for field names are a-z, A-Z, 0-9, or _.
- Field names cannot begin with 0-9 or _. Splunk reserves leading underscores for its internal variables.
- Avoid assigning field names that match any of the default field names.
- Do not assign field names that contain international characters.
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):

```plaintext
[<unique_transform_stanza_name>]
REGEX = <regular_expression>
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>
```

Note the following:

- 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:

  ```plaintext
  REGEX = ([a-z]+)=(([a-z]+)
  FORMAT = $1::$2
  ```

  Not using FORMAT:

  ```plaintext
  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.

- **FORMAT** behaves differently depending on whether the extraction takes place at search time or index time.
  - 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 `<unique_transform_stanza_name>::$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")
    - If the REGEX does not have $n$ groups, the matching fails.
    - **FORMAT** defaults to `<unique_transform_stanza_name>::$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")
      - However you can also set up index-time field extractions that create concatenated fields:
        - `FORMAT = ipaddress::$1.$2.$3.$4` When you create concatenated fields with **FORMAT**, it's important to understand that $\$ is the only special character. It is treated as a prefix for regex capturing groups only if it is followed by a number and only if that number applies to an existing capturing group.

So if your regex has only one capturing group and its value is `bar`, then:

- `FORMAT = foo$1` would yield `foobar`
- `FORMAT = foo$bar` would yield `foobar`
- `FORMAT = foo$1234` would yield `foo$1234`
- `FORMAT = foo$1\$2` would yield `foobar\$2`

- **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 4096. You might want to increase your **LOOKAHEAD** value if you have events with line lengths longer than 4096 characters.
  - Specifically, if the text you need to match is past this number of characters, you will need to increase this value.
  - Be aware, however, that complex regexes can have very high costs when scanning larger text segments. The speed may fall off
quadratically or worse when using multiple greedy branches or lookaheads / lookbehinds.

**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 follow field name syntax restrictions. They can only contain ASCII characters (a-z, A-Z, 0-9 or _). International characters will not work.

*Link the new field to props.conf*

To *props.conf*, add the following lines:

```
[<spec>]
TRANSFORMS-<class> = <unique_stanza_name>
```

Note the following:

- **<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.
- **<class>** is a unique literal string that identifies the namespace of the field (key) you are extracting. **Note:** `<class>` values *do not* have to follow field name syntax restrictions (see above). You can use characters other than a-z, A-Z, and 0-9, and spaces are allowed.
- **<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-<class>`, 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
```

Note the following:

- `<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 on all affected components.

**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 might 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 field which ends with a backslash\\"
```

**When Splunk creates field names**

**Remember:** When Splunk creates field names, it applies field name syntax restrictions to them.

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 fields.

**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]
REGENCY = 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:
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]
TRANSFORMS-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 files with structured data

Many structured data files, such as comma-separated value (CSV) files and Internet Information Server (IIS) web server logs, have information in the file header that can be extracted as fields during indexing. You can configure Splunk Enterprise and the Splunk universal forwarder to automatically extract these values into fields that can be searched. For example, a CSV file starts with a header row that contains column headers for the values in subsequent rows:

```
host,status,message,"start date"
srv1.splunk.com,error,"No space left on device",2013-06-10T06:35:00srv2.splunk.com,ok,-,2013-06-11T06:00:00
```

Input types that the indexed field extraction feature supports

This feature works with the following input types:

- File-based inputs only (such as monitoring files, directories, or archives.)
- Inputs that use the oneshot input type (or through the "Upload" feature in Splunk Web.)

It does not work with modular inputs, network inputs, or any other type of input.

More information on source types and time stamps

- For information on how to set source types when importing structured data files, see The "Set source type" page.
- For information on how to adjust timestamps when previewing indexing results, see Adjust time stamps and event breaks.
- For more general information about configuration files, see About configuration files in the Admin manual.

Use Splunk Web to extract fields from structured data files

When you upload or monitor a structured data file, Splunk Web loads the "Set Source type" page. This page lets you preview how your data will be indexed. See The 'Set Source type' page.

1. From the Add Data page in Splunk Web, choose Upload or Monitor as the method that you want to add data.
2. Specify the structured data file that you want the software to monitor. Splunk Web loads the "Set Source type" page. It sets the source type of the data based on its interpretation of that data. For example, if you
upload a CSV file, it sets the source type to csv.

3. Review the events in the preview pane on the right side of the page. The events are formatted based on the current source type.

4. If the events appear to be formatted correctly, click "Next" to proceed to the "Modify input settings" page. Otherwise, configure event formatting by modifying the timestamp, event breaking, and delimited settings until the previewed events look the way that you want.

5. If you don't want to save the settings as a new source type, return to Step 4. Otherwise, click the Save As button to save the settings as a new source type.

6. In the dialog that appears, type in a name and description for the new source type.

7. Select the category for the source type by selecting the category you want from the "Category" drop-down.

8. Select the application context that the new source type should apply to by choosing from the entries in the "App" drop-down.

9. Click "Save" to save the source type.

10. Return to Step 4 to proceed to the "Modify input settings" page.

**Structured data files with large numbers of columns might not display all extracted fields in Splunk Search**

If you index a structured data file with a large number of columns (for example, a CSV file with 300 columns), you might experience a problem later where the Search app does not appear to return or display all of the fields for that file. While Splunk software has indexed all of the fields correctly, this anomaly occurs because of a configuration setting for how Splunk software extracts the fields at search time.

Before Splunk software displays fields in Splunk Web, it must first extract those fields by performing a search time field extraction. By default, the limit for the number of fields that can be extracted automatically at search time is 100. You can set this number higher by editing the limits.conf file in $SPLUNK_HOME/etc/system/local and changing the limit setting to a number that is higher than the number of columns in the structured data file.

```
[kv]
limit = 300
```

If you work with a lot of large CSV files, you might want to configure the setting to a number that reflects the largest number of columns you expect your structured data files to have.
Use configuration files to enable automatic header-based field extraction

You can also use a combination of inputs.conf and props.conf to extract fields from structured data files. Edit these files in `$SPLUNK_HOME/etc/system/local/` or in your own custom application directory in `$SPLUNK_HOME/etc/apps/<app_name>/local`. Inputs.conf specifies the files you want to monitor and the source type to be applied to the events they contain, and props.conf defines the source types themselves. If you have Splunk Enterprise, you can edit the settings on indexer machines or machines where you are running the Splunk universal forwarder. You must restart Splunk Enterprise for any changes that you make to `inputs.conf` and `props.conf` to take effect. If you have Splunk Cloud and want to configure the extraction of fields from structured data, use the Splunk universal forwarder.

Props.conf attributes for structured data

To configure field extraction for files that contain headers, modify the following attributes in props.conf. For additional attributes in props.conf, review the props.conf specification file.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDEXED_EXTRACTIONS =&lt;CSV</td>
<td>W3C</td>
<td>TSV</td>
</tr>
<tr>
<td><strong>Note:</strong> If you set INDEXED_EXTRACTIONS=JSON, check that you have not also set KV_MODE = json for the same source type, which would extract the JSON fields twice, at index time and again at search time.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PREAMBLE_REGEX</td>
<td>Some files contain preamble lines. This attribute contains a regular expression that Splunk software uses to ignore any matching lines.</td>
<td>n/a</td>
</tr>
<tr>
<td>FIELD_HEADER_REGEX</td>
<td>A regular expression that specifies a pattern for prefixed</td>
<td>n/a</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>FIELD_DELIMITER</td>
<td>Specifies which character delimits or separates fields in the monitored file or source. You can specify special characters in this attribute.</td>
<td>n/a</td>
</tr>
<tr>
<td>FIELD_QUOTE</td>
<td>Specifies the character to use for quotes in the specified file or source. You can specify special characters in this attribute.</td>
<td>n/a</td>
</tr>
<tr>
<td>HEADER_FIELD_SPECIAL_CHARACTERS</td>
<td>Specifies which special characters can appear in header fields. When not set, the Splunk software replaces all characters that are neither alphanumeric or a space with underscores. If this setting is configured, the Splunk software does not perform a special character replacement in header field names when the special character matches one that you specify. For example, if you configure this setting to <code>.</code>, this setting does not replace that character with underscores during CSV ingestion.</td>
<td>n/a</td>
</tr>
<tr>
<td>HEADER_FIELD_DELIMITER</td>
<td>Specifies which character delimits or separates field names in the header line. You</td>
<td>n/a</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>HEADER_FIELD_DELIMITER</td>
<td>Specifies which character is used for quotes around field names in the header line. You can specify special characters in this attribute. If HEADER_FIELD_DELIMITER is not specified, FIELD_DELIMITER applies to the header line.</td>
<td></td>
</tr>
<tr>
<td>HEADER_FIELD_QUOTE</td>
<td>Specifies which character is used for quotes around field names in the header line. You can specify special characters in this attribute. If HEADER_FIELD_QUOTE is not specified, FIELD_QUOTE applies to the header line.</td>
<td>n/a</td>
</tr>
<tr>
<td>HEADER_FIELD_LINE_NUMBER</td>
<td>Specifies the line number of the line within the file that contains the header fields. If set to 0, Splunk attempts to locate the header fields within the file automatically.</td>
<td>0</td>
</tr>
<tr>
<td>TIMESTAMP_FIELDS</td>
<td>Some CSV and structured files have their timestamp encompass multiple fields in the event separated by delimiters. This attribute tells Splunk software to specify all such fields which constitute the timestamp in a comma-separated fashion.</td>
<td>Splunk Enterprise tries to automatically extract the timestamp of the event.</td>
</tr>
<tr>
<td>FIELD_NAMES</td>
<td>Some CSV and structured files might have missing headers. This attribute specifies the header field names.</td>
<td>n/a</td>
</tr>
<tr>
<td>MISSING_VALUE_REGEX</td>
<td>If Splunk software finds data that matches the specified regular expression in the structured data file, it considers the value for the field in the row to be empty.</td>
<td>n/a</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Special characters or values are available for some attributes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You can use special characters or values such as spaces, vertical and horizontal tabs, and form feeds in some attributes. The following table lists these characters:

<table>
<thead>
<tr>
<th>Special value</th>
<th>Props.conf representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>form feed</td>
<td>\f</td>
</tr>
<tr>
<td>space</td>
<td>space or ' '</td>
</tr>
<tr>
<td>horizontal tab</td>
<td>\t or tab</td>
</tr>
<tr>
<td>vertical tab</td>
<td>\v</td>
</tr>
<tr>
<td>whitespace</td>
<td>whitespace</td>
</tr>
<tr>
<td>none</td>
<td>none or \0</td>
</tr>
<tr>
<td>file separator</td>
<td>fs or \034</td>
</tr>
<tr>
<td>group separator</td>
<td>gs or \035</td>
</tr>
<tr>
<td>record separator</td>
<td>rs or \036</td>
</tr>
<tr>
<td>unit separator</td>
<td>us or \037</td>
</tr>
</tbody>
</table>

You can use these special characters for the following attributes only:

- FIELD_DELIMITER
- FIELD_HEADER_REGEX
- FIELD_QUOTE

Edit configuration files to create and reference source types

To create and reference the new source types to extract files with headers:

1. Using a text editor, open the file props.conf in the appropriate location as described in Enable automatic header-based field extraction earlier in this topic. If the props.conf file does not exist, you must create it.
2. Define a new sourcetype by creating a stanza which tells Splunk Enterprise how to extract the file header and structured file data, using the attributes described above. You can define as many stanzas - and thus, as many sourcetypes - as you like in the file. For example:

```bash
[HeaderFieldsWithFewEmptyFieldNamesWithSpaceDelim]
FIELD_DELIMITER=,
HEADER_FIELD_DELIMITER=\s
FIELD_QUOTE="
```

3. Save the props.conf file and close it.
4. Create a file `inputs.conf` in the same directory, if it does not already exist.
5. Open the file for editing.
6. Add a stanza which represents the file or files that you want Splunk Enterprise to extract file header and structured data from. You can add as many stanzas as you wish for files or directories from which you want to extract header and structured data. For example:

```bash
[monitor:///opt/test/data/StructuredData/HeaderFieldsWithFewEmptyFieldNamesWithSpaceDelim.csv]
sourcetype=HeaderFieldsWithFewEmptyFieldNamesWithSpaceDelim
```

7. Save the inputs.conf file and close it.
8. Restart Splunk Enterprise or the universal forwarder for the changes to take effect.

**Forward data extracted from structured data files**

You can also forward fields extracted from a structured data file to a heavy forwarder or a universal forwarder.

To forward fields extracted from structured data files:

1. Configure the Splunk instance that monitors the files to forward data to a heavy forwarder or a universal forwarder.
2. Configure the receiving instance.
3. On the monitoring instance, configure `props.conf` and `inputs.conf` to properly handle event breaking and timestamps for your data. You can do this in one of two ways:

   · To use Splunk Web, follow the instructions in *Use Splunk Web to extract fields from structured data files* earlier in this topic.
   · To use configuration files, follow the instructions in *Edit configuration files to create and reference sourcetypes*.
earlier in this topic.

4. Optionally, if you need to transform this data in any way prior to indexing it, edit `transforms.conf`.

5. Restart the receiving instance.

6. Restart the monitoring instance.

7. On the receiving instance, use the Search app to confirm that the fields have been extracted from the structured data files and properly indexed.

Caveats to extracting fields from structured data files

_Splunk software does not parse structured data that has been forwarded to an indexer_

When you forward structured data to an indexer, it is not parsed when it arrives at the indexer, even if you have configured `props.conf` on that indexer with `INDEXED_EXTRACTIONS`. Forwarded data skips the following pipelines on the indexer, which precludes any parsing of that data on the indexer:

- parsing
- merging
- typing

The forwarded data must arrive at the indexer already parsed.

_Field extraction settings for forwarded structured data must be configured on the forwarder_

If you want to forward fields that you extract from structured data files to another Splunk instance, you must configure the `props.conf` settings that define the field extractions on the forwarder that sends the data. This includes configuration of `INDEXED_EXTRACTIONS` and any other parsing, filtering, anonymizing, and routing rules. Performing these actions on the instance that indexes the data will have no effect, as the forwarded data must arrive at the indexer already parsed.

When you use Splunk Web to modify event break and time stamp settings, it records all of the proposed changes as a stanza for `props.conf`. You can find those settings in the "Advanced" tab on the "Set Source type" page.

Use the "Copy to clipboard" link in the "Advanced" tab to copy the proposed changes to `props.conf` to the system clipboard. You can then paste this stanza into `props.conf` in a text editor on Splunk instances that monitor and forward similar files.
**Only header fields containing data are indexed**

When Splunk software extracts header fields from structured data files, it only extracts those fields where data is present in at least one row. If the header field has no data in any row, it is skipped (that is, not indexed). Take, for example, the following csv file:

```
header1,header2,header3,header4,header5
one,1,won,,111
two,2,too,,222
tree,3,thri,,333
four,4,fore,,444
five,5,faiv,,555
```

When Splunk software reads this file, it notes that the rows in the `header4` column are all empty, and does not index that header field or any of the rows in it. This means that neither `header4` nor any of the data in its row can be searched for in the index.

If, however, the `header4` field contains rows with empty strings (for example, ""), the field and all the rows underneath it are indexed.

**Take care when allowing special characters for header fields**

The `HEADER_FIELD_ACCEPTABLE_SPECIAL_CHARACTERS` setting is designed to manage situations where column headers have characters like . or ::. If you do not use this setting, the Splunk software replaces those characters with underscores during the ingestion process.

**No support for mid-file renaming of header fields**

Some software, such as Internet Information Server, supports the renaming of header fields in the middle of the file. Splunk software does not recognize changes such as this. If you attempt to index a file that has header fields renamed within the file, the renamed header field is not indexed.

**Example configuration and data files**

Following are an example inputs.conf and props.conf to give you an idea of how to use the file header extraction attributes.
To extract the data locally, edit inputs.conf and props.conf to define inputs and sourcetypes for the structured data files, and use the attributes described above to specify how to deal with the files. To forward this data to another Splunk instance, edit inputs.conf and props.conf on the forwarding instance, and props.conf on the receiving instance.

**Inputs.conf**

```plaintext
[monitor:///opt/test/data/StructuredData/CSVWithFewHeaderFieldsWithoutAnyValues.csv]
sourcetype=CSVWithFewHeaderFieldsWithoutAnyValues

[monitor:///opt/test/data/StructuredData/VeryLargeCSVFile.csv]
sourcetype=VeryLargeCSVFile

[monitor:///opt/test/data/StructuredData/UselessLongHeaderToBeIgnored.log]
sourcetype=UselessLongHeaderToBeIgnored

[monitor:///opt/test/data/StructuredData/HeaderFieldsWithFewEmptyFieldNamesWithSpaceDelim.csv]
sourcetype=HeaderFieldsWithFewEmptyFieldNamesWithSpaceDelim

[monitor:///opt/test/data/FieldHeaderRegex.log]
sourcetype=ExtractCorrectHeaders
```

**Props.conf**

```plaintext
[CSVWithFewHeaderFieldsWithoutAnyValues]
FIELD_DELIMITER=,

[VeryLargeCSVFile]
FIELD_DELIMITER=,

[UselessLongHeaderToBeIgnored]
HEADER_FIELD_LINE_NUMBER=35
TIMESTAMP_FIELDS=Date,Time,TimeZone
FIELD_DELIMITER=\s
FIELD_QUOTE=

[HeaderFieldsWithFewEmptyFieldNamesWithSpaceDelim]
FIELD_DELIMITER=,
HEADER_FIELD_DELIMITER=\s
FIELD_QUOTE=

[ExtractCorrectHeaders]
FIELD_HEADER_REGEX=Ignore_This_Stuff:\s(.*)
FIELD_DELIMITER=,
```

365
Sample files

The following are snippets of the files referenced in the above inputs.conf and props.conf examples, to give you an idea of what the files look like.

You might need to scroll right quite a bit to see all of the content.

CSVWithFewHeaderFieldsWithoutAnyValues.csv

```
vqmcallhistoryid,serialnumber,vqmavgjbenvdelay,vqmavgjbenvnegdelta,vqmavgjbenvposdelta,vqmbitrate,vqmburstcount,vqmburstle...t,vqmterminatedelaymaximum,vqmthroughputindex,vqmto,vqmunderrundiscardpkts,vqmvocoderclass,vqmvocodertype,created,modified
99152,CFG0730084,-3,-2,356,64000,1,280,14,14.29,36,3499,201000,BW163736844290611-173170743@10.253.143.13,2011-06-29
12:37:37.292,0,4.68,1.43,0.19,0,0,0,0,52,60,15,17,60,10,0,Loopback,0.48,48,46,0,30,1334,10,99.55,10008,9962,0,0,,,,,60961470...49,0,286,0,520,10054,87,87,89,93,9,79,12,12,6096147089,10.255.43.12,10010,706222942,10.253.136.231,25110,6096147095,eth0/1,2,0,54,80,80,18500,6096147089,48,1,0,2011-06-29 12:41:47.303
99154,CFG0730084,-3,-1,251,64000,4,195,9,20.52,28,3494,359000,BW163502270290611594566299@10.253.143.13,2011-06-29 12:35:02.324,0,2.88,1.11,3.44,0,0,0,0,40,40,26,24,50,10,0,Loopback,0.31,54,46,0,31,2455,10,99.8,17769,17732,0,0,,,,,6096147089,10.255.43.12,10000,536353626,10.253.136.231,25096,6096147095,eth0/1,2,0,48,60,70,30400,6096147089,54,1,0,2011-06-29 12:41:47.342
```

VeryLargeCSVFile.csv

```
IncidntNum,Category,Descript,DayOfWeek,Date,Time,PdDistrict,Resolution,Location,X,Y
030203898,FRAUD,"FORGERY, CREDIT CARD",Tuesday,02/18/2003,16:30,NORTHERN,NONE,2800 Block of VAN NESS AV,-122.424612993055,37.8014488257836
030203901,LARCENY/THEFT,GRAND THEFT PICKPOCKET,Tuesday,02/18/2003,16:05,NORTHERN,NONE,VAN NESS AV / MCALLISTER ST,-122.42025048261,37.7800745746105
030203923,DRUG/NARCOTIC,SALE OF BASE/ROCK COCAINE,Tuesday,02/18/2003,17:00,BAYVIEW,"ARREST, BOOKED",1600 Block of KIRKWOOD AV,-122.390718076188,37.7385560584619
030203923,OTHER OFFENSES,CONSPIRACY,Tuesday,02/18/2003,17:00,BAYVIEW,"ARREST, BOOKED",1600 Block of KIRKWOOD AV,-122.390718076188,37.7385560584619
030203923,OTHER OFFENSES,PROBATION VIOLATION,Tuesday,02/18/2003,17:00,BAYVIEW,"ARREST, BOOKED",1600 Block of KIRKWOOD AV,-122.390718076188,37.7385560584619
```

UselessLongHeaderToBeIgnored.log

```
************ Start Display Current Environment ************
WebSphere Platform 6.1 [ND 6.1.0.21 cf210844.13] running with process name sammys_cell_A\fsgwsi89Node_A\sammys_A_c01_s189_m06 and process id 17904
Detailed IFix information: ID: 6.1.0-WS-WASSDK-AixPPC32-FP0000021
BuildVrsn: null Desc: Software Developer Kit 6.1.0.21
```
Answers

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

Process events with ingest-time eval

An ingest-time eval is a type of transform that evaluates an expression at index-time. Ingest-time eval provides much of the same functionality provided by search-time eval. The primary difference is that an ingest-time eval processes event data before indexing occurs and new fields and values that result from the evaluation are sent to indexers.

For more information on search-time eval expressions, see Use the eval command and functions in the Search Manual.

You can use ingest-time eval expressions to create new fields and perform a wide range of operations on incoming data, including mathematical, statistical,
and cryptographic functions. See Evaluation functions in the *Search Reference*.

**Why use ingest-time eval?**

Ingest-time eval provides an alternative to ingest-time transformations that are difficult or impossible with regular expressions alone, such as normalizing metrics data. See Example of targeted log-to-metrics conversions in the *Metrics* manual.

Ingest-time eval also gives you more direct control over index-time fields. For example, you can use ingest-time eval to control exactly how an index-time field is stored in the *rawdata journal* of a Splunk Enterprise index. For more information, see How the indexer stores indexes in *Managing Indexers and Clusters of Indexers*.

**Ingest-time eval syntax and usage**

Ingest-time eval takes a similar format to the search-time `| eval` command. For more information, see `eval` in the *Search Reference*.

An ingest-time eval stanza in `transforms.conf` contains an `INGEST_EVAL` expression. For example:

```
[eval1]
INGEST_EVAL= field3=length (_raw) *2
```

You can also chain multiple comma-separated `INGEST_EVAL` expressions, for example:

```
[eval2]
INGEST_EVAL= field4=_time, field5=field4+1
```

For detailed usage information and examples of `INGEST_EVAL`, see `transforms.conf`.

Search-time calculated fields that use the `EVAL-fieldname` setting in `props.conf` are not available.

Data processing that occurs before indexing with ingest-time eval can impact performance.

**Configure an ingest-time eval transform**

You configure eval-based transforms the same way you configure other
index-time transforms, using a transforms.conf file that contains the transform stanza, in conjunction with a props.conf file that references it. You must also configure a fields.conf file on the search head to enable searching of newly indexed eval fields.

To process event data with ingest-time eval, configure the following files:

**Configure transforms.conf**

To configure transforms.conf for ingest-time eval, follow these steps:

1. Create a transforms.conf file in the $SPLUNK_HOME/etc/system/local directory.
2. Add an ingest-time eval stanza that specifies the INGEST_EVAL expression. For example, the following INGEST_EVAL expression creates a new field called eval_user and populates the field with the lowercase version of the values in the username field:

   ```
   [myeval]
   INGEST_EVAL = eval_user=lower(username)
   ```

**Configure props.conf**

To configure props.conf for ingest-time eval, follow these steps:

1. Create a props.conf in the $SPLUNK_HOME/etc/system/local directory.
2. Add a stanza that specifies the data you want to process, such as <sourcetype>, and references the ingest-time eval stanza in transforms.conf. For example:

   ```
   [mysourcetype]
   TRANSFORMS = myeval
   ```

   You can mix eval-based transforms and regex-based transforms in props.conf in any order. The order in which you list the transforms determines when the transforms run relative to other stanzas in transforms.conf. For example, `TRANSFORMS = eval1,regex1,eval2,regex2` runs four different transforms.conf stanzas in that specific order.

**Configure fields.conf**

To configure fields.conf to enable search of ingest-time eval fields, do the following:
1. On the search head, create a `fields.conf` file in the
   `$SPLUNK_HOME/etc/system/local` directory.

2. Add a stanza that references the newly indexed field created by the
   `INGEST_EVAL` expression, as follows:

   ```
   [eval_user]
   INDEXED = True
   ```

For more information on how to configure index-time transforms, see Define
additional indexed fields.

**Examples**

For basic and extended examples of eval expressions, see `eval` in the *Search
Reference*. 
Configure host values

About hosts

The **host** field value of an event is the name of the physical device from which the event originates. Because it is a **default field**, which means that Splunk software assigns a host to every event it indexes, you can use it to search for all events that have been generated by a particular host.

The host value is typically the hostname, IP address, or fully qualified domain name of the network host on which the event originated.

**How the host value is determined**

Splunk software assigns a host value to each event by examining settings in the following order and using the first host setting it encounters:

1. Any event-specific host assignment that you specify in `transforms.conf`.
2. The default host value for the input that created the event, if any.
3. The default host value for the Splunk indexer or forwarder that initially consumes the data.

An overview of these assignment methods and their use cases follows. Subsequent topics describe the methods in greater detail.

**The default host value**

If no other host rules are specified for a source, Splunk software assigns the host field a default value that applies to all data coming into the instance from any input. The default host value is the hostname or IP address of the Splunk indexer or forwarder initially consuming the data. When the Splunk instance runs on the server where the event occurred, this is correct and no manual intervention is required.

For more information, see **Set a default host for a Splunk instance** in this manual.
**The default host for a file or directory input**

If you run Splunk Enterprise on a central log archive, or you are working with files that are forwarded from other hosts in your environment, you might 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 software 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.

**Event-specific assignments**

Some situations require you to assign host values by examining the event data. For example, if you have a central log host sending events to your Splunk deployment, you might have several host servers that feed data to that main log server. To ensure that each event has the host value of its originating server, you need to use the event's data to determine the host value.

For more information, see Set host values based on event data in this manual.

**Handle incorrectly-assigned host values**

If your event data gets tagged with the wrong host value, don't worry. There are a number of ways to fix or work around the problem.

For details, see Change host values after indexing in this manual.

**Tag host values**

You can tag host values to aid in the execution of robust searches. Tags enable you to cluster groups of hosts into useful, searchable categories.

For details, see About tags and aliases in the Knowledge Manager manual.
Set a default host for a Splunk instance

An event 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 software 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 subsequent topics in this chapter), the default host value for an event is the hostname or IP address of the server running the Splunk instance (forwarder or indexer) consuming the event data. When the event originates on the server on which the Splunk instance is running, that host assignment is correct and there’s no need to change anything. However, if all your data is being forwarded from a different host or if you’re bulk-loading archive data, you might want to change the default host value for that data.

To set the default value of the host field, you can use Splunk Web or edit inputs.conf.

**Set the default host value using Splunk Web**

1. In Splunk Web, click 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 default value of the host field for all events coming into that Splunk instance. You can override the value for invidividual sources or events, as described later in this chapter.

**Set the default host value using inputs.conf**

The default host assignment is set in inputs.conf during installation. You can modify the host value by editing that file in $SPLUNK_HOME/etc/system/local/ or in your own custom application directory in $SPLUNK_HOME/etc/apps/.
The host assignment is specified in the [default] stanza.

This is the format of the default host assignment in inputs.conf:

```
[default]
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.

**Warning:** Do not put quotes around the `<string>` value: host=foo, not host="foo".

After editing inputs.conf, you must restart your Splunk instance to put your changes into effect.

**Note:** By default, the host attribute is set to the variable $decideOnStartup, which means that it’s set to the hostname of the machine splunkd is running on. The splunk daemon re-interprets the value each time it starts up.

### Override the default host value for data received from a specific input

If you are running Splunk Enterprise on a central log archive, or you are working with files forwarded from other hosts in your environment, you might 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 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 an file or directory input in this manual.

### Override the default host value using event data

Some situations require you to assign host values by examining the event data. For example, If you have a central log host sending events to your Splunk deployment, you might have several host servers feeding data to that main log server. To ensure that each event has the host value of its originating server, you
need to use the event's data to determine the host value.

For more information, see *Set host values based on event data* in this manual.

**Set a default host for a file or directory input**

You can set a host value for all data from a particular file or directory input. You can set the host statically or dynamically.

If you set the host value statically, the same host is assigned to every event received from a designated file or directory input.

If you set the host value dynamically, the host name is extracted from the source input using a regular expression or segment of the full directory path of the source.

You can also assign host values to events that come through a particular file or directory input based on their source or source type values (as well as other kinds of information). See *Set host values based on event data*.

At this time, you cannot enable the setting of default host values for network (TCP and UDP) or **scripted inputs**.

**Statically set the default host value**

This method applies a single default host value to each event that a specific file or directory input generates.

A static host value assignment only affects new events that a certain input generates. You cannot assign a default host value to data that has already been indexed. Instead, you must tag the host value to the existing events. See Define and manage tags in the *Knowledge Manager Manual*.

**Use Splunk Web**

You can define a host for a file or directory input whenever you add or edit an input of that type.

To set the default host when creating a new input, see *Set a default host for a new input*.  

375
1. Click **Settings > Data Inputs**.
2. Click **Files & Directories**.
3. On the Files & directories page, click the name of an existing input to update it.
4. In the **Host** section, select the "constant value" option from the **Set host** dropdown.
5. Enter the static host value for the input in the **Host field value** field.
6. Click **Save**.

**Set a default host for a new input**

The process to set a default host is different when you create a new input.

1. Click **Settings > Data Inputs**.
2. Click **Files & Directories**.
3. On the Files & directories page, click **New** to add an input.
4. Specify the file or directory that you want to monitor, and specify any whitelists or blacklists.
5. Click **Next**.
6. (Optional) Set the source type for your new input.

   **Note:** If you specified a directory, the "Set Sourcetype" page does not appear.
7. Click **Next**.
8. On the **Input Settings** page, in the **Host** section, click the **Constant Value** button.
9. In the **Host field value** field, enter the host name for the input.
10. Click **Review** to continue to the Review page.
11. Click **Submit** to create the input.

**Edit inputs.conf**

To specify a host value for a monitored file or directory input, edit inputs.conf to specify a host value for a monitored file or directory input. When you edit inputs.conf, set the **host** attribute in the stanza that defines the input. If you have Splunk Cloud, you configure this setting on the machines where you run the Splunk universal forwarder.

```
[monitor://<path>]
host = <your_host>
```

Edit inputs.conf in $SPLUNK_HOME/etc/system/local/ or in 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 data can I index? in this manual.

**Example of static host value assignment**

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 set the default host value**

This method dynamically extracts 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 extract this information and assign it to the host field.

For a primer on regular expression syntax and usage, see Regular-Expressions.info. You can test regular expressions by using them in searches with the rex search command and by using third-party tools for writing and testing regular expressions.

**Use Splunk Web**

1. Click **Settings > Data Inputs**.
2. Click **Files & Directories**.
3. On the Files & directories page, click the name of an existing input to update it.
4. In the **Host** section, select one of the following two options from the Set host dropdown.

   - **regex on path** - Choose this option if you want to extract the host name with a regular expression. Then 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. Then enter the segment number in the **Segment number** 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".

5. Click **Save**.

**Dynamically set a default host for a new input**

The process to set a default host dynamically is different when you create a new input.

1. Click **Settings > Data Inputs**.
2. Click **Files & Directories**.
3. On the Files & directories page, click **New** to add an input.
4. Specify the file or directory that you want to monitor, and specify any whitelists or blacklists.
5. Click **Next**.
6. (Optional) Set the source type for your new input. **Note:** If you specified a directory, the "Set Sourcetype" page does not appear.
7. Click **Next**.
8. On the **Input Settings** page, in the **Host** section, click either **Regular expression on path** or **Segment in path**.
9. If you chose **Regular expression on path**, enter a regular expression to be used to extract the hostname from the source path in the "Regular expression" field. Otherwise, enter the number for the source path segment to be used to determine the hostname in the "Segment Number" field.
10. Click **Review** to continue to the Review page.
11. Click **Submit** to create the input.

**Edit inputs.conf**

You can set up dynamic host extraction rules by configuring **inputs.conf**. For more information on configuration files in general, see About configuration files in the Admin manual.

**Set the event host with the host_regex attribute**

1. Edit **inputs.conf** in **$SPLUNK_HOME/etc/system/local/ or in your own custom application directory in $SPLUNK_HOME/etc/apps/**.
2. Use the **host_regex** attribute to override the host field with a value extracted through a regular expression.

```
[monitor://<path>]
host_regex = <your_regular_expression>
```
3. Save the inputs.conf file.
4. Restart the Splunk instance.

The regular expression extracts the host value from the filename of each input. The input uses the first capturing group of the regular expression as the host. If the regular expression fails to match, the input sets the default host attribute as the host.

**Set the event host with the host_segment attribute**

The host_segment value overrides the host field with a value that has been extracted from a segment in the path of your data source.

1. Edit inputs.conf in $SPLUNK_HOME/etc/system/local/ or in your own custom application directory in $SPLUNK_HOME/etc/apps/.
2. Add a host_segment attribute to a stanza to override the host field with a value that has been extracted from a segment in the path of your data source. 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, set host_segment as follows:

   [monitor://var/log/]
   host_segment = 3

3. Save the inputs.conf file.
4. Restart the Splunk instance.

**Examples of dynamic host assignment**

In this example, the regular expression assigns all events from /var/log/foo.log a host value of "foo":

   [monitor://var/log]
   host_regex = /var/log/(\w+)
This example assigns the host value to the third segment in the path apache/logs:

   [monitor://apache/logs/]
   host_segment = 3
Caveats to setting the `host_segment` attribute to extract a host name

There are some caveats to using the `host_segment` attribute in an `inputs.conf` stanza:

- You cannot simultaneously specify the `host_regex` and `host_segment` attributes in the same stanza.

- When you simultaneously specify a `host_segment` and `source` attribute in the same stanza, the behavior of the `host_segment` attribute changes:
  - If the value you specify for the `source` contains a `/` (forward slash), the host value is extracted based on the segment number you specify in `host_segment`.
  - If `source` does not contain a `/`, or you specify a `host_segment` value that is larger than the number of segments available in `source`, then Splunk software cannot extract the host value, and instead uses the name of the host that extracted the data. See the following examples:

Example 1: Host name is `server01`, source path is `/mnt/logs/server01`, `inputs.conf` contains:

```
[monitor://mnt/logs/]
host_segment = 3
```
**Resulting host value:** `server01`

Example 2: Host name is `server01`, source path is `/mnt/logs/server01`, `inputs.conf` contains:

```
[monitor://mnt/logs/server01]
source = /mnt/logs/server01
host_segment = 3
```
**Resulting host value:** `server01`

Example 3: Host name is `server02`, source path is `/mnt/logs/server02`, `inputs.conf` contains:

```
[monitor://mnt/logs/server02]
source = serverlogs
host_segment = 3
```
**Resulting host value:** `server02`
Set host values based on event data

You can configure Splunk software to assign host names to your events based on the data in those events. This topic shows you how to use event data to override default host assignments with props.conf, transforms.conf, and regular expressions.

For a primer on regular expression syntax and usage, see Regular-Expressions.info. You can test regular expressions by using them in searches with the rex search command. The Splunk community wiki also has a list of useful third-party tools for writing and testing regular expressions.

Configuration

To configure per-event overrides, you need to create two stanzas, one in transforms.conf and another in props.conf. Edit these files in $SPLUNK_HOME/etc/system/local/ or in your own custom application directory in $SPLUNK_HOME/etc/apps/. If you have Splunk Cloud, edit these settings on the machines where you run the Splunk universal forwarder. For more information about configuration files in general, see About configuration files in the Admin manual.

transforms.conf

Create a stanza in transforms.conf that follows this syntax:

```
[<unique_stanza_name>]
REGEX = <your_regex>
FORMAT = host::$1
DEST_KEY = MetaData:Host
```

Note the following:

- `<unique_stanza_name>` should reflect that it involves a host value. You'll use this name later in the props.conf stanza.
- `<your_regex>` is a regular expression that identifies where in the event you want to extract the host value.
- `FORMAT = host::$1` writes the REGEX value into the host:: field.

props.conf

Next, create a stanza in props.conf that references the transforms.conf stanza:
[<spec>]
TRANSFORMS=<class> = <unique_stanza_name>

Note the following:

- `<spec>` can be:
  - `<sourcetype>`, the source type 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 unique identifier that you want to give to your transform.
- `<unique_stanza_name>` is the name of the stanza you created in transforms.conf.

Example

Assume that you're starting with the following set of events from the houseness.log file. The host is in the third position ("fflanda", etc.).

41602046:53 accepted fflanda
41602050:29 accepted rhallen
41602052:17 accepted fflanda

First, create a new stanza in transforms.conf with a regular expression that extracts the host value:

[houseness]
DEST_KEY = MetaData:Host
REGEX = \s(\w*)$
FORMAT = host::$1

Next, reference your transforms.conf stanza in a props.conf stanza. For example:

[source:..../houseness.log]
TRANSFORMS-rhallen=houseness
SHOULD_LINEMERGE = false

The above stanza has the additional attribute/value pair SHOULD_LINEMERGE = false, to break events at each newline.

The events will now appear in search results like this:
Change host values after indexing

At some point after indexing, you might discover that the host value for some of your events is not correct. For example, you might be collecting some Web proxy logs into a directory directly on your Splunk Enterprise server and you add that directory as an input without remembering to override the value of the host field, which results in the host value being the same as your Splunk Enterprise host.

If something like that happens, here are your options, from easiest to hardest:

- Delete and reindex the data.
- Use a search to delete the specific events that have the incorrect host value, and reindex those events.
- Tag the incorrect host values, and use the tag to search.
- Set up a Comma-separated values (CSV) 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 CSV 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, deleting and reindexing gives you the best performance and is the easiest. If you cannot delete and reindex the data, then the last option provides the cleanest alternative.
Configure source types

Why source types matter

The source type is one of the default fields that Splunk software assigns to all incoming data. It tells Splunk software what kind of data you have, so that it can format the data intelligently during indexing. Source types also let you categorize your data for easier searching.

Source types determine how incoming data is formatted

Because the source type controls how Splunk software formats incoming data, it is important that you assign the correct source type to your data. That way, the indexed version of the data (the event data) looks the way you want, with appropriate timestamps and event breaks. This facilitates easier searching of the data later.

Splunk software comes with a large number of predefined source types. When consuming data, Splunk software will usually select the correct source type automatically. If your data is specialized, you might need to manually select a different predefined source type. If your data is unusual, you might need to create a new source type with customized event processing settings. And if your data source contains heterogeneous data, you might need to assign the source type on a per-event (rather than a per-source) basis.

Like any other field, you can also use the source type field to search event data, once the data has been indexed. You will use it a lot in your searches since the source type is a key way to categorize your data.

Typical source types

Any common data input format can be a source type. Most source types are log formats. For example, some common source types that Splunk software automatically recognizes include:

- 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, and ACS), usually via remote syslog to a central log host.
• **websphere_core**, a core file export from WebSphere.

For a complete list of predefined source types, see [List of pretrained source types](#) in this manual.

**Configure source types**

There are two basic types of configuration you can do with source types:

- Assign source types explicitly to your incoming data.
- Create new source types, either from scratch or by modifying an existing source type.

**Assign source types**

In most cases, Splunk software determines the best source type for your data and automatically assigns it to incoming events. In some cases, however, you might need to explicitly assign a source type to your data. You usually do this when defining the data input. For details on how to improve source type assignment, see:

- Override automatic source type assignment
- Override source types on a per-event basis
- Configure rule-based source type recognition
- Create source types
- Rename source types

Later in this topic, there is a section that explains how Splunk software assigns source types.

**Create new source types**

If none of the existing source types fits the needs of your data, create a new one.

Splunk Web lets you adjust source type settings to fit your data. In essence, it is a visual source type editor. See [The Set Sourcetype page](#).

If you have Splunk Enterprise, you can also create a new source type by directly editing props.conf and adding a source type stanza. See [Create source types](#). If you have Splunk Cloud, use Splunk Web to define source types.
Preview data to test and modify source types

Splunk Web lets you review the effects of applying a source type to an input. It lets you preview the resulting events without actually committing them to an index. You can also edit timestamp and event breaking settings interactively and then save the modifications as a new source type. For information on how data preview functions as a source type editor, see The Set Sourcetype page.

Search on source types

`sourcetype` is the name of the source type search 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).

How Splunk software assigns source types

Splunk software employs a variety of methods to assign source types to event data at index time. As it processes event data, Splunk software 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 software applies source type values to specific kinds of events, while assigning source type values to other events automatically.

The following list shows how Splunk software goes about determining the source type for a data input. Splunk software starts with the first method and then descends through the others as necessary, until it can determine the source type. The list also provides an overview on how you configure source type assignment for each level.

Explicit source type specification based on the data input

If Splunk software finds an explicit source type for the data input, it stops here.

You configure this in `inputs.conf` or Splunk Web. Here is the `inputs.conf` syntax for assigning source types to a file input:

```
[monitor://<path>]
```
source=<source>
You can also assign a source type when defining an input in Splunk Web. For information on doing this for file inputs, see Monitor files and directories with Splunk Web in this manual. The process is similar for network or other types of inputs.

For more information, see Specify source type for an input.

**Explicit source type specification based on the data source**

If Splunk software finds an explicit source type for the particular source, it stops here.

You configure this in props.conf, using this syntax:

```
[source::<source>]
sourcetype=<sourcetype>
```

For more information, see Specify source type for a source.

**Rule-based source type recognition**

Splunk software looks next for any rules you’ve created for source types.

You can create source type classification rules in `props.conf`:

```
[rule::<rule_name>]
sourcetype=<sourcetype>
MORE_THAN_[0-100] = <regex>
LESS_THAN_[0-100] = <regex>
```

For information about setting up source type recognition rules, see Configure rule-based source type recognition.

**Automatic source type matching**

Splunk software next attempts to use automatic source type recognition to match similar-looking files and assign a source type.

Splunk software calculates signatures for patterns in the first few thousand lines of any file or network input stream. These signatures identify things like repeating word patterns, punctuation patterns, line length, and so on. When Splunk software calculates a signature, it compares it to its set of signatures for known,
"pretrained" source types. If it identifies a match, it assigns that source type to the data.

See List of pretrained source types in this manual for a list of the source types that Splunk software can recognize out of the box.

**Delayed rule-based source type association**

If Splunk software hasn’t identified a source type by now, it looks for any delayed rules.

This works like rule-based associations. You create a delayedrule:: stanza in props.conf. This is a useful "catch-all" for source types, in case Splunk software missed any with intelligent matching (see above).

A good use of delayed rule associations is for generic versions of very specific source types that were 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.

Here is the syntax:

```
[delayedrule::$RULE_NAME]
source_type=$SOURCETYPE
MORE_THAN_[0-100] = $REGEX
LESS_THAN_[0-100] = $REGEX
```

For more information about setting up or removing delayed rules for source type recognition, see Configure rule-based source type recognition.

**Automatic source type learning**

If Splunk software is unable to assign a source type for the event using the preceding methods, it creates a new source type for the event signature (see step 4, above). Splunk software stores learned pattern information in sourcetypes.conf.

**Override automatic source type assignment**

Splunk software attempts to assign a source type to your data automatically. You
can specify what source type to assign. You can also configure Splunk software so that it assigns a source type based on either the data input or the data source.

For details on the precedence rules that Splunk software uses to assign source types to data, read How Splunk software assigns source types.

Overrides only work on file and directory monitoring inputs or files you have uploaded. You cannot override the source type on network inputs. Additionally, overrides only affect new data that arrives after you set up the override. To correct the source types of events that have already been indexed, create a tag for the source type instead.

This topic describes how to specify a source type based for data based on its input and source.

Specify source type for an input

You can assign the source type for data coming from a specific input, such as /var/log/. If you have Splunk Enterprise, you do this in Splunk Web or by editing the inputs.conf configuration file. If you have Splunk Cloud, use Splunk Web to define source types.

Note: While assigning source type by input seems like a simple way to handle things, it is not very granular--when you use it, Splunk software assigns the same source type to all data from an input, even if some of the data comes from different sources or hosts. To bypass automatic source type assignment in a more targeted manner, you can assign source types based on the source of the data, as described later in this topic.

Use Splunk Web

When you define a data input, you can set a source type value to be applied to all incoming data from that input. You can pick a source type from a list or enter your own source type value.

To select a source type for an input, change the source type settings for the data input type you want to add. For example, for file inputs:

1. Click Settings in the upper right-hand corner of Splunk Web.
2. In the Data section of the Settings pop-up, click Data Inputs.
3. Click Files & Directories.
4. Click the **New** button to add an input.

5. In the "Add Data" page, browse or enter the name of the file you want to monitor, then click "Next".

6. In the "Set Sourcetype" page, click the "Sourcetype" drop-down and choose from the list of pre-trained source types. Splunk Web updates the page to show how the data looks when it receives the new source type.

7. If you want to make changes to the source type, use the "Event Breaks", "Timestamp", and "Advanced" tabs to modify settings and refresh the data preview. See The **Set Sourcetype page** in this manual.

8. If you want to save the source type as a different name, click **Save As?** to open a dialog box to save the new source type. Otherwise, proceed to Step 10.

9. If you chose to save the source type, Splunk Web displays the "Save Sourcetype" dialog. Enter the name, description, category, and app that the source type should apply to. See **Save modifications as a new source type**.

10. Click "Next" to set the source type for the data and proceed to the **Input settings** page.

Splunk software now assigns your selected source type to all events it indexes for that input.

**Use the inputs.conf configuration file**

When you configure an input in inputs.conf, you can specify a source type for the input. Edit **inputs.conf** in `$SPLUNK_HOME/etc/system/local/` or in your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For information on configuration files in general, see About configuration files in the Admin manual.

To specify a source type, include a **sourcetype** attribute within the stanza for the input. For example:

```plaintext
[tcp://:9995]
connection_host=dns
sourcetype=log4j
source=tcp:9995
```

This example sets the source type to "log4j" for any events coming from your TCP input on port 9995.
Caution: Do not put quotes around the attribute value: sourcetype=log4j, not sourcetype="log4j".

Specify source type for a source

Use props.conf to override automated source type matching and explicitly assign a single source type to all data coming from a specific source.

Edit props.conf in $SPLUNK_HOME/etc/system/local/ or in your own custom application directory in $SPLUNK_HOME/etc/apps/. For information on configuration files in general, see About configuration files.

Note: If you forward data, and you want to assign a source type for a source, you must assign the source type in props.conf on the forwarder. If you do it in props.conf on the receiver, the override has no effect.

To override source type assignment, add a stanza for your source to props.conf. In the stanza, identify the source path, using regular expression (regex) syntax for flexibility if necessary. Then specify the source type by including a sourcetype attribute. For example:

[source:.../var/log/anaconda.log(\d+)?]
sourcetype=anaconda

This example sets the source type to "anaconda" for events from any sources containing the string /var/log/anaconda.log followed by any number of numeric characters.

Your stanza source path regular expressions (such as [source:.../web/....log]) should be as specific as possible. Avoid using a regex that ends in "...". For example, do not do this:

[source::/home/fflanda/...]
sourcetype=mytype

This is dangerous. It tells Splunk software to process any gzip files in /home/fflanda as "mytype" files rather than gzip files.

Instead, write:

[source::/home/fflanda/....log(\d+)?]
sourcetype=mytype
Configure rule-based source type recognition

You can use rule-based source type recognition to expand the range of source types that Splunk software recognizes. In props.conf, you create a rule:: stanza that associates a specific source type with a set of qualifying criteria. When consuming data, Splunk software assigns the specified source type to file inputs that meet the rule's qualifications.

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 software checks them during the source typing process. As it processes each set of incoming data, Splunk software uses several methods to determine source types:

- After checking for explicit source type definitions based on the data input or source, Splunk software looks at any rule:: stanzas defined in props.conf and tries to match source types to the data based on the classification rules specified in those stanzas.
- If Splunk software 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.
- If that method fails, Splunk software then checks any delayedrule:: stanzas in props.conf and tries to match the data to source types using the rules in those stanzas.

For details on the precedence rules that Splunk software uses to assign source types to data, read How Splunk software assigns source types.

For a primer on regular expression syntax and usage, see Regular-Expressions.info. You can test regular expressions by using them in searches with the rex search command and third-party tools for writing and testing regular expressions.

You can configure your system so that rule:: stanzas contain classification rules for specialized source types, while delayedrule:: stanzas contain classification rules for generic source types. That way, Splunk software applies the generic source types to broad ranges of events that haven’t qualified for more specialized source types. For example, you could use rule:: stanzas to catch data with specific syslog source types, such as sendmail_syslog or cisco_syslog, and then configure a delayedrule:: stanza to apply the generic syslog source type to any remaining syslog data.
Configuration

To set source typing rules, edit `props.conf` in `$SPLUNK_HOME/etc/system/local/` or in your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For information on configuration files in general, see About configuration files in the Admin 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 in the body of the stanza. After the source type declaration, list the source type assignment rules. These rules use one or more `MORE_THAN` and `LESS_THAN` statements to find patterns in the data that fit given regular expressions by specific percentages.

To create a rule, use this syntax:

```
[rule::<rule_name>] OR [delayedrule::<rule_name>]
sourcetype=<source_type>
MORE_THAN_[0-99] = <regex>
LESS_THAN_[1-100] = <regex>
```

You set a numerical value in the `MORE_THAN` and `LESS_THAN` attributes, corresponding to the percentage of input lines that must contain the string specified by the regular expression. For example, `MORE_THAN_80` means at least 80% of the lines must contain the associated expression. `LESS_THAN_20` means that less than 20% of the lines can contain the associated expression.

**Note:** Despite its nomenclature, the `MORE_THAN_` attribute actually means "more than or equal to". Similarly the `LESS_THAN_` attribute means "less than or equal to".

A rule can contain any number of `MORE_THAN` and/or `LESS_THAN` conditions. The rule's source type is assigned to a data file only if the data qualifies all the statements in the rule. For example, you could define a rule that assigns a specific source type to a file input only if more than 60% of the lines match one regular expression and less than 20% match another regular expression.

**Examples**

**Postfix syslog files**

```
# postfix_syslog sourcetype rule
[rule::postfix_syslog]
```
source type = 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 = [: ]\[012\]?\[0-9\]:\[0-5\]\[0-9\]

**List of pretrained source types**

Splunk software ships with a set of built-in source types that are known as "pretrained" source types.

Splunk software can automatically recognize and assign many of these pretrained source types to incoming data. Splunk software also includes some pretrained source types that it does not recognize automatically but that you can manually assign via Splunk Web or `inputs.conf`, using methods described in Override automatic source type assignment.

It is a good idea to use a pretrained source type if it matches your data, as Splunk software already knows how to properly index pretrained source types. However, if your data does not fit any pretrained source types, you can create your own source types, as described in Create source types. Splunk software can also index virtually any format of data even without custom properties.

For an introduction to source types, see Why source types matter.

**Automatically recognized source types**

<table>
<thead>
<tr>
<th>Source type name</th>
<th>Origin</th>
<th>Example</th>
</tr>
</thead>
</table>
| 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 -0700] "GET / HTTP/1.0" 200 0442 ""check_http/1.10 (nagios-
| access_combined_wcookie    | NCSA combined format http web server logs (can be generated by apache or other web servers), with | "66.249.66.102.1124471045570513" 99.0442 "-" "check_http/1.10 (nagios-
<p>|                           |                                                                        | &quot;<a href="http://www.splunk.org/index.php/default/en-US">http://www.splunk.org/index.php/default/en-US</a>; rv:1.7.8) Gecko/20050524 Fedora/1.0.4-4 Firefox/1.0.4&quot; |
|                           |                                                                        | &quot;61.3.110.148.1124404439914689&quot;                                                              |</p>
<table>
<thead>
<tr>
<th>Source type name</th>
<th>Origin</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>access_common</td>
<td>NCSA common format http web server logs (can be generated by apache or other web servers)</td>
<td>10.1.1.140 -- [16/May/2005:15:01:52 -0700] /themes/ComBeta/images/bullet.png</td>
</tr>
<tr>
<td>asterisk_event</td>
<td>Standard Asterisk event log (management events)</td>
<td>Aug 24 14:08:05 asterisk[14287]: Manager 'randy' logged on from 127.0.0.1</td>
</tr>
<tr>
<td>asterisk_messages</td>
<td>Standard Asterisk messages log (errors and warnings)</td>
<td>Aug 24 14:48:27 WARNING[14287]: Channel 'Zap/1-1' sent into invalid extension 's' in context 'default'</td>
</tr>
<tr>
<td>asterisk_queue</td>
<td>Standard Asterisk queue log</td>
<td>1124909007</td>
</tr>
<tr>
<td>cisco_syslog</td>
<td>Standard Cisco syslog produced by all Cisco network devices including PIX firewalls, routers, ACS, etc., usually via remote syslog to a central log host</td>
<td>Sep 14 10:51:11 stage-test.splunk 00:08:49: %PIX-2-106001: Inbound TCP connection denied from IP_addr/port flags TCP_flags on interface int_name Inbound TCP connection denied from 144.1.10.222/9876 to 10.0.253.252/6161 flags SYN on interface outside</td>
</tr>
<tr>
<td>db2_diag</td>
<td>Standard IBM DB2 database administrative and error log</td>
<td>2005-07-01-14.08.15.304000-420 127.0.0.1: 4760 PROC : db2fmp.exe INSTANCENAME: db2.04 Automatic Runstats: evaluation has finished on database TRADEDB</td>
</tr>
<tr>
<td>exim_main</td>
<td>Exim MTA mainlog</td>
<td>2005-08-19 09:02:43 1E69KN-0001u6-8E =&gt; <a href="mailto:support-notifications@splunk.com">support-notifications@splunk.com</a> R=send_to_relay T=remote_smtp H=mail.int.splunk.com [10.2.1.10]</td>
</tr>
<tr>
<td>linux_messages_syslog</td>
<td>Standard linux syslog (/var/log/messages on most platforms)</td>
<td>Aug 19 10:04:28 db1 sshd(pam_unix) by (uid=0)</td>
</tr>
<tr>
<td>linux_secure</td>
<td>(Red Hat, Debian, and equivalent distributions)</td>
<td>Aug 18 16:19:27 db1 sshd[29330]: Accepted publickey for root from ::ffff:10.2.1.5 port 40892 ssh2</td>
</tr>
<tr>
<td>Source type name</td>
<td>Origin</td>
<td>Examples</td>
</tr>
<tr>
<td>------------------</td>
<td>--------</td>
<td>----------</td>
</tr>
<tr>
<td>log4j</td>
<td>Log4j standard output produced by any J2EE server using log4j</td>
<td>2005-03-07 16:44:03,110 53223013 property...</td>
</tr>
<tr>
<td>mysqld</td>
<td>Standard MySQL query log; also matches the MySQL binary log following conversion to text</td>
<td>53 Query SELECT xar_dd_itemid, xar_dd_propid, xar_dd_value FROM xar_dynamic_data WHERE xar_dd_propid = 27 AND xar_dd_itemid = 2</td>
</tr>
<tr>
<td>postfix_syslog</td>
<td>Standard Postfix MTA log reported via the Unix/Linux syslog facility</td>
<td>Mar 1 00:01:43 avas postfix/smtpd[1822]: 0141A61A83: client=host76-117.pool80180.interbusiness.it[80.180.117.76] sendpipe</td>
</tr>
<tr>
<td>sendmail_syslog</td>
<td>Standard Sendmail MTA log reported via the Unix/Linux syslog facility</td>
<td>Aug 6 04:03:32 nmrj100 sendmail[5200]: q64F01Vr001110: to=root, ctladdr=root (0/0), delay=00:00:01, xdelay=00:00:00, mailer=relay, min=00026, relay=[101.0.0.1] [101.0.0.1], dsn=2.0.0, stat=Sent (v00F3HmX004301 Message accepted)</td>
</tr>
<tr>
<td>sugarcrm_log4php</td>
<td>Standard Sugarcrm activity log reported using the log4php utility</td>
<td>Fri Aug 5 12:39:55 2005,244 [28666] FATAL layout_utils - Unable to load the application list language file for the selected language (en_us) or the default language (en_us)</td>
</tr>
<tr>
<td>weblogic_stdout</td>
<td>Weblogic server log in the standard native BEA format</td>
<td>****&lt;Sep 26, 2005 7:27:24 PM MDT&gt; asiAdminServer &gt; ListenThread.Default &lt;&lt;WLS Kernel&gt;&gt; BEA-000372 HostName: 0.0.0.0, maps to multiple IP addresses:</td>
</tr>
<tr>
<td>wasphere_activity</td>
<td>Websphere activity log, also often referred to as the service log</td>
<td>ProcessId: 2580 ThreadId: 0000001c SourceId: com.ibm.ws.channel.framework.</td>
</tr>
<tr>
<td>wasphere_core</td>
<td>Corefile export from Websphere</td>
<td>NULLsubcomponent dump routine NULL==signal 0 received 1TIDATETIME Date:</td>
</tr>
<tr>
<td>Source type name</td>
<td>Origin</td>
<td>Examples</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>websphere_trlog_syserr</td>
<td>Standard Websphere system error log in the IBM native trlog format</td>
<td>[7/1/05 13:41:00:516 PDT] 000003ae SystemErr R at com.ibm.ws.http.channel. inbound.impl.HttpICLReadCallback.java(Compiled)</td>
</tr>
<tr>
<td>websphere_trlog_sysout</td>
<td>Standard Websphere system out log in the IBM native trlog format; similar to the log4j server log for Resin and Jboss, sample format as the system error log but containing lower severity and informational events</td>
<td>[7/1/05 13:44:28:172 PDT] 0000082d SystemOut O Fri Jul 01 13:44:28 PDT 2005 TradeStreamerMDB: 100 Trade stock update alerts messages (in s 1.0365270454545454 The current price is:393 old price = 15.47 new price = 21.50)</td>
</tr>
<tr>
<td>windows_snare_syslog</td>
<td>Standard windows event log reported through a 3rd party Intersect Alliance Snare agent to remote syslog on a Unix or Linux server</td>
<td>0050818050818 Sep 14 10:49:46 stage-test.splunk.com Windows_Host MSWinEventLog 0 Security 3030 Day Aug 24 00:16:29 2005 560 Security admin4 User Success Audit Test_Host Object Open: Object Server: Security Object Type: File Object Name: C:\Directory\secrets1.doc New Handle ID: 1220 Operation ID: {0,117792} Process Primary Domain: FLAME Primary Logon ID: (0x0,0x8F9F) Client User Name: Client Domain: - Client Logon ID: - Accesses SYNCHRONIZE ReadData (or ListDirectory) Privileges -Sep</td>
</tr>
</tbody>
</table>

Special source types

<table>
<thead>
<tr>
<th>Source type name</th>
<th>Origin</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>known_binary</td>
<td>The filename matches a pattern that is generally known to be a binary file, not a log file</td>
<td>mp3 files, images, .rdf, .dat, etc. This is intended to catch obvious non-text files</td>
</tr>
</tbody>
</table>

Pretrained source types

These are all the pretrained source types, including both those that are automatically recognized and those that are not.

<table>
<thead>
<tr>
<th>Category</th>
<th>Source type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application servers</td>
<td>log4j, log4php, weblogic_stdout, websphere_activity, websphere_core, websphere_trlog, catalina, ruby_on_rails</td>
</tr>
<tr>
<td>Databases</td>
<td>db2_diag, mysqlld, mysqlld_error, mysqlld_bin, mysql_slow</td>
</tr>
<tr>
<td>E-mail</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Source type(s)</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Operating systems</td>
<td>exim_main, exim_reject, postfix_syslog, sendmail_syslog, procmail, 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, rpmkgs</td>
</tr>
<tr>
<td>Metrics</td>
<td>collectd_http, metrics_csv, statsd</td>
</tr>
<tr>
<td>Network</td>
<td>novell_groupwise, tcp</td>
</tr>
<tr>
<td>Printers</td>
<td>cups_access, cups_error, spooler</td>
</tr>
<tr>
<td>Routers and firewalls</td>
<td>cisco_cdr, cisco:asa, cisco_syslog, clavister</td>
</tr>
<tr>
<td>VoIP</td>
<td>asterisk_cdr, asterisk_event, asterisk_messages, asterisk_queue</td>
</tr>
<tr>
<td>Webservers</td>
<td>access_combined, access_combined_wcookie, access_common, apache_error, iis?</td>
</tr>
<tr>
<td>Splunk</td>
<td>splunk_com_php_error, splunkd, splunkd_crash_log, splunkd_misc, splunkd_stderr, splunk-blocksignature, splunk_directory_monitor, splunk_directory_monitor_misc, splunk_search_history, splunkd_remote_searches, splunkd_access, splunkd_ui_access, splunk_web_access, splunk_web_service, splunkd_conf?, django_access, django_service, django_error, splunk_help, mongodb</td>
</tr>
<tr>
<td>Non-Log files</td>
<td>csv?, psv?, tsv?, _json?, json_no_timestamp, fs_notification, exchange?, generic_single_line</td>
</tr>
<tr>
<td>Miscellaneous / Other</td>
<td>snort, splunk_disk_objects?, splunk_resource_usage?, kvstore?</td>
</tr>
</tbody>
</table>

? These source types use the INDEXED_EXTRACTIONS attribute, which sets other attributes in props.conf to specific defaults, and requires special handling to forward to another Splunk instance. See Forward data extracted from structured data files.

Finding out how a pretrained source type is configured to work
To find out what configuration information Splunk software uses to index a given source type, you can invoke the `btool` utility to list out the properties. For more information on using `btool`, refer to Use `btool` to troubleshoot configurations in the Troubleshooting manual.

The following example shows how to list out the configuration for the `tcp` source type:

```bash
$ ./splunk 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
```

**Override source types on a per-event basis**

This topic shows you how to override source types on a per-event basis. You do this at parse-time, after Splunk software has made its initial assignment as described in Why source types matter.

To configure per-event overrides, you use `transforms.conf` in tandem with `props.conf`.

399
Since this type of override occurs at parse-time, it works only on an indexer or heavy forwarder, not on a universal forwarder. See Configuration parameters and the data pipeline in the Admin manual for more information on what configurations are available at different points in the input/parsing/indexing process.

For information about configuring basic (not per-event) source type overrides for event data that comes from specific inputs or that has a particular source, see Override automatic source type assignment in this manual.

**Configuration**

To configure per-event overrides, you need to create two stanzas, one in transforms.conf and another in props.conf. Edit these files in $SPLUNK_HOME/etc/system/local/ or in your own custom application directory in $SPLUNK_HOME/etc/apps/. For more information about configuration files in general, see About configuration files in the Admin manual.

*transforms.conf*

Create a stanza in transforms.conf that follows this syntax:

```
[<unique_stanza_name>]
REGEX = <your_regex>
FORMAT = sourcetype::<your_custom_sourcetype_value>
DEST_KEY = MetaData:Sourcetype
```

Note the following:

- `<unique_stanza_name>` should reflect that it involves a source type. 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 source type to (such as events carrying a particular hostname or other field value).
- `<your_custom_sourcetype_value>` is the source type 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 regular expressions by using them in searches with the rex search command and using third-party tools.
Next, create a stanza in props.conf that references the transforms.conf stanza:

```
[<spec>]
TRANSFORMS-<class> = <unique_stanza_name>
```

Note the following:

- `<spec>` can be:
  - `<sourcetype>`, the source type 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 unique identifier that you want to give to your transform.
- `<unique_stanza_name>` is the name of the stanza you created in transforms.conf.

**Example: Assign a source type to events from a single input but different hosts**

Let's say that you have a shared UDP input, "UDP514". Your Splunk deployment indexes a wide range of data from a number of hosts through this input. You've found that you need to apply a particular source type called "my_log" to data originating from three specific hosts (host1, host2, and host3) reaching your Splunk deployment through UDP514.

To start, you can use the regular expression that Splunk software 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\[\w\./-]{2,})?\]?:\[\w\[\w.-]{2,}\]\s
FORMAT = host::$1
DEST_KEY = MetaData:Host
```

You can easily modify this regular expression to only match events from the hostnames you want (in this example, host1, host2, and host3):

```
REGEX = :\d\d\s+(?:\d+\s+|(?:user|daemon|local.?)\./\w+\s+)*\[?(host1|host2|host3)[\w./-]*\]?:\[\w\[\w.-]{2,}\]\s
```

Now you can use the modified regular expression in a transform that applies the
my_log source type 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+)\*\[?(host1|host2|host3)[\w\.-]*?\]?\nFORMAT = sourcetype::my_log
DEST_KEY = MetaData:Sourcetype
```

Then you can specify that transform in a props.conf stanza that identifies the specific input for the events:

```
[source::udp:514]
TRANSFORMS-changesourcetype = set_sourcetype_my_log_for_some_hosts
```

### Create source types

You can create new source types in several ways:

- Use the "Set Sourcetype" page in Splunk Web as part of adding the data.
- Create a source type in the "Source types" management page, as described in Add source type.
- Edit the props.conf configuration file directly.

Although you can configure individual forwarders to create source types by editing the .conf files that reside on the forwarder machine, a best practice for creating source types is to use Splunk Web, to guarantee that source types are created consistently across your Splunk deployment.

### Set the source type in Splunk Web

The "Set Sourcetype" page in Splunk Web provides an easy way to view the effects of applying a source type to your data and to make adjustments to the source type settings as necessary. You can save your changes as a new source type, which you can then assign to data inputs.

The page lets you make the most common types of adjustments to timestamps and event breaks. For other modifications, it lets you edit the underlying props.conf file directly. As you change settings, you can immediately see the changes to the event data.

The page appears only when you specify or upload a single file. It does not appear when you specify any other type of source.

To learn more about the page, see The "Set Sourcetype" page in this manual.
Create a source type

You can use the "Source types" management page to create a new source type. See Add source type in this manual.

Edit props.conf

If you have Splunk Enterprise, you can create a new source type by editing props.conf and adding a new stanza. For detailed information on props.conf, read the props.conf specification in the Admin manual. For information on configuration files in general, see About configuration files in the Admin manual.

The following is an example of an entry in props.conf. This entry defines the access_combined source type and then assigns that source type to files that match the specified source. You can specify multiple files or directories in a source by using a regular expression.

```plaintext
[access_combined]
pulldown_type = true
maxDist = 28
MAX_TIMESTAMP_LOOKAHEAD = 128
REPORT-access = access-extractions
SHOULD_LINEMERGE = False
TIME_PREFIX = \[
category = Web
description = National Center for Supercomputing Applications (NCSA) combined fo
rmat HTTP web server logs (can be generated by apache or other web servers)

[source::/opt/weblogs/apache.log]
sourcetype = iis
```

To edit props.conf:

1. On the host where you want to create a source type, create
   $SPLUNK_HOME/etc/system/local/props.conf
   **Note:** You might need to create the local directory. If you use an app, go to the app directory in $SPLUNK_HOME/etc/apps.
2. With a text editor, open the props.conf file in
   $SPLUNK_HOME/etc/system/local.
3. Add a stanza for the new source type and specify any attributes that Splunk software should use when handling the source type.

   ```plaintext
   [my_sourcetype]
   ```
attribute1 = value
attribute2 = value

Note: See the props.conf specification for a list of attributes and how they should be used.

4. (Optional) If you know the name of the file (or files) to which the source type is to be applied, specify them in the [source::<source>] stanza:

    [my_sourcetype]
    attribute1 = value
    attribute2 = value
    <br>
    [source:.../my/logfile.log]
    sourcetype = my_sourcetype

5. Save the props.conf file.

6. Restart Splunk Enterprise. The new source types take effect after the restart completes.

Specify event breaks and time stamping

When you create a source type, there are some key attributes that you should specify:

- **Event breaks.** To learn how to use props.conf to specify event breaks, see Configure event linebreaking.
- **Timestamps.** To learn how to use props.conf to specify timestamps, see Configure timestamp recognition, as well as other topics in the "Configure timestamps" chapter of this manual.

There are also a number of additional settings that you can configure. See the props.conf specification for more information.

Manage source types

The "Source Types" page lets you create, edit, and delete source types. It loads when you select "Source types" from the Settings menu. While this page and the "Set Sourcetypes" page have similar names, it is different from that page.
The Source Types page displays all source types that have been configured on the instance. It shows the default source types provided by your Splunk deployment and any source types that you have added.

**Sort source types**

By default, the Source Types management page sorts source types alphabetically. You can change how the page sorts by clicking the header bar for the "Name", "Category", and "App" columns.

Each header bar (except for "Actions") acts as a toggle. Click once to sort in ascending order and click again to sort in descending order.

**Filter source types**

You can filter the number of source types you see in the Source Type management page.

- To see only the most common source types, click the **Show only popular** checkbox along the top of the page. Popular source types are the most common source types that customers use. They have a `pulldown_type` source type field value of 1. When the "Show only popular" checkbox is not selected, the page shows all source types that have been defined on the instance.

- To see only source types that belong to a certain category, click the **Category** drop-down and select the category you want. Only source types that belong to that category will display. To see all source types again, select "All" from the "Category" drop-down.
To see only source types that belong in a certain application context, click the **App** drop-down and select the application context that the source type applies to. Only source types that apply to that application context will display. To see all source types again, select "All" from the "App" drop-down.

To see only source types whose names contain a certain string, type that string in the **Filter** text box next to the "App" drop-down, then press Enter. Only source types whose names or descriptions match what you have typed in the "Filter" box display. To see all source types again, click the "x" button on the right side of the "Filter" text box.

By default, the Source Types management page shows up to 20 source types on a page. If you want to see more or less, click the **20 per page** link on the right side of the page and select the number of source types you want to see. Choices are 10, 20, 50, or 100 listings on a page.

**Modify source types**

To modify a source type, click its name in the list, or click its **Edit** link in the **Actions** column. The "Edit Source type" page appears.
The **Edit Source Type** dialog box lets you change the configuration of a source type. You can change the following:

**Description:** Type in the description of the source type in the "Description" field.

**Destination app:** The application context that the source type applies to.

**Note:** You cannot change the app destination for source types that come with Splunk software.

**Category:** The category that the source type is a member of. Click the button to select from the list of categories and choose the one you want. When you save, the source type appears in the category you selected.

The **Log to Metrics** source type category is specifically for converting log events to metric data points. See **About the Log to Metrics source type category**.

**Indexed Extractions:** A format for extracting fields at index time from files with structured data. Select the type of indexed extraction that best represents the contents of the file:

- none: The file does not contain structured data.
- json: The file is in JavaScript Object Notation (json) format.
- csv: The file has comma-separated values.
- tsv: The file has tab-separated values.
- psv: The file has pipe (|) separated values.
- w3c: The file conforms to the World Wide Web Consortium (W3C) logging format.
- field_extraction: The file contains unstructured events with fields in `<field>=<value>` format.

**Timestamp:**

The Timestamp section of the dialog controls how timestamps are determined for events from the source file.

- Auto: Use default logic to extract timestamp from event.
- Current Time: Use current system time
- Advanced: Use advanced internal logic to extract timestamp from event.

**Advanced time stamp extraction configurations**
The following advanced configurations are available when you select **Advanced** in the "Timestamp Extraction" section:

- **Time zone**: Specify the time zone to be assigned to the timestamps.
- **Timestamp format**: Specify the format of the time stamp in the source event. The available formats come from the properties of the `strptime()` programming function. For example, if the source file contains logs with timestamps in this format:

  6 Jun 2015 18:35:05

...specify the following in the "Timestamp format" field:

  `%d %b %Y %H:%M:%S`

Another example:

  Tue Jun 4 2:55:18 PM 2015

maps to

  `%a %b %d %I:%M:%S %p %Y`

For a list of the strings that you can use to define the time stamp format, see `strptime(3)` (http://linux.die.net/man/3/strptime) on the die.net Linux man page site.

**Timestamp prefix**: A regular expression that represents the characters that come before a time stamp. When Splunk software sees this set of characters in an event, it expects a time stamp to occur after that.

**Lookahead**: Specifies the maximum number of characters to scan into an event for the time stamp. If you specified a regular expression in the "Timestamp prefix" field, it looks no more than the number of characters specified past the string that the regular expression represents for the time stamp.

**Advanced**

The Advanced section of the dialog shows you all of the configurations for the source type, in key/value format. This represents what is in the `props.conf` file that defines the source type. You can edit each setting directly, or add and delete settings. To delete settings, click the "x" on the right side of each setting. To add an entry, click the "New setting" link at the bottom of the dialog. This exposes a key/value pair of fields. Enter the key name in the "Name" field and its value in
the "Value" field.

**Caution:** Use the "Advanced" section with care. Adding or changing values here can cause data to be incorrectly indexed.

**About the Log to Metrics source type category**

The **Log to Metrics** source type category is used for the ingest-time conversion of logs to metric data points. If you select it, a **Metrics** tab will appear. For more information about this source type and the fields in the **Metrics** tab, see Convert event logs to metric data points in **Metrics**.

**Add Source Type**

To create a new source type, click the **New Source Type** button at the top right of the screen. The **Create Source Type** dialog box opens.

This dialog is exactly the same as the "Edit Source Type" dialog. See **Modify source types** for information on the controls in the dialog.

When you have finished configuring the source type, click "Save."

**Delete Source Type**

To delete a source type, click the **Delete** link in the "Actions" column for the source type that you want to delete. You cannot delete built-in source types, only source types that you create or that come with apps.

When you delete a source type, the following dialog appears:

```
Delete Source Type
Are you sure you want to delete the source type named earthquakes.csv?
Cancel Delete
```

**Caution:** Deleting a source type has significant consequences, especially if the source type is already in use:

- Data can be indexed incorrectly after you delete the source type. Making the data searchable in the way you want later can take a lot of effort. Many apps and add ons use source types to look for data, and data indexed
under a missing source type is data those apps and add-ons do not see.

- Any configurations that the source type uses, such as field extractions, index time filtering, and time stamp formats, are irretrievably lost.
- You cannot undo a source type deletion. The only options available in this case are to restore the `props.conf` file that defines the source type from a backup, or recreate the source type manually.

If you are sure you want to delete the source type, click "Delete". The dialog closes and Splunk Web returns you to the Source Types management page.

**Rename source types at search time**

You might want to rename a source type in certain situations. For example, say you accidentally assigned an input to the wrong source type. Or you realize that two differently named source types should be handled exactly the same at search time.

If you have Splunk Enterprise, you can use the `rename` attribute in `props.conf` to assign events to a new source type at search time. In case you ever need to search on it, the original source type is moved to a separate field, `_sourcetype`.

**Note:** The indexed events still contain the original source type name. The renaming occurs only at search time. Also, renaming the source type does only that; it does not fix any problems with the indexed format of your event data caused by assigning the wrong source type in the first place.

To rename the source type, add the `rename` attribute to your source type stanza:

```
rename = <string>
```

**Note:** A source type name can only contain the letters `a` through `z`, the numerals `0` through `9`, the `:` character, and the `_` (underscore) character.

For example, say you're using the source type "cheese_shop" for your application server. Then, accidentally, you index a pile of data as source type "whoops". You can rename "whoops" to "cheese_shop" with this `props.conf` stanza:

```
[whoops]
rename=cheese_shop
```
Now, a search on "cheese_shop" will bring up all the "whoops" events as well as any events that had a "cheese_shop" source type from the start:

```
sourcetype=cheese_shop
```

If you ever need to single out the "whoops" events, you can use `_sourcetype` in your search:

```
_sourcetype=whoops
```

**Important:** Data from a renamed source type will only use the search-time configuration for the target source type ("cheese_shop" in this example). Any field extractions for the original source type ("whoops" in the example) will be ignored.
Manage event segmentation

About event segmentation

Segmentation breaks events up into searchable segments at index time, and again at search time. Segments can be classified as major or minor. Minor segments are breaks within major segments. For example, the IP address 192.0.2.223 is a major segment. But this major segment can be broken down into minor segments, such as "192", as well as groups of minor segments like "192.0.2".

You can define how detailed the event segmentation should be. This is important because index-time segmentation affects indexing and search speed, storage size, and the ability to use typeahead functionality (where Splunk Web provides items that match text you type into the Search bar). Search-time segmentation, on the other hand, affects search speed and the ability to create searches by selecting items from the results displayed in Splunk Web.

For more information about the distinction between "index time" and "search time," see "Index time versus search time" in the Managing Indexers and Clusters manual.

You can assign segmentation to specific categories of events in props.conf, as described in "Set the segmentation for event data".

If you have Splunk Enterprise, you configure index-time segmentation on the indexer or heavy forwarder machines, and search-time segmentation on the search head.

If you have Splunk Cloud, you configure index-time segmentation on heavy forwarder machines, and must file a Support ticket to configure search-time segmentation.

Types of event segmentation

There are three main types, or levels, of segmentation, configurable at index or search time:

- **Inner segmentation** breaks events down into the smallest minor segments possible. For example, when an IP address such as 192.0.2.223 goes through inner segmentation, it is broken down into 192,
0, 2, and 223. Setting inner segmentation at index time leads to faster indexing and searching and reduced disk usage. However, it restricts the typeahead functionality, so that a user can only type ahead at the minor segment level.

- **Outer segmentation** is the opposite of inner segmentation. Under outer segmentation, Splunk software only indexes major segments. For example, the IP address 192.0.2.223 gets indexed as 192.0.2.223, which means that 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 192.0* and you will get any events that have IP addresses that start with 192.0. Also, outer segmentation disables the ability to click on different segments of search results, such as the 192.0 segment of the same IP address. Outer segmentation tends to be marginally more efficient than full segmentation, while inner segmentation tends to be much more efficient.

- **Full segmentation** is 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 192.0 and 192.0.2. This is the least efficient indexing option, but it provides the most versatility in terms of searching.

The `segmenters.conf` file, located in `$SPLUNK_HOME/etc/system/default`, defines all available segmentation types. By default, index-time segmentation is set to the indexing type, which is a combination of inner and outer segmentation. Search-time segmentation is set to full segmentation.

**No segmentation**

The most space-efficient segmentation setting is to disable segmentation completely. This has significant implications for search, however. By disabling segmentation, you restrict searches to indexed fields, such as time, source, host, and source type. Searches for keywords will return no results. 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 capability.

**Configure segmentation types**

`segmenters.conf` defines segmentation types. You can define custom segmentation types, if necessary.

For information on the types of segmentation available by default, look at the `segmenters.conf` file in `$SPLUNK_HOME/etc/system/default`.
Important: Do not modify the default file. If you want to make changes to the existing segmentation stanzas or create new ones altogether, you can copy the default file to $SPLUNK_HOME/etc/system/local/ or to a custom app directory in $SPLUNK_HOME/etc/apps/. For information on configuration files and directory locations, see "About configuration files".

Set segmentation types for specific hosts, sources, or source types

You can configure index-time and search-time segmentation to apply to specific hosts, sources, or source types. If you run searches that involve a particular source type on a regular basis, you could use this capability 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 apply segmentation types to specific event categories, see "Set the segmentation for event data".

Set the segmentation for event data

By default, Splunk software segments events during indexing to allow for the most flexible searching. There are numerous types of segmentation available, and you can create others if necessary. The type of segmentation that you employ affects indexing speed, search speed, and the amount of disk space the indexes occupy. To learn more about segmentation and the trade-offs between the various types of segmentation, refer to "About segmentation".

Splunk software can also segment events at search time. You can set search-time segmentation in Splunk Web, as described in "Set search-time segmentation in Splunk Web".

If you know how you want to search for or process events from a specific host, source, or source type, you can configure index-time segmentation for that specific type of event. You can also configure search-time segmentation options for specific types of events.

Specify segmentation in props.conf

Specify segmentation for events of particular hosts, sources, or source types by assigning segmentation types to the appropriate stanzas in props.conf. In the
stanzas, you assign segmentation types, or "rules", that have been defined in segmenters.conf. These can either be predefined types (such as inner, outer, or full), or custom types that you’ve defined. For more information on defining custom types, read “Configure segmentation types”.

The attribute you configure in props.conf to use these types depends on whether you’re configuring index-time or search-time segmentation:

- For index-time segmentation, use the SEGMENTATION attribute.
- For search-time segmentation, use the SEGMENTATION-<segment selection> attribute.

You can define either one of the attributes or both together in the stanza.

Add your stanza to $SPLUNK_HOME/etc/system/local/props.conf.

Index-time segmentation

The SEGMENTATION attribute determines the segmentation type used at index time. Here’s the syntax:

```
[<spec>]
SEGMENTATION = <seg_rule>
```

[<spec>] can be:

- <sourcetype>: A source type in your event data.
- host::<host>: A host value in your event data.
- source::<source>: A source of your event data.

SEGMENTATION = <seg_rule>

- This specifies the type of segmentation to use at index time for [<spec>] events.
- <seg_rule>
  - A segmentation type, or "rule", defined in segmenters.conf
  - Common settings are inner, outer, none, and full, but the default file contains other predefined segmentation rules as well.
  - Create your own custom rule by editing $SPLUNK_HOME/etc/system/local/segmenters.conf, as described in "Configure segmentation types".
Search-time segmentation

The SEGMENTATION-<segment_selection> attribute helps determine the segmentation type used at search time. Here's the syntax:

```{code}
[<spec>]
SEGMENTATION-<segment_selection> = <seg_rule>
```

[<spec>] can be:

- `<sourcetype>`: A source type in your event data.
- `host::<host>`: A host value in your event data.
- `source::<source>`: A source of your event data.

SEGMENTATION-<segment_selection> = <seg_rule>

- This specifies the type of segmentation to use at search time in Splunk Web for [<spec>] events.
- `<segment_selection>` can be one of the following: full, inner, outer, or raw.
  - These four values are the set of options displayed in the Event segmentation dropdown box in the Results display options panel, invoked from Options directly above search results in Splunk Web.
  - Note that these values are just the set of available Splunk Web dropdown options. You use this attribute to specify the actual segmentation type that the option invokes, which might not be of the same name as the dropdown option itself. For example, you could even define the "inner" dropdown option to invoke the "outer" segmentation type, not that you'd likely want to.
  - By mapping the dropdown option to a `<seg_rule>`, a user can later specify the option when looking at search results to set search-time segmentation, as described in "Set search-time segmentation in Splunk Web".
- `<seg_rule>`
  - A segmentation type, or "rule", defined in segmenters.conf
  - Common settings are inner, outer, none, and full, but the default file contains other predefined segmentation rules as well.
  - Create your own custom rule by editing
    `$SPLUNK_HOME/etc/system/local/segmenters.conf`, as described in "Configure segmentation types".
Example

This example sets both index-time and search-time segmentation rules for syslog events.

Add the following to the [syslog] source type stanza in props.conf:

```
[syslog]
SEGMENTATION = inner
SEGMENTATION-full = inner
```

This stanza changes the index-time segmentation for all events with a syslog source type to inner segmentation. It also causes the full radio button in Splunk Web to invoke inner segmentation for those same events.

**Note:** You must restart Splunk Enterprise to apply changes to search-time segmentation. You must re-index your data to apply index-time segmentation changes to existing data.

Set search-time event segmentation in Splunk Web

Splunk Web allows you to set segmentation for search results. While this has nothing to do with index-time segmentation, search-time segmentation in Splunk Web 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 options: full, inner, outer, or raw. The default is “full”.

You can configure the meaning of these dropdown options, as described in “Set the segmentation for event data”.

417
Improve the data input process

Use a test index to test your inputs

Before adding new inputs to your production index, it is best to test them out. Add the inputs to a test index. Once you’ve verified that you’re receiving the right data inputs and that the resulting events are in a usable form, you can point the inputs to your default "main" index. You can continue to test new inputs this way over time.

If you find that the inputs you started with are not the ones you want, or that the indexed events don’t appear the way you need them to, you can keep working with the test index until you get results you like. When things start looking good, you can edit the inputs to point to your main index instead.

You can preview how Splunk software will index your data into a test index. During preview, you can adjust some event processing settings interactively. See "The “Set Sourcetype” page" for details.

Use a test index

To learn how to create and use custom indexes, read "Create custom indexes" in the Managing Indexers and Clusters manual. There are a few basic steps, described in detail in that topic:

1. Create the test index, using Splunk Web, or, if you have Splunk Enterprise, using the CLI or by editing indexes.conf directly. See "Create custom indexes" for details.

2. When configuring the data inputs, route events to the test index. You can usually do this in Splunk Web. For each input:

   a. When configuring the input from the Add data page, check the More settings option. It reveals several new fields, including one called Index.

   b. In the Index dropdown box, select your test index. All events for that data input will now go to that index.

   c. Repeat this process for each data input that you want to send to your test index.
You can also specify an index when configuring an input in `inputs.conf`, as described here.

3. When you search, specify the test index in your search command. (By default, Splunk software searches the "main" index.) Use the `index=` command:

```
index=test_index
```

**Note**: When searching a test index for events coming in from your newly created input, use the `Real-time > All time(real-time)` time range for the fields sidebar. The resulting **real-time search** will show all events being written to that index regardless of the value of their extracted time stamp. This is particularly useful if you are indexing historical data into your index that a search for "Last hour" or "Real-time > 30 minute window" would not show.

**Delete indexed data and start over**

If you want to clean out your test index and start over again, use the CLI `clean` command, described here.

**Point your inputs at the default index**

Once you're satisfied with the results and are ready to start indexing for real, you'll want to edit your data inputs so that they point to the default, "main" index, instead of the test index. This is a simple process, just the reverse of the steps you took to use the test index in the first place. For each data input that you've already set up:

1. Go back to the place where you initially configured the input. For example, if you configured the input from the **Add data** page in Splunk Web, return to the configuration screen for that input:

   a. Select **System > System configurations > Data inputs**.

   b. Select the input's data type to see a list of all configured inputs of that type.

   c. Select the specific data input that you want to edit. This will take you to a screen where you can edit it.

   d. Select the **Display advanced settings** option. Go to the field named **Index**.

   e. In the **Index** dropdown box, select the **main** index. All events for that data input will now go to that index.
If you instead used `inputs.conf` to configure an input, you can change the index directly in that file, as described here.

2. Now when you search, you no longer need to specify an index in your search command. By default, Splunk software searches the "main" index.

**Use persistent queues to help prevent data loss**

**Persistent queuing** lets you store data in an input queue to disk. This can help prevent data loss if the **forwarder** or **indexer** gets backed up.

By default, forwarders and indexers have an in-memory input queue of 500KB. If the input stream runs at a faster rate than the forwarder or indexer can process, to a point where the queue is maxed out, undesired consequences occur. In the case of UDP, data drops off the queue and gets lost. For other input types, the application generating the data gets backed up.

By implementing persistent queues, you can help prevent this from happening. With persistent queuing, once the in-memory queue is full, the forwarder or indexer writes the input stream to files on disk. It then processes data from the queues (in-memory and disk) until it reaches the point when it can again start processing directly from the data stream.

**Note:** While persistent queues help prevent data loss if processing gets backed up, you can still lose data if Splunk software crashes. For example, Splunk software holds some input data in the in-memory queue as well as in the persistent queue files. The in-memory data can get lost if a crash occurs. Similarly, data that is in the parsing or indexing pipeline but that has not yet been written to disk can get lost in the event of a crash.

**When can you use persistent queues?**

Persistent queuing is available for certain types of inputs, but not all. Generally speaking, it is available for inputs of an ephemeral nature, such as network inputs, but not for inputs that have their own form of persistence, such as file monitoring.

Persistent queues are available for these input types:

- TCP
- UDP
- FIFO
- Scripted inputs
- Windows Event Log inputs

Persistent queues are not available for these input types:

- Monitor
- Batch
- File system change monitor
- splunktcp (input from Splunk forwarders)

**Configure a persistent queue**

Use the inputs.conf file to configure a persistent queue.

Inputs do not share queues. You configure a persistent queue in the stanza for the specific input.

**Syntax**

To create the persistent queue, specify these two attributes within the particular input's stanza:

```
persistentQueueSize = <integer>(KB|MB|GB|TB)
```

* Max size of the persistent queue file on disk.
* Defaults to 0 (no persistent queue).

**Example**

Here's an example of specifying a persistent queue for a tcp input:

```
[tcp://9994]
persistentQueueSize=100MB
```

**Persistent queue location**

The persistent queue has a hardcoded location, which varies according to the input type.

For network inputs, the persistent queue is located here:

```
$SPLUNK_HOME/var/run/splunk/[tcpin|udpin]/pq__<port>
```
Note: There are two underscores in the file name: \texttt{pq\_<port>}, not \texttt{pq<port>}. For example:

- The persistent queue for TCP port 2012:
  \begin{verbatim}
  $SPLUNK_HOME/var/run/splunk/tcpin/pq__2012
  \end{verbatim}

- The persistent queue for UDP port 2012:
  \begin{verbatim}
  $SPLUNK_HOME/var/run/splunk/udpin/pq__2012
  \end{verbatim}

For FIFO inputs, the persistent queue resides under
\begin{verbatim}
$SPLUNK_HOME/var/run/splunk/fifoin/<encoded\ path>.
\end{verbatim}

For scripted inputs, it resides under
\begin{verbatim}
$SPLUNK_HOME/var/run/splunk/exec/<encoded\ path>.
\end{verbatim}
The FIFO/scripted input stanza in \texttt{inputs.conf} derives the \texttt{<encoded\ path>}.

Troubleshoot the input process

This topic discusses some initial steps you can take to troubleshoot the data input process.

Determine why you do not find the events you expect

When you add an input to your Splunk deployment, that input gets added relative to the app you are in. Some apps write input data to a specific index. If you cannot find data that you are certain is in your Splunk deployment, confirm that you are looking at the right index. See Retrieve events from indexes in the \textit{Search Manual}. You might want to add indexes to the list of default indexes for the role you are using.

- For more information about roles, refer to the topic about roles in the \textit{Securing Splunk Enterprise} manual.
- For more information about troubleshooting data input issues, read the rest of this topic or see I can't find my data! in the \textit{Troubleshooting Manual}.

Note: If you have Splunk Enterprise and add inputs by editing \texttt{inputs.conf}, the inputs might not be recognized immediately. Splunk Enterprise looks for inputs every 24 hours, starting from the time it was last restarted, so if you add a new stanza to monitor a directory or file, it could take up to 24 hours for Splunk Enterprise to start indexing the contents of that directory or file. To ensure that
your input is immediately recognized and indexed, add the input through Splunk Web or the CLI, or restart Splunk services after making edits to inputs.conf.

Troubleshoot your tailed files

You can use the FileStatus Representational State Transfer (REST) endpoint to get the status of your tailed files. For example:

```
curl https://serverhost:8089/services/admin/inputstatus/TailingProcessor:FileStatus
```
You can also monitor the fishbucket, a subdirectory used to keep track of how much of a file’s contents has been indexed. In Splunk Enterprise deployments, the fishbucket resides at $SPLUNK_DB/fishbucket/splunk_private_db. In Splunk Cloud deployments you do not have physical access to this subdirectory.

To monitor the fishbucket, use the REST endpoint. Review the REST API Reference manual for additional information.

Troubleshoot monitor inputs

For a variety of information on dealing with monitor input issues, read "Troubleshooting Monitor Inputs" in the Community Wiki.

Can’t find forwarded data?

Confirm that the forwarder functions properly and is visible to the indexer. You can use the Distributed Management Console (DMC) to troubleshoot Splunk topologies and get to the root of any forwarder issues. Read Monitoring Splunk Enterprise for details.

Resolve data quality issues

This topic helps you troubleshoot event-processing and data quality issues such as the following:

- Incorrect line breaking
- Incorrect event breaking
- Incorrect time stamp extraction
Line breaking issues

Problem

Indicators that you have line breaking issues include the following:

- You have fewer events than you expect and the events are very large, especially if your events are single-line events.
- Line breaking issues are present in the Monitoring Console Data Quality dashboard.
- In the Splunk Web Data Input workflow or in splunkd.log, an error message like the following.

```
12-12-2016 13:45:48.709 -0800 WARN LineBreakingProcessor - Truncating line because limit of 10000 bytes has been exceeded with a line length >= 301367
```

Diagnosis

To confirm that your Splunk software has line breaking issues, do one or more of the following:

- Visit the Monitoring Console Data Quality dashboard. Check the dashboard's table for line breaking issues. See About the Monitoring Console in Monitoring Splunk Enterprise.
- Look for messages in splunkd.log like the following:

```
12-12-2016 13:45:48.709 -0800 WARN LineBreakingProcessor - Truncating line because limit of 10000 bytes has been exceeded with a line length >= 301367
```

- Search for events. Multiple events combined, or a single event broken into many, indicates a line breaking issue.

Solution

To resolve line breaking issues, perform this procedure in Splunk Web:

1. Click Settings > Add data.
2. Click add a file to test or monitor to redo the monitor input.
3. Select a file with a sample of your data.
4. Click **Next**.
5. On the **Set Source Type** page, work with the options on the left until your sample data is correctly broken into events. To configure LINE_BREAKER or TRUNCATE, click **Advanced**.
6. Complete the data input workflow or record the correct settings and use them to correct your existing input configurations.

While you are working with the options on the Set Source Type page, the LINE_BREAKER setting might not be properly set. LINE_BREAKER must have a capturing group and the group must match the events.

For example, you might have a value of LINE_BREAKER that is not matched. Look for messages with "Truncating line because limit of 10000 bytes has been exceeded" in splunkd.log or look for the following message in Splunk Web:

If you find such a message, do the following:

1. Check that LINE_BREAKER is properly configured to segment your data into lines as you expect. Make sure that the string exists in your data.
2. If LINE_BREAKER is configured correctly, and you simply have very long lines, or if you are using LINE_BREAKER as the only method to define events, bypassing line merging later in the indexing pipeline, make sure that TRUNCATE is set large enough to contain the entire data fragment delimited by LINE_BREAKER. The default value for TRUNCATE is 10,000. If your events are larger than the TRUNCATE value, you might want to increase the value of TRUNCATE. For performance and memory usage reasons, do not set TRUNCATE to unlimited.

If you do not specify a capturing group, LINE_BREAKER is ignored.

See **Configure event line breaking.**
Event breaking, or aggregation, issues

Event breaking issues can pertain to BREAK_ONLY_BEFORE_DATE, MAX_EVENTS, and any props.conf setting that contains the keyword "BREAK".

Problem

Indicators that you have aggregation issues include:

- Aggregation issues present in the Monitoring Console Data Quality dashboard.
- An error in the Splunk Web Data Input workflow.
- Count events. If events are missing and are very large, especially if your events are single-line events, you might have event breaking issues.

Diagnosis

To confirm that your Splunk software has event breaking issues, do one or more of the following:

- View the Monitoring Console Data Quality dashboard.
- Search for events that are multiple events combined into one.
- Check splunkd.log for messages such as the following:

  12-07-2016 09:32:32.876 -0500 WARN??AggregatorMiningProcessor - Breaking event because limit of 256 has been exceeded
  12-07-2016 09:32:32.876 -0500 WARN??AggregatorMiningProcessor - Changing breaking behavior for event stream because MAX_EVENTS (256) was exceeded without a single event break. Will set BREAK_ONLY_BEFORE_DATE to False, and unset any MUST_NOT_BREAK_BEFORE or MUST_NOT_BREAK_AFTER rules. Typically this will amount to treating this data as single-line only.

Solution

For line and event breaking, determine whether this is happening for one of the following reasons:

- Your events are properly recognized but too large for the limits in place. MAX_EVENTS defines the maximum number of lines in an event.
- Your events are not properly recognized.

If your events are larger than the limit set in MAX_EVENTS, you can increase limits. But be aware that large events are not optimal for indexing performance,
search performance, and resource usage. Large events can be costly to search. The upper values of both limits result in 10,000 characters per line, as defined by TRUNCATE, times 256 lines, as set by MAX_EVENTS. The combination of those two limits is a very large event.

If the cause is that your events are not properly recognized, which is more likely, your Splunk software is not breaking events as it should. Check the following:

- Your event breaking strategy. The default is to break before the date, so if Splunk software does not extract a time stamp, it does not break the event. To diagnose and resolve, investigate time stamp extraction. See How timestamp assignment works.
- Your event breaking regex.

For more information, see the following topics:

- How timestamp extraction works
- Tune timestamp extraction for better indexing performance
- Configure event line breaking

**Time stamping issues**

Time stamping issues can pertain to the following settings in props.conf:

- DATETIME_CONFIG
- TIME_PREFIX
- TIME_FORMAT
- MAX_TIMESTAMP_LOOKAHEAD
- TZ

See props.conf.spec in the Admin Manual.

See How timestamp assignment works.

**Problem**

Indicators that you have time stamp parsing issues include the following:

- Timestamp parsing issues are present in the Monitoring Console Data Quality dashboard.
- An error occurs in the Splunk Web Data Input work flow.
- Count events. If you are missing events and have very large events, especially if your events are single-line events, parsing might be a
problem.
• Time zone is not properly assigned.
• The value of \_time assigned by Splunk software does not match the time in the raw data.

**Diagnosis**

To confirm that you have a time stamping issue, do one or more of the following:

• Visit the Monitoring Console **Data Quality** dashboard. Check for timestamp parsing issues in the table. Time stamp assignment resorts to various fallbacks, as described in How timestamp assignment works. For most of the fallbacks, even if one of them successfully assigns a time stamp, you still get an issue in the Monitoring Console dashboard.
• Search for events, find that they are multiple events combined into one.
• Look in splunkd.log for messages like:

12-09-2016 00:45:29.956 -0800 WARN DateParserVerbose - Failed to parse timestamp. Defaulting to timestamp of previous event (Fri Dec 9 00:45:27 2016).

Context:
source::/disk2/sh-demo/splunk/var/log/splunk/entity.log|host::svdev-sh-demo|entity-too_small|682235

12-08-2016 12:33:56.025 -0500 WARN??AggregatorMiningProcessor - Too many events (100K) with the same timestamp: incrementing timestamps 1 second(s) into the future to insure retrievability

All events are indexed with the same time stamp, which makes searching that time range ineffective.

**Solution**

To resolve a time stamping issue:

• Make sure that each event has a complete time stamp, including a year, full date, full time, and a time zone.
• See Configure time stamp recognition for additional possible resolution steps.