# Table of Contents

## How to get data into Splunk
- How to get data into Splunk ................................................................................................................1
  - What Splunk can index ..............................................................................................................................1
  - Where is my data? Is it local or remote? .................................................................................................4
  - Use forwarders ........................................................................................................................................4
  - Use apps ..................................................................................................................................................6
  - How to get going .......................................................................................................................................6
  - Configure your inputs ..............................................................................................................................8
- About Windows data and Splunk .............................................................................................................11
- What Splunk does with your data (and how to make it do it better) ......................................................12

## Get data from files and directories
- Get data from files and directories ..........................................................................................................15
  - Monitor files and directories ..................................................................................................................15
  - Use Splunk Web .......................................................................................................................................16
  - Use the CLI ..............................................................................................................................................18
  - Edit inputs.conf .....................................................................................................................................19
  - Specify input paths with wildcards ........................................................................................................23
  - Whitelist or blacklist specific incoming data .........................................................................................26
  - How log file rotation is handled ............................................................................................................28

## Get network events
- Get network events ...................................................................................................................................30
  - Get data from TCP and UDP ports .........................................................................................................30
  - Send SNMP events to Splunk ..................................................................................................................36

## Get Windows data
- Get Windows data ......................................................................................................................................38
  - Considerations for deciding how to monitor remote Windows data .....................................................38
  - Monitor Active Directory .......................................................................................................................43
  - Monitor Windows event log data ............................................................................................................51
  - Monitor WMI-based data ..........................................................................................................................56
  - Monitor Windows Registry data ..............................................................................................................65
  - Real-time Windows performance monitoring ........................................................................................70
  - Monitor Active Directory .......................................................................................................................78

## Other ways to get stuff in
- Other ways to get stuff in .........................................................................................................................79
  - Get data from FIFO queues .....................................................................................................................79
  - Monitor changes to your filesystem .........................................................................................................80
  - Get data from APIs and other remote data interfaces through scripted inputs ..................................84
  - Find more things to monitor with crawl .................................................................................................89

## Recipes
- Recipes ......................................................................................................................................................90
  - Forwarders ..............................................................................................................................................90
  - Files and directories - local .....................................................................................................................91
  - Files and directories - remote ..................................................................................................................91
  - Syslog - local .........................................................................................................................................92
  - Syslog - TCP .........................................................................................................................................92
  - Syslog - UDP .........................................................................................................................................93
<table>
<thead>
<tr>
<th>Recipes</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows event logs - local</td>
<td>94</td>
</tr>
<tr>
<td>Windows event logs - remote</td>
<td>95</td>
</tr>
<tr>
<td>Windows event logs - many remote</td>
<td>96</td>
</tr>
<tr>
<td>Windows registry - local</td>
<td>96</td>
</tr>
<tr>
<td>Windows registry - remote</td>
<td>97</td>
</tr>
<tr>
<td>Windows performance monitoring - local</td>
<td>98</td>
</tr>
<tr>
<td>Windows performance monitoring - remote</td>
<td>99</td>
</tr>
<tr>
<td>Windows performance - many remote</td>
<td>100</td>
</tr>
<tr>
<td>Windows Active Directory</td>
<td>100</td>
</tr>
<tr>
<td>Unix logs - local</td>
<td>101</td>
</tr>
<tr>
<td>FSChange - local</td>
<td>102</td>
</tr>
<tr>
<td>WebSphere - local</td>
<td>102</td>
</tr>
<tr>
<td>IIS logs - local</td>
<td>103</td>
</tr>
<tr>
<td>Apache logs - local</td>
<td>103</td>
</tr>
<tr>
<td>JMS/JMX - write your own</td>
<td>104</td>
</tr>
</tbody>
</table>

| Configure event processing                                           | 106  |
| Overview of event processing                                          | 106  |
| Configure character set encoding                                     | 106  |
| Configure linebreaking for multi-line events                         | 110  |
| Handle event timestamps                                              | 113  |
| Extract default fields automatically                                 | 114  |
| Improve data compression with segmentation                            | 115  |
| Assign metadata to events dynamically                                | 116  |
| Anonymize data with sed                                              | 117  |
| Anonymize data using configuration files                               | 118  |

| Configure event timestamping                                         | 120  |
| How timestamp assignment works                                       | 120  |
| Configure timestamp recognition                                      | 121  |
| Improve Splunk's ability to recognize timestamps                     | 126  |
| Configure timestamp assignment for events with multiple timestamps  | 130  |
| Specify timezones of timestamps                                      | 131  |
| Tune timestamp recognition for better indexing performance           | 133  |

| Configure indexed field extraction                                   | 135  |
| About default fields (host, source, sourcetype, and more)            | 135  |
| Set a default host for a Splunk server                               | 139  |
| Set a default host for a file or directory input                     | 141  |
| Override default host values based on event data                     | 144  |
| Handle incorrectly-assigned host values                              | 146  |
| Override automatic source type assignment                            | 146  |
| Advanced source type overrides                                       | 149  |
| Configure rule-based source type recognition                        | 151  |
# Table of Contents

## Configure indexed field extraction
- Rename source types ............................................................................................................ 152
- Train Splunk's source type autoclassifier ............................................................................... 153
- List of pretrained source types ........................................................................................... 154
- Specify source type settings in props.conf ........................................................................... 159
- Configure index-time field extractions .................................................................................. 160
- Extract fields from file headers at index time ...................................................................... 167

## Improve the data input process ............................................................................................. 173
- Test your inputs ...................................................................................................................... 173
- Use persistent queues to help prevent data loss ................................................................... 174
- Troubleshoot the input process ............................................................................................ 176
How to get data into Splunk

What Splunk can index

The first step in using Splunk is to feed it data. Once Splunk gets some data, it immediately indexes it, so that it's available for searching. With its universal indexing ability, Splunk transforms your data into a series of individual events, consisting of searchable fields. There's lots you can do to massage the data before and after Splunk indexes it, but you don't usually need to. In most cases, Splunk can determine what type of data you're feeding it and handle it appropriately.

Basically, you point Splunk at data and it does the rest. In moments, you can start searching on the data, or use it to create charts, reports, alerts, and other interesting outputs.

What kind of data?

Any data. In particular, any and all IT streaming and historical data. Stuff like event logs, web logs, live application logs, network feeds, system metrics, change monitoring, message queues, archive files, or anything else of interest. Any data. Really.

Point Splunk at a data source. Tell Splunk a bit about the source. That source then becomes a data input to Splunk. Splunk begins to index the data stream, transforming it into a series of individual 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 you're satisfied.

The data can be on the same machine as the Splunk indexer (local data), or it can be on another machine altogether (remote data). You can easily get remote data into Splunk, either by using network feeds or by installing Splunk forwarders on the machines where the data originates. Forwarders are lightweight versions of Splunk that consume data and then forward it on to the main Splunk instance for indexing and searching. For more information on local vs. remote data, see "Where is my data?".

To make the job easier, Splunk offers lots of free apps and add-ons, with pre-configured inputs for things like Windows- or Linux-specific data sources, Cisco security data, Blue Coat data, and so on. Look in Splunkbase for an app or add-on that fits your needs. Splunk also comes with dozens of recipes for data sources like web logs, J2EE logs, or Windows performance metrics. You can get to these from the Add data section of Splunk Web, described later. If the recipes and apps don't cover your needs, then you can use Splunk's more general input configuration capabilities to specify your particular data source. These generic data sources are discussed here.

How to specify data inputs

You add new types of data to Splunk by telling it about them. There are a number of ways you can specify a data input:

- **Apps.** Splunk has a large and growing variety of apps and add-ons that offer preconfigured inputs for various types of data sources. Take advantage of Splunk apps and free yourself from having to configure the inputs yourself. For more information, see "Use apps".
• **Splunk Web.** You can configure most inputs using the Splunk Web data input pages. These provide a GUI-based approach to configuring inputs. You can access the Add data landing page from either Splunk Home or Manager. See "Use Splunk Web".

• **Splunk's CLI.** You can use the CLI (command line interface) to configure most types of inputs. See "Use the CLI".

• **The inputs.conf configuration file.** When you specify your inputs with Splunk Web or the CLI, the configurations get saved in an inputs.conf file. You can edit that file directly, if you prefer. To handle some advanced data input requirements, you might need to edit it. See "Edit inputs.conf".

In addition, if you use forwarders to send data from outlying machines to a central indexer, you can specify some inputs during forwarder installation. See "Use forwarders".

For more information on configuring inputs, see "Configure your inputs".

**Types of data sources**

As described earlier, Splunk provides tools to configure all sorts of data inputs, including many 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 the data you might be interested in comes directly from files and directories. For the most part, you can use Splunk's files and directories monitor input processor to get data from files and directories.

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

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

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

**Network events**

Splunk can index data from any network port. For example, Splunk can index remote data from syslog-ng or any other application that transmits via TCP. Splunk can also index UDP data, but we recommend using TCP instead whenever possible, for enhanced reliability.

Splunk 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".

To get SNMP data, see "Send SNMP events to Splunk".

**Windows sources**

The Windows version of Splunk includes a wide range of Windows-specific inputs. It also provides pages in Splunk Manager for defining the Windows-specific input types listed below:

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

**Important:** You can index and search Windows data on a non-Windows instance of Splunk, but you must first use a Windows instance to gather the data. You can do this with a Splunk forwarder running on Windows. You configure the forwarder to gather Windows inputs and then forward the data to the non-Windows instance. See "Considerations for deciding how to monitor remote Windows data" for details.

For a more detailed introduction to using Windows data in Splunk, see "About Windows data and Splunk".

**Other sources**

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

- FIFO queues
- Scripted inputs -- for getting data from APIs and other remote data interfaces and message queues.

**Other things to consider**

The topics that follow this one discuss issues to consider when specifying Splunk data:

- "Where is my data?". A concise explanation of remote vs. local data, and why it matters.
- "Use forwarders". How to use forwarders to simplify the remote collection of data.
- "Use apps". How to use Splunk apps to get your data into Splunk quickly.
- "How to get going". An overview of the process of getting and configuring data sources, with tips on best practices.
- "Configure your inputs". The ways you can configure data inputs in Splunk.
- "About Windows data and Splunk". An introduction to getting Windows data into Splunk.
- "What Splunk does with your data (and how to make it do it better)". What happens to your data once it enters Splunk, and how you can configure Splunk to make the data even more useful.
Where is my data? Is it local or remote?

When initially getting data into Splunk, you might encounter some confusion as to what type of data is considered "local" and what type of data is considered "remote." The distinction between local and remote is particularly important when you are navigating the new data input pages.

The answer to this question depends on a number of criteria, including (but not limited to):

- The operating system on which your Splunk instance resides
- The types of data stores that are directly connected to your Splunk instance
- Whether any authentication or other intermediate steps are needed to access the data store that contains the data you want to index

Local

A **local** resource is a fixed resource that your Splunk server has direct access to, meaning you are able to access it - and whatever is contained within it - without having to attach, connect, or perform any other intermediate action (such as authentication or mapping a network drive) in order to have that resource appear available to your system. If your data is on such a resource, the data is considered "local."

Remote

A **remote** resource is any resource where the above definition is not satisfied. Network drives mapped from Windows systems, Active Directory schemas, and NFS or other network-based mounts on *nix systems are examples that qualify for this designation. Data gathered from these resource endpoints is also considered "remote."

Exceptions

There are cases where resources that would normally be considered remote are actually not. Here are some examples:

- A machine has a volume permanently mounted over a high-bandwidth physical connection such as USB or FireWire. Since the computer can mount the resource at boot time, it's treated as a local resource, even though the resource can theoretically be disconnected at a later time.
- A machine has a resource 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

**Forwarders** are lightweight Splunk instances, whose main purpose is to consume data and forward it on to Splunk **indexers** for further processing. They require minimal resources and have little impact on performance, so they can usually reside on the machines where the data originates.
For example, say you have a number of Apache servers generating data that you want to search centrally. You can install a Splunk indexer on its own Linux machine and then set up forwarders on the Apache machines. The forwarders can take the Apache data and send it on to the Splunk indexer, which then consolidates and indexes it and makes it available for searching. Because of their light footprint, the forwarders won't affect the performance of the Apache servers.

Similarly, you can install forwarders on your employees' Windows desktops. These can send logs and other data to a central Splunk instance, where you can view the data as a whole to track malware or other issues.

**What forwarders do**

You can use forwarders to get data from remote machines. They represent a much more robust solution than raw network feeds, with their capabilities for:

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

Forwarders consume data in the same way as any other Splunk instance. They can handle exactly the same types of data as a Splunk indexer. The difference is that they usually do not index the data themselves. Instead, they just get the data and send it on to a central Splunk indexer, which does the indexing and searching. A single indexer can index data coming from many forwarders. For detailed information on forwarders, see the "Forwarding data" section of the Distributed Deployment manual.

In most Splunk deployments, forwarders serve as the primary consumers of data. It's only in single-machine deployments that the Splunk indexer is likely to also be the main data consumer. In a large Splunk deployment, you might have hundreds or even thousands of forwarders consuming data and forwarding it on to a group of indexers for consolidation.

**How to configure forwarder inputs**

As lightweight instances of Splunk, forwarders, by design, have limited capabilities. For example, most forwarders do not include Splunk Web, so you do not have direct access to Splunk Manager for setting up the forwarder's data inputs. Here are the main ways that you can configure a forwarder's data inputs:

- Specify inputs during initial deployment. For Windows forwarders, you can specify common inputs during the installation process itself. For *nix forwarders, you can specify inputs directly after installation.
- Use the CLI.
- Edit inputs.conf.
- Deploy an app containing the desired inputs.
- Use Splunk Web on a full Splunk test instance to configure the inputs and then distribute the resulting inputs.conf file to the forwarder itself.
For detailed information on forwarders, including use cases, typical topologies, and configurations, see "About forwarding and receiving" in the Distributed Deployment manual.

For details on forwarder deployment, including how to use the deployment server to simplify distribution of configuration files and apps to multiple forwarders, see "Universal forwarder deployment overview" in the Distributed Deployment manual.

For information on using forwarders for monitoring of remote Windows data, see "Considerations for deciding how to monitor remote Windows data".

**Use apps**

Splunk offers a wide variety of apps and add-ons that extend Splunk’s capabilities. Apps simplify the process of getting data into Splunk. Instead of configuring inputs yourself for a particular environment or application, you can often find an app with the data inputs preconfigured. 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, there's a Windows app that provides pre-built data inputs, searches, reports, alerts, and dashboards for Windows server and desktop management. There's a *nix app that offers the same for Unix and Linux environments. There's a wide range of other apps that handle specific types of application data, such as the Splunk for Blue Coat app, the Splunk for F5 app, the Splunk for Cisco Security app, and the Splunk for Websphere Application Server app.

The Add Data landing page in Splunk Web is a good place to start. Its data input pages link directly to a number of the most popular apps. You can also go directly to Splunkbase to browse through the large and growing set of apps available for download. Check Splunkbase frequently; new apps are being added all the time.

**Important:** If Splunk Web is located behind a proxy server, you might have trouble accessing Splunkbase directly within Splunk. To solve this problem, you need to set the http_proxy environment variable, as described in "Specify a proxy server" in the Admin manual.

For more information on apps in general, see the set of topics beginning with "What are apps and add-ons?" in the Admin manual. In particular, this topic tells you how to download and install apps: "Where to get more apps and add-ons".

For information on how to create your own apps, read the Developer Manual.

**How to get going**
How to get going

It's easy to get started with Splunk. You just point Splunk at some data by configuring an input from the Add data page. Or, even easier, you can download and enable a relevant app, such as one of the OS apps (Splunk for Windows or Splunk for Unix and Linux). Once you've configured the inputs or enabled the app, Splunk immediately starts indexing the specified data. In a short time, you can go to the Search app (reachable from Splunk Home, the starting page for Splunk Web) and begin to explore the data in detail.

It's easy, but still... it's a good idea to work first with a test index.

Add new inputs

Here's a recommended way to start out:

1. Understand your needs. Some of the questions you might ask yourself include:
   - What kind of data do I want Splunk to index? Look here for a quick guide to the types of data Splunk indexes.
   - Is there an app for that? See "Use apps" to find out if there's a pre-configured app that will meet your needs.
   - 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".
   - What do I want to do with the indexed data? Get a sense of the possibilities; start by reading "What is Splunk knowledge?'

2. Start out small, by creating a test index and adding just a few inputs. Look here for information on setting up a test index.

Important: Try to keep the amount of test data to a minimum; any data added to your test index counts against your maximum daily indexing volume for licensing purposes.

3. Run some searches on the test data:
   - Are you seeing the sort of data you were expecting?
   - Did Splunk's default configurations work well for your events?
   - Is there stuff missing or mangled?
   - Are the results optimal?

4. If necessary, massage your input and event processing configurations until events look the way you want them to. To learn how to configure event processing, see "What Splunk does with your data" in this manual.

5. Delete the data from your test index and start over, if necessary. Look here for information on how to do that.

6. When you're ready for prime time, point your inputs to the default "main" index, as described here.

When you've got other inputs to add, adopt the same approach.
Note: The remaining sections in this topic provide details for several of the steps outlined above. The details for the other steps are described elsewhere in the online documentation. Just follow the steps outlined in the procedure and click on the links provided.

Got custom data? It might need some extra TLC

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

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

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

Configure your inputs

To add a new type of data to Splunk, you first need to tell it a few things about the data. You do this by configuring a data input. There are a number of ways to configure your inputs:

- **Apps.** Splunk has a large and growing variety of apps that offer preconfigured inputs for various types of data. Take advantage of Splunk apps and free yourself from having to configure the inputs yourself. For more information, see "Use apps".

- **Splunk Web** You can configure most inputs using the Splunk Web data input pages. These provide a GUI-based approach to configuring inputs. You can access the Add data landing page from Splunk Home. You can also use Manager to add new inputs or view and manage existing inputs.

- **Splunk’s CLI.** You can use the CLI to configure most types of inputs.

- **inputs.conf configuration file.** When you specify your inputs with Splunk Web or the CLI, the configurations get saved in a configuration file, inputs.conf. You can edit that file directly, if you prefer. To handle some advanced data input requirements, you might need to edit it.

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

This topic describes how to configure data inputs yourself, using Splunk Web, the CLI, or inputs.conf.
You can add data inputs from Splunk Home or Splunk Manager:

- From Splunk Home, select Add Data. This takes you to the Add Data page, with links to recipes for a wide variety of data input types. This is the easiest way to start adding inputs.

- From anywhere in Splunk Web, select Manager. Then select Data inputs from the Data section of the Manager page. This takes you to a page where you can view and manage your existing inputs, as well as add new ones.

The Add data page contains two groups of links. The first group contains links for some common data types, with recipes to get you going. The second group contains links to all the types of inputs that you can configure.

If you're just starting out, look in the first group of links to see if there's a data type that matches your needs. For example, if you click on Syslog, you'll go to a page with information on the different types of syslog data and links to recipes for each type. Or click on Apache logs for specific recipes for that data type.

For more information on using Splunk Web to configure your inputs, look in the topics covering specific inputs later in this manual. For example, to learn how to use Splunk Web to configure network inputs, look here: "Get data from TCP and UDP ports". You can configure most inputs with Splunk Web. For a small number of input types, such as file system change monitoring, you'll need to edit inputs.conf directly. In addition, some advanced settings for other input types are available only through inputs.conf.

Important: When you add an input through Splunk Web, Splunk adds that input to a copy of inputs.conf that belongs to the app you're currently in. This has consequences that you need to consider. For example, if you navigated to Splunk Manager directly from the Search page and then added an input there, your input will be added to $SPLUNK_HOME/etc/apps/search/local/inputs.conf. Make sure you're in the desired app when you add your inputs. For background on how configuration files work, read "About configuration files".

Use the CLI

You can use the Splunk CLI to configure most inputs. Navigate to the $SPLUNK_HOME/bin directory and use the ./splunk command from the UNIX or Windows command prompt. For example, this command adds /var/log/ as a data input:

./splunk add monitor /var/log/

If you get stuck, Splunk's CLI has built-in help. For the list of CLI commands, type:

./splunk help commands

Individual commands have their own help pages as well. To see them, type:

./splunk help <command>
For information on how to use the CLI to configure a specific input, read the topic in this manual for that input. For example, to learn how to use the CLI to configure network inputs, look here: "Add a network input using the CLI".

For general information on the CLI, read "About the CLI" and the topics that follow it in the Admin manual.

**Edit inputs.conf**

To add an input by directly editing inputs.conf, add a **stanza** for the input. You can add the stanza to the inputs.conf file in $SPLUNK_HOME/etc/system/local/, or in your own custom application directory (in $SPLUNK_HOME/etc/apps/<app name>/local). If you have not worked with Splunk's configuration files before, read "About configuration files" before you begin.

You configure the data input by adding attribute/value pairs to its stanza. You can set multiple attributes in an input stanza. If you do not specify a value for an attribute, Splunk uses the default value that's preset in $SPLUNK_HOME/etc/system/default/inputs.conf.

Here's a simple example of adding a network input. This configuration directs Splunk to listen on TCP port 9995 for raw data from any remote server. The host of the data is set as the DNS name of the remote server. All data will also be assigned the source type "log4j" and the source "tcp:9995".

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

For information on how to configure a specific input, read the topic in this manual for that input. For example, to learn how to configure file inputs, look here.

The topic for each data input describes the main attributes available for that input. However, you should always refer to the inputs.conf spec file, located here, for the complete list of available attributes. The spec file contains detailed descriptions of the attributes. It's followed by a file containing several examples.

**A word about source types**

As part of the input process, Splunk assigns a source type to the data. The source type identifies the format of the data. Splunk uses the source type during indexing to format events correctly. It usually knows what source type to assign. For instance, syslog data gets a source type of "syslog". If you're not happy with the source type Splunk assigns to a particular input, you can substitute a different source type -- either one of the default source types or one that you create yourself. You set the source type at the time you configure the input, using any of the configuration methods described in this topic.

For more information on source types and other default fields, see "About default fields". The topic "Override automatic source type assignment" describes source type assignment options in detail.

To learn how to set the source type on a per-event basis, see "Advanced source type overrides". 
About Windows data and Splunk

Splunk is an extensible, powerful tool. It can index many different kinds of Windows data. This data can be pretty much anything - a log file, a directory full of files, an event log channel, the Registry, or Active Directory.

Splunk provides several specialized inputs to monitor Windows data, including:

- **Windows Event Logs**: Splunk can monitor logs generated by the Windows Event Log service on any event log channel that is available on the machine. You can collect logs on the local machine, or you can gather log data remotely using either a universal forwarder or Windows Management Instrumentation (WMI).

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

- **Remote monitoring over WMI**: Splunk 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 Splunk's built-in Registry monitoring capabilities. You can use a universal forwarder to gather Registry data from remote machines.

- **Active Directory monitoring**: Splunk can audit any changes to the Active Directory - including changes to user, group, machine, and group policy objects.

These specialized inputs are available only on Windows installations of Splunk. You also have available the standard set of Splunk inputs, such as "files and directories," the network monitoring inputs, and scripted inputs.

**The Splunk for Windows app**

The Splunk for Windows app provides pre-built 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 scripted inputs for CPU, disk I/O, memory, event logs, configurations, and user data, plus a web-based setup UI for indexing Windows event logs. This free app makes getting started with Splunk on Windows easier.

**The Windows app and Splunk's new performance metric collection features**

The Windows app currently does not make use of the new Windows performance monitor collection features available in Splunk 4.2. While the app does work, and is supported, by default it will continue to gather local performance metrics over WMI.

If you want to use the new features, or you're using a universal forwarder to send data with the default performance monitoring data collections to an instance that's running the app, then you'll need
to update the searches within the app to reflect the performance monitoring collections you have defined.

Initial considerations for deploying Splunk on Windows

When installing and deploying Splunk on Windows, consider the following:

- **Authentication.** To perform any operations on remote Windows machines in your network, Splunk needs the appropriate access to your Active Directory domain or forest. This means that Splunk must run as a user with those credentials. It is best practice to have these credentials available before proceeding with any kind of deployment. Review "Considerations for deciding how to monitor remote Windows data" for additional information about how best to accomplish this task.

- **Disk bandwidth.** Splunk indexers require lots of disk I/O bandwidth, particularly when indexing large amounts of data. **You must pay particular attention to systems that have any program that intermediates between Splunk and the operating system installed. This includes anti-virus software.** Make sure that you configure any installed anti-virus software to avoid monitoring Splunk directories or processes, as such scans will significantly reduce performance.

- **Shared servers.** Before installing Splunk on a server running other services, read "Hardware capacity planning for your Splunk deployment" in the Installation manual. This is particularly important if you plan to install Splunk on a domain controller, or a computer running memory-intensive services such as Exchange, SQL Server or a virtual host server.

The most efficient way to index data from any Windows server is to install a universal forwarder on the machines from which you want to gather data. Unlike the full version of Splunk, Splunk forwarders are designed to have a small footprint and use very little resources. In some cases, such as Registry monitoring, you must use a forwarder, as you cannot poll Registry changes remotely.

**What Splunk does with your data (and how to make it do it better)**

Splunk consumes any sort of data and **indexes** it, transforming it into useful and searchable knowledge in the form of **events.** The **data pipeline,** displayed below, shows the main processes that act on the data during indexing. These processes constitute **event processing.** After the data has been processed into events, you can associate the events with **knowledge objects** to further enhance their usefulness.

**The data pipeline**

Once a chunk of data enters Splunk, it moves through the data pipeline, which transforms the data into searchable events. This diagram shows the main steps in the data pipeline:
For a concise description of the data pipeline, see "How data moves through Splunk" in the Distributed Deployment manual.

Splunk makes reasonable decisions for most types of data during event processing, so that the resulting events are immediately useful and searchable. However, depending on the data and what sort of knowledge you need to extract from it, you might want to tweak one or more steps of event processing.

**Event processing**

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

During both parsing and indexing, Splunk acts on the data, transforming it in various ways. Most of these processes are configurable, so you have the ability to adapt them to your needs. In the description that follows, each link takes you to a topic that discusses one of these processes, with information on ways you can configure it.

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

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

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

The distinction between parsing and indexing pipelines matters mainly for forwarders. Heavy forwarders can fully parse data locally and then forward the parsed data on to receiving indexers, where the final indexing occurs. With universal forwarders, on the other hand, the data gets forwarded after very minimal parsing. Most parsing then occurs on the receiving indexer.

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

**Enhance and refine events**

Once the data has been transformed into events, you can make the events even more useful by associating them with knowledge objects, such as event types, field extractions, and saved searches. For information about managing Splunk knowledge, read the Knowledge Manager manual, starting with "What is Splunk knowledge?".
Get data from files and directories

Monitor files and directories

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

You can add inputs to monitor or upload using any of these methods:

- Splunk Web
- The CLI
- inputs.conf

How monitor works in Splunk

Specify a path to a file or directory and Splunk's 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 J2EE or .NET applications, Web access logs, and so on. Splunk continues to monitor and index the file or directory as new data appears. You can also specify a mounted or shared directory, including network file systems, so long as Splunk can read from the directory. If the specified directory contains subdirectories, Splunk recursively examines them for new files.

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

When using monitor, note the following:

- On most file systems, files can be read even as they are being written to. However, Windows file systems have the ability to prevent files from being read while they are being written, and some Windows programs may use these modes, though most do not.
- Files or directories can be included or excluded via **whitelists** and **blacklists**.
- Upon restart, Splunk continues processing files where it left off.
- Splunk decompresses archive files before it indexes them. It can handle these common archive file types: .tar, .gz, .bz2, .tar.bz2, and .zip.
- Splunk detects log file rotation and does not process renamed files it has already indexed (with the exception of .tar and .gz archives; for more information see "Log file rotation" in this manual).
- The entire **dir/filename** path must not exceed 1024 characters.
- Removing an input does not stop the the input's files from being indexed. Rather, it stops files from being checked again, but all the initial content will be indexed. To stop all in-process data, you must restart the Splunk server.
Note: You cannot currently use both monitor and file system change monitor to follow the same directory or file. If you want to see changes in a directory, use file system change monitor. If you want to index new events in a directory, use monitor.

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

Why use upload or batch

To index a static file once, select Upload a local file or Index a file on the Splunk server in Splunk Web. The file will not be monitored on an ongoing basis.

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

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

Use Splunk Web

Use Splunk Web

To use Splunk Web to add inputs from files and directories:

A. Go to the Add New page

You add an input from the Add New page in Splunk Web. You can get there through two routes:

- Splunk Manager
- Splunk Home

It doesn’t matter which route you use to get there; the Add New page itself is the same either way.

Via Splunk Manager:

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

2. In the Data section of the Manager page, click Data Inputs.

3. Click Files & Directories.

4. Click the New button to add an input.

Via Splunk Home:

1. Click the Add Data link in Splunk Home.

2. Click the From files and directories link to add an input.
B. Specify the input

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

   - **Continuously index data from a file or directory this Splunk instance can access.** Sets up an ongoing input. Whenever data is added to this file or directory, Splunk will index it. Read the next section for advanced options specific to this choice.
   - **Upload and index a file.** Uploads a file from your local machine into Splunk.
   - **Index a file once from this Splunk server.** Copies a file on the server into Splunk via the batch directory.

2. Specify the **Full path** to the file or directory. (If you selected the **Upload a local file** radio button, the field is called **File** instead.)

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

3. To access other settings, check **More settings.** A number of additional settings appear. You can usually go with the defaults for these settings. If you want to set them explicitly, here’s what they’re for:

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

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

   b. You can set the **Source type.** **Source type** is a default field added to events. Source type is used to determine processing characteristics, such as timestamps and event boundaries. For information on overriding Splunk's automatic source typing, see "Override automatic source type assignment" in this manual.

      For directories, set the source type to **Automatic.** If the directory contains files of different formats, do not set a value for the source type manually. By doing so, you'll force a single source type for all files in that directory.

   c. You can set the **Index** 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 has a number of utility indexes, which also appear in this dropdown box.

4. Click **Save**.

   **Advanced options for file/directory monitoring**

   If you selected the **Monitor a file or directory** radio button for your source, the **More settings** section also includes an **Advanced options** section, which allows you to configure some additional settings:

   - **Follow tail.** If checked, monitoring begins at the end of the file (like `tail -f`).

17
• **Whitelist.** If a path is specified, files from that path are monitored only if they match the specified regex.

• **Blacklist.** If a path is specified, files from that path are not monitored if they match the specified regex.

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

## Use the CLI

### Use the CLI

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

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

### CLI commands for input configuration

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

<table>
<thead>
<tr>
<th>Command</th>
<th>Command syntax</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>add monitor $SOURCE [-parameter value] ...</td>
<td>Add inputs from $SOURCE.</td>
</tr>
<tr>
<td>edit</td>
<td>edit monitor $SOURCE [-parameter value] ...</td>
<td>Edit a previously added input for $SOURCE.</td>
</tr>
<tr>
<td>remove</td>
<td>remove monitor $SOURCE</td>
<td>Remove a previously added $SOURCE.</td>
</tr>
<tr>
<td>list</td>
<td>list monitor</td>
<td>List the currently configured monitor.</td>
</tr>
<tr>
<td>spool</td>
<td>spool source</td>
<td>Copy a file into Splunk via the sinkhole directory.</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>Yes</td>
<td>Path to the file or directory to monitor for new input.</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</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>hostregex</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
Specify a regular expression on the source file path to set as the host field value for events from the input source.

<table>
<thead>
<tr>
<th>hostsegmentnum</th>
<th>No</th>
<th>Set the number of segments of the source file path to set as the host field value for events from the input source.</th>
</tr>
</thead>
<tbody>
<tr>
<td>follow-only</td>
<td>No</td>
<td>(T/F) True or False. Default False. When set to True, Splunk will read from the end of the source (like the &quot;tail -f&quot; Unix command).</td>
</tr>
</tbody>
</table>

Example 1. monitor files in a directory

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

Add /var/log/ as a data input:

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

Example 2. monitor windowsupdate.log

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

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

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

Example 3. monitor IIS logging

This example shows how to monitor the default location for Windows IIS logging.

Add C:\windows\system32\LogFiles\W3SVC as a data input:

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

Edit inputs.conf

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

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

Note: To ensure that new events are indexed when you copy over an existing file with new contents, set CHECK_METHOD = modtime in props.conf for the source. This checks the modtime of the file and re-indexes it when it changes. Be aware that the entire file will be re-indexed, which can result in duplicate events.
Configuration settings

There are separate stanza types for monitor and batch. See "Monitor files and directories" for detailed information about monitor and batch.

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

host = <string>

- Sets the host key/field to a static value for this stanza.
- Sets the host key's initial value. The key is used during parsing/indexing, in particular to set the host field. It is also the host field used at search time.
- The <string> is prepended with 'host::'.
- If not set explicitly, this defaults to the IP address or fully qualified domain name of the host where the data originated.

index = <string>

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

sourcetype = <string>

- Sets the sourcetype key/field for events from this input.
- Primarily used to explicitly declare the source type for this data, as opposed to allowing it to be determined via automated methods. This is typically important both for searchability and for applying the relevant configuration for this type of data during parsing and indexing.
- Sets the sourcetype key's initial value. The key is used during parsing/indexing, in particular to set the source type field during indexing. It is also the source type field used at search time.
- The <string> is prepended with 'sourcetype::'.
- If not set explicitly, Splunk picks a source type based on various aspects of the data. There is no hard-coded default.
- For more information about the sourcetype field, see "About default fields (host, source, sourcetype, and more)", in this manual.

queue = parsingQueue | indexQueue

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

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

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

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

host_regex = <regular expression>

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

host_segment = <integer>

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

Monitor syntax and examples

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

[monitor://<path>]
<attribute1> = <val1>
<attribute2> = <val2>
...

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

source = <string>

- Sets the source key/field for events from this input.
- **Note:** Overriding the source key is generally not recommended. Typically, the input layer will provide a more accurate string to aid 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.
- The <string> is prepended with 'source::'.
- Defaults to the input file path.

crcSalt = <string>

- Use this setting to force Splunk to consume files that have matching CRCs (cyclic redundancy checks). (Splunk only performs CRC checks against the first few lines of a file. This behavior prevents Splunk from indexing the same file twice, even though you may have renamed it -- as, for example, with rolling log files. However, because the CRC is based on only the first few lines of the file, it is possible for legitimately different files to have matching CRCs, particularly
if they have identical headers.)

- If set, string is added to the CRC.
- If set to <SOURCE>, the full source path is added to the CRC. This ensures that each file being monitored has a unique CRC.
- Be cautious about using this attribute with rolling log files; it could lead to the log file being re-indexed after it has rolled.
- **Note:** This setting is case sensitive.

**ignoreOlderThan = <time window>**

- Causes the monitored input to stop checking files for updates if their 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 are co-located with old files that are no longer being written to).
- **Note:** A file whose modtime falls outside <time window> when monitored for the first time will not get indexed.
- Value must be: <number><unit>. For example, "7d" indicates one week. Valid units are "d" (days), "m" (minutes), and "s" (seconds).
- Defaults to 0 (disabled).

**followTail = 0|1**

- If set to 1, monitoring begins at the end of the file (like tail -f).
- This only applies to files the first time they are picked up.
- After that, Splunk's internal file position records keep track of the file.
- Defaults to 0.

**whitelist = <regular expression>**

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

**blacklist = <regular expression>**

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

**alwaysOpenFile = 0 | 1**

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

**time_before_close = <integer>**

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

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

followSymlink

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

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

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

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

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

**Batch syntax and examples**

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

[batch://<path>]
move_policy = sinkhole
<attribute1> = <val1>
<attribute2> = <val2>
...

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

**Example:** This example batch loads all files from the directory /system/flight815/, but does not recurse through any subdirectories under it:

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

For details on using the asterisk in input paths, see "Specify input paths with wildcards".

**Specify input paths with wildcards**

This topic is only relevant when using inputs.conf to specify inputs, as described here.

**Important:** Input path specifications in inputs.conf don't use regex-compliant expressions but rather Splunk-defined wildcards.
A wildcard is a character that you can substitute for one or more unspecified characters when searching text or selecting multiple files or directories. In Splunk, you can use wildcards to specify your input path for monitored input.

<table>
<thead>
<tr>
<th>Wildcard</th>
<th>Description</th>
<th>Regex equivalent</th>
<th>Example(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>The ellipsis wildcard recurses through directories and any number of levels of subdirectories to find matches.</td>
<td>.*</td>
<td>/foo/.../bar matches the files /foo/bar, /foo/1/bar, /foo/2/bar, /foo/1/2/bar, etc.</td>
</tr>
<tr>
<td></td>
<td>Note: Because a single ellipse recurses through all directories and subdirectories, /foo/.../bar matches the same as /foo/.../.../bar</td>
<td></td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>The asterisk wildcard matches anything in that specific directory path segment. Unlike &quot;...&quot;, &quot;*&quot; doesn't recurse through any subdirectories.</td>
<td>[^/]*</td>
<td>/foo/*/bar matches the files /foo/bar, /foo/1/bar, /foo/2/bar, etc. However, it does not match /foo/1/2/bar.</td>
</tr>
<tr>
<td></td>
<td>/foo/m*r/bar matches /foo/mr/bar, /foo/mir/bar, /foo/moor/bar, etc.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></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>
<td></td>
</tr>
</tbody>
</table>

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

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

Warning: You cannot use a wildcard at the root level. For example, this does not work:

[monitor://E:\.\foo\*.log]

Input examples

To monitor /apache/foo/logs, /apache/bar/logs, /apache/bar/1/logs, etc.:

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

To monitor /apache/foo/logs, /apache/bar/logs, etc., 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/.*.log]
To monitor any file under /apache/ (under any level of subdirectory) that ends in .log:

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

**Wildcards and whitelisting**

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

When you specify wildcards in a file input path, Splunk creates an implicit whitelist for that stanza. The longest fully qualified path becomes the monitor stanza, and the wildcards are translated into regular expressions, as listed in the table above.

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

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

For example, if you specify

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

Splunk translates this into

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

**Important:** Because Splunk converts wildcards in the stanza heading into whitelists, it's important to be aware of a couple of issues:

- You cannot have multiple stanzas with wildcards for files in the same directory.
- A stanza with a wildcard in its heading cannot include an explicit whitelist.

These cases are dealt with in the following sections.

**Multiple stanzas with wildcards**

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

For example:

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

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

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

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

**Wildcards with explicit whitelists**

You cannot use a *whitelist* declaration in conjunction with input wildcards. Since Splunk translates the input wildcard into a whitelist, it will ignore any explicit whitelist in that stanza.

In this example, Splunk returns 0 events because the explicit whitelist gets ignored, due to the wildcard in the stanza header:

```
[monitor:///splunk/sybase/.../sysmon/]
disabled = false
host_segment = 3
sourcetype = sysmon
whitelist = sysmon*
```

Instead, you can combine the whitelist with the header (implicit) whitelist, like this:

```
[monitor:///splunk/sybase/.../sysmon/sysmon*]
disabled = false
host_segment = 3
sourcetype = sysmon
```

For more information on using whitelists with file inputs, see "Whitelist or blacklist specific incoming data".

**Whitelist or blacklist specific incoming data**

**Whitelist or blacklist specific incoming data**

Use *whitelist* and *blacklist* rules to explicitly tell Splunk which files to consume when monitoring directories. You can also apply these settings to batch inputs. When you define a *whitelist*, Splunk indexes only the files in that list. When you define a *blacklist*, Splunk ignores the files in that list and consumes everything else. You define whitelists and blacklists in the particular input’s stanza in inputs.conf.

You don’t have to define both a whitelist and a blacklist; they are independent settings. If you do define both and a file matches both of them, that file will not be indexed; blacklist will override whitelist.

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

Instead of whitelisting or blacklisting your data inputs, you can filter specific events and send them to different queues or indexes. Read more about routing and filtering data. You can also use the crawl feature to predefine files you want Splunk to index or not index automatically when they are added to your filesystem.
Important: Define whitelist and blacklist entries with exact regex syntax; the "..." wildcard used for input paths (described here) is not supported.

Whitelist (allow) files

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

whitelist = <your_custom regex>

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

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

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

whitelist = query.log$|my.log$

Or, to whitelist exact matches:

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

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

For information on how whitelists interact with wildcards in input paths, see "Wildcards and whitelisting".

Blacklist (ignore) files

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

blacklist = <your_custom regex>

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

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

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

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

[monitor:///mnt/logs]
    blacklist = .(txt|gz)$
If you want Splunk to ignore entire directories beneath a monitor input refer to this example:

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

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

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

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

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

**How log file rotation is handled**

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

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

**Example:**

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

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

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

**How Splunk recognizes log rotation**

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

There are three possible outcomes of a CRC check:

1. There is no begin and end CRC matching this file in the database. This indicates a new file. Splunk will pick it up and consume its data from the start of the file. Splunk updates the database with the
new CRCs and seekPtrs as the file is being consumed.

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

3. The begin CRC is present, but the end CRC does not match. This means that Splunk has previously read the file but that some of the material that it read has since changed. In this case, Splunk must re-read the whole file.

Important: Since the CRC start check is run against only the first 256 bytes of the file, 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 crcSalt attribute when configuring the file in inputs.conf, as described here. The crcSalt attribute ensures that each file has a unique CRC. You do not want to use this attribute with rolling log files, however, because it defeats Splunk’s ability to recognize rolling logs and will cause Splunk to re-index the data.
Get network events

Get data from TCP and UDP ports

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

TCP is the protocol underlying Splunk’s data distribution and is the recommended method for sending data from any remote machine to your Splunk server. Splunk can index remote data from syslog-ng or any other application that transmits via TCP.

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

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

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

Add a network input using Splunk Web

To add inputs from network ports using Splunk Web:

A. Go to the Add New page

You add a network input from the Add New page in Splunk Web. You can get there through two routes:

- Splunk Manager
- Splunk Home

It doesn't matter which route you use to get there; the Add New page itself is the same either way.

Via Splunk Manager:

1. Click Manager in the upper right-hand corner of Splunk Web.
2. In the Data section of the Manager page, click Data inputs.
3. Pick TCP or UDP.
4. Click the New button to add an input.

Via Splunk Home:
1. Click the Add Data link in Splunk Home. This brings you to a page called "Data recipes".

2. Click either the From a TCP port or the From a UDP port link to add an input.

B. Specify the network input

1. Enter a port number. The user you run Splunk as must have access to the port. On a Unix system, you must run as root to access a port under 1024.

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

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

   Important: Consult Splunk support before changing this value.

4. To access other settings, check More settings. A number of additional settings appear. You can usually go with the defaults for these settings. If you want to set them explicitly, here’s what they’re for:

   a. You can set the Host by selecting a radio button:

      • IP. Sets the input processor to rewrite the host with the IP address of the remote server.

      • DNS. Sets the host to the DNS entry of the remote server.

      • Custom. Sets the host to a user-defined label.

   b. You can set the Source type. Source type is a default field added to events. Source type is used to determine processing characteristics, such as timestamps and event boundaries. For information on overriding Splunk’s automatic source typing, see "Override automatic source type assignment" in this manual.

   c. You can set the Index. Leave the value as "default" unless you have defined multiple indexes to handle different types of events. In addition to indexes for user data, Splunk has a number of utility indexes, which also appear in this dropdown box.

5. Click Save.

Add a network input using the CLI

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

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

The following CLI commands are available for network input configuration:
<table>
<thead>
<tr>
<th>Command</th>
<th>Command syntax</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>add tcp</td>
<td>udp $SOURCE [-parameter value] ...</td>
</tr>
<tr>
<td>edit</td>
<td>edit tcp</td>
<td>udp $SOURCE [-parameter value] ...</td>
</tr>
<tr>
<td>remove</td>
<td>remove tcp</td>
<td>udp $SOURCE</td>
</tr>
<tr>
<td>list</td>
<td>list tcp</td>
<td>udp</td>
</tr>
</tbody>
</table>

Change the configuration of each data input type by setting additional parameters. Parameters are set with this syntax:

```
-<parameter> <value>
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SOURCE</td>
<td>Yes</td>
<td>Port number to listen for data to index. The user you run Splunk as must have access to this port. On a Unix system, you must run as root to access a port under 1024.</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</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 True of False (T</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's host value via DNS. Use `auth` with your username and password:

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

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

**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 uses the defaults that are preset in
$SPLUNK_HOME/etc/system/default/ (noted below).

TCP

[tcp://<remote server>:<port>]
<attribute1> = <val1>
<attribute2> = <val2>
...

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

Note: The user you run Splunk as must have access to the listening port. On a Unix system, you must run as root to access a port under 1024.

host = <string>

• Sets the host key/field to a static value for this stanza.
• Sets the host key's initial value. The key is used during parsing/indexing, in particular to set the host field. It is also the host field used at search time.
• The <string> is prepended with 'host::'.
• If not set explicitly, this defaults to the IP address or fully qualified domain name of the host where the data originated.

index = <string>

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

sourcetype = <string>

• Sets the sourcetype key/field for events from this input.
• Primarily used to explicitly declare the source type for this data, as opposed to allowing it to be determined via automated methods. This is typically important both for searchability and for applying the relevant configuration for this type of data during parsing and indexing.
• Sets the sourcetype key's initial value. The key is used during parsing/indexing, in particular to set the source type field during indexing. It is also the source type field used at search time.
• The <string> is prepended with 'sourcetype::'.
• If not set explicitly, Splunk picks a source type based on various aspects of the data. There is no hard-coded default.
• For more information about the sourcetype field, see "About default fields (host, source, sourcetype, and more)", in this manual.

source = <string>

• Sets the source key/field for events from this input.
• **Note:** Overriding the source key is generally not recommended. Typically, the input layer will provide a more accurate string to aid 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.

- The `<string>` is prepended with 'source::'.
- Defaults to the input file path.

```plaintext
queue = [parsingQueue|indexQueue]
```

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

```plaintext
connection_host = [ip|dns|none]
```

- "ip" sets the host to the IP address of the remote server.
- "dns" sets the host to the DNS entry of the remote server.
- "none" leaves the host as specified.
- Defaults to "ip".

**TCP over SSL**

```plaintext
[tcp-ssl:<port>]
```

Use this stanza type if you are receiving encrypted, unparsed data from a forwarder or third-party system. Set `<port>` to the port on which the forwarder or third-party system is sending unparsed, encrypted data.

**UDP**

```plaintext
[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.

**Note:**

- If `<remote server>` is specified, the specified port will only accept data from that server.
- If `<remote server>` is empty - [udp://<port>] - the port will accept data sent from any server.

```plaintext
host = <string>
```

- Sets the host key/field to a static value for this stanza.
- Sets the host key's initial value. The key is used during parsing/indexing, in particular to set the host field. It is also the host field used at search time.
- The `<string>` is prepended with 'host::'.

34
• If not set explicitly, this defaults to the IP address or fully qualified domain name of the host where the data originated.

index = <string>

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

sourcetype = <string>

• Sets the sourcetype key/field for events from this input.
• Primarily used to explicitly declare the source type for this data, as opposed to allowing it to be determined via automated methods. This is typically important both for searchability and for applying the relevant configuration for this type of data during parsing and indexing.
• Sets the sourcetype key's initial value. The key is used during parsing/indexing, in particular to set the source type field during indexing. It is also the source type field used at search time.
• The <string> is prepended with 'sourcetype::'.
• If not set explicitly, Splunk picks a source type based on various aspects of the data. There is no hard-coded default.
• For more information about the sourcetype field, see "About default fields (host, source, sourcetype, and more)", in this manual.

source = <string>

• Sets the source key/field for events from this input.
• Note: Overriding the source key is generally not recommended. Typically, the input layer will provide a more accurate string to aid 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.
• The <string> is prepended with 'source::'.
• Defaults to the input file path.

queue = [parsingQueue|indexQueue]

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

_rcvbuf = <integer>

• Specify the receive buffer for the UDP port, measured in bytes.
• If the value is 0 or negative, it is ignored.
• Defaults to 1,572,864.
• Note: If the default value is too large for an OS, Splunk will try to set the value to 1572864/2. If that value also fails, Splunk will retry with 1572864/(2*2). It will continue to retry by halving the value until it succeeds.
no_priority_stripping = [true|false]

- Setting for receiving syslog data.
- If this attribute is set to true, Splunk does NOT strip the <priority> syslog field from received events.
- **Note:** Do NOT include this attribute if you want to strip <priority>.
- Default is false (Splunk strips <priority>).

no_appending_timestamp = [true|false]

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

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,

# Send SNMP events to Splunk

**Send SNMP events to Splunk**

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

**On UNIX**

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

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

The simplest configuration is:

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

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

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

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

On Windows

To log SNMP traps to a file on Windows:

1. **Install** **NET-SNMP** from [http://www.net-snmp.org/](http://www.net-snmp.org/)

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

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

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

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

**MIBs**

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

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

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

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

   - Adding a leading '+' character will load the MIB in addition to the default list, instead of overwriting the list.
   - The special keyword **ALL** is used to load all MIB modules in the MIB directory.

The safest argument seems to be: `-m +ALL`:

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

Considerations for deciding how to monitor remote Windows data

Splunk is capable of powerful indexing, searching and reporting of your remote Windows data. However, as with any software, getting the best out of Splunk requires careful planning and execution. This is particularly important for remote operations - the goal is to get the most amount of data, using the least amount of resource overhead. This topic helps you decide how and where to deploy Splunk components for maximum efficiency when indexing data remotely from a large number of Windows hosts.

Remote Windows data overview

Splunk indexes remote Windows data in one of several ways:

Using WMI and remote authentication

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

This configuration:

- Gives Splunk as much access to the network as the specified account has for remote access.
- Lets Splunk 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 Splunk indexer in each network segment.

There are some caveats to this method of collection, however. They are explained in the "Forwarders versus remote collection through WMI" section of this topic.

Note: While Active Directory (AD) monitoring does not use WMI, it has the same authentication considerations as data inputs that do. For more information on how Splunk monitors AD, see the topic "Monitor Active Directory" in this manual.

Using a universal forwarder

Splunk can also collect remote Windows data remotely with a forwarder. There are several types of forwarders: light, heavy and universal. You can read more about the universal forwarder in "Introducing the universal forwarder" in the Distributed Deployment Manual.

There are some advantages to using a Splunk universal forwarder to get Windows data:
Splunk recommends using it to gather remote Windows data whenever possible.

- If the forwarder is installed as the Local System user, then it has access to everything on the machine on which it's installed, and requires no authentication to get data, as WMI does.

- You can deploy universal forwarders manually, by using either a Microsoft deployment tool such as System Center Configuration Manager (SCCM) or Systems Management Server (SMS), or a third party distribution solution such as BigFix/Tivoli.

Once the forwarder is installed, it gathers data you specify by using either information provided during the installation process or configuration files. You can then manage their configurations by using Splunk's deployment server.

There are some drawbacks to using the universal forwarder, depending on your network configuration and layout. More discussion on these trade-offs is available in the "Forwarders versus remote collection through WMI" section of this topic.

Polling data remotely over WMI? Be sure to read this

When collecting remote Windows data over WMI, you'll need to consider some important common issues associated with doing so.

Authentication

Windows requires authentication for just about every kind of remote activity it performs. While this is well understood by system administrators, it's a particularly important concept to grasp when using Splunk in a Windows environment. Failure to understand how Splunk interacts with Windows over the network can lead to suboptimal search results, or no results at all. This section explains the various security-related factors surrounding getting remote Windows data into Splunk.

When you install Splunk, you can specify the Local System user, or another user. More specific information about the installation ramifications of this choice can be found in the Installation Manual, but there are consequences for getting data in as well.

The user you tell Splunk to run as has a direct impact on the amount and available types of data that is retrievable from remote machines. In order to get the data you want, you'll need to give this user the permissions needed to properly access remote data sources.

The easiest way to do so is to make the user that Splunk runs as a member of the Administrators (or Domain Admins) groups. But security concerns make this choice problematic and, depending on how your Active Directory is configured, you may not even have permission to do this.

In most cases, you'll need to configure the Splunk user account to have a minimum level of permissions needed to access the data source. This is called "least-permissive" access, and it 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 is configured to enforce password changes regularly, you'll also need to:

- Make sure that either the Splunk account's password is set to never expire, or that you manually change the password before it expires, as defined by the policy.
- Restart Splunk services that run as that account on all servers in your network, once the password is changed.

It's also a good idea to assign the Splunk account the "Deny log on locally" user rights assignment in Local Security Policy to prevent the user from logging in interactively.

While this method is initially more time-consuming, it gives you more control, and is overall much more secure than handing out domain administrator access.

Individual topics in this manual that deal with remote access to Windows machines will have additional information and recommendations on how to configure the user Splunk runs as for least-permissive access. Review the Security and remote access considerations topic on those pages for specifics.

Bandwidth

Pure I/O bandwidth is more important when performing Splunk indexing on local machines, but remote data collection activities also require consideration.

Network bandwidth usage should be monitored closely, especially in networks with slow or thin WAN links. For this reason alone, in many cases, universal forwarders are a better choice for large remote collection operations.

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

Forwarders versus remote collection through WMI

The recommended way to get data in from a remote Windows host is with a forwarder. Beginning in version 4.2, Splunk introduced the universal forwarder specifically for this purpose.

Using a forwarder offers the most types of data sources, minimizes network overhead, and reduces operational risk and complexity, particularly where authentication is concerned. It's also more scalable than WMI in many cases.

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

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

Data Source Considerations
<table>
<thead>
<tr>
<th>Data Source</th>
<th>Local Forwarder</th>
<th>Remote Polling</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 wish to collect. On local forwarders, you have the option to collect all logs, regardless of name.

** Remote log file collection using the "\SERVERNAME\SHARE" syntax is supported, however you must use CIFS (Common Internet File System, or Server Message Block) as your application layer file access protocol and Splunk must have at least read access to both the share and the underlying file system.

Tradeoffs

Performance

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

- You are collecting local event logs or flat files. This is because a forwarder requires less CPU and performs basic pre-compression 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 a forwarder is installed as the Local System user, it has administrative access to the machine on which it is installed. This allows you to collect any available data from that machine.

- You are collecting data from busy hosts such as domain controllers, AD operations masters, or machines that consistently experience periods of high resource utilization. This includes servers that run resource-intensive applications such as Exchange, SQL Server/Oracle, VMWare/VSphere/VCenter, Hyper-V, or SharePoint. This is because WMI may have problems keeping up with the amount of data that is generated. WMI polling is best-effort by design, and Splunk also throttles WMI calls to prevent unintentional denial-of-service attacks.

- You are concerned about CPU and network utilization on the machines from which you’re collecting data. Forwarders are designed to use as little of these resources as possible, while WMI is more CPU intensive for the same set of data. WMI also uses more network resources to transfer that data to the collecting machine.

- You are concerned about scalability. Universal forwarders scale very well. Heavy forwarders do not scale as well as universal forwarders do, but both types of forwarder scale considerably better than WMI.

WMI is a better choice when:
• You are concerned 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 collected requires transformation of any kind.

Note: You cannot use a universal forwarder to transform data. You must use a heavy forwarder in that case.

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’s configuration later, or at mass deployment time.

Management

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

Search Windows data on a non-Windows instance of Splunk

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

There are two main ways to proceed:

• Set up forwarders locally on each Windows machine from which you want data. These forwarders can send the Windows data to the non-Windows receiving instance of Splunk.

• 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.

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

For information on setting up forwarders, see "Set up forwarding and receiving" also in the Distributed Deployment Manual.

**Monitor Active Directory**

Active Directory (AD) is an integral part of any Windows network. The Active Directory database (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. When you make a change to Active Directory, such as adding or deleting a user, member server or domain controller, those changes are recordable. Splunk lets you alert and monitor those changes in real time.

You can configure AD monitoring to watch 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.

Once you've configured Splunk to monitor your Active Directory, it will take a baseline snapshot of your AD data and the AD schema. It will use this snapshot to establish a starting point against which to monitor. This process might take a little time before it completes.

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

**Why monitor Active Directory?**

If you are charged with maintaining the integrity, security and health of your Active Directory, then you will want to know what is happening with it day to day. Splunk allows you to see what has changed in your AD, who or what made the changes, and when they were made.

You can transform this data into reports for corporate security compliance or forensics. 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 domain controllers (DCs).

**Considerations for monitoring Active Directory**

To get the best results from monitoring AD, be aware of the following:

- This feature is only available with Splunk on Windows. You won't be able to monitor AD changes from a *nix version of Splunk (though you can forward AD data gathered from a Windows version of Splunk to a *nix indexer).
- The AD monitoring process can run under a full Splunk instance or within any kind of forwarder.
• The machine that's monitoring changes to AD must belong to the domain or forest you want to monitor.
• The user Splunk runs as must be part of the domain too. This is because the permissions that the user has determine what parts of AD Splunk can monitor.

For additional information on deciding how to monitor Windows data remotely, see "Considerations for deciding how to monitor remote Windows data" in this manual. For information on deciding which user Splunk should run as at installation time, review "Choosing the user Splunk should run as" in the Installation Manual.

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

1. Click Manager in the upper right-hand corner of Splunk Web.
2. Under "Data", click Data Inputs.
3. Click Active Directory monitoring.
4. Click New to add an input.
5. Enter a unique name for the AD monitor input.
6. Either supply a Target domain controller or leave the field blank to tell Splunk to discover the nearest domain controller automatically.
7. Either select a Starting node or leave the field blank to tell Splunk to monitor AD objects from the highest available part of the directory tree.

Note: Review "Considerations on how to monitor remote Windows data" to learn how the user Splunk runs as affects what Splunk can see.
8. Select Monitor subtree if you want Splunk to monitor the child nodes below the starting node you specified in Step 7.
9. Choose the destination index for this input.
10. Click Save.

Splunk adds and enables the input.

Configure AD monitoring with configuration files

Active Directory monitoring configurations are controlled by inputs.conf and admon.conf. Be sure to edit copies of these configuration files in the %SPLUNK_HOME%\etc\system\local directory. If you edit them in the default directory, any changes you make will be overwritten when you upgrade
Splunk. For more information about configuration file precedence, refer to "About configuration files" in this manual.

1. Make a copy of %SPLUNK_HOME%/etc/system/default/inputs.conf and put it in %SPLUNK_HOME%/etc/system/local/inputs.conf.

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

3. Make a copy of %SPLUNK_HOME%/etc/system/default/admon.conf and put it in %SPLUNK_HOME%/etc/system/local/admon.conf.

4. Use the "Settings in admon.conf" section below to make edits to admon.conf.

**Note:** By default, when AD monitoring inputs are enabled, Splunk will gather AD change data from the first domain controller that it can attach to. If that is acceptable, no further configuration is necessary.

**admon.conf settings**

**admon.conf** contains one stanza for each AD monitoring input. In each stanza, you specify:

**targetDC:** The unique name of the domain controller host you want Splunk to use for AD monitoring.

Specify a unique name for this attribute if:

- You have a very large AD and you only want to monitor information in a particular Organizational Unit (OU), subdomain, etc.
- You have a specific (read-only) domain controller that is offered for monitoring purposes in a high security environment.
- You have multiple domains or forests in with transitive trusts established, and want to use this to target a different tree than the one where Splunk resides.
- You want to configure multiple AD monitoring inputs to target multiple domain controllers. For example, to monitor AD replication across a distributed environment.

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

**startingNode:** A fully qualified Lightweight Directory Access Protocol (LDAP) name (for example "LDAP://OU=Computers,DC=ad,DC=splunk,DC=com") that specifies where Splunk should begin its indexing. Splunk starts there and enumerates down to sub-containers, depending on the configuration of the monitorSubtree attribute. If startingNode is not specified or is blank, Splunk will start monitoring AD data from the highest root domain in the tree it can access.

**Note:** The value of startingNode must be within the scope of the DC you are targeting in order for Splunk to successfully get AD data.
monitorSubtree: How much of the target AD container to index. A value of 0 tells Splunk to index only the target container, and not traverse into subcontainers within that container. A value of 1 tells Splunk to enumerate all sub-containers and domains that it has access to. The default is 1.

index: The index to route AD monitoring data to. If not present, the 'default' index is used.

disabled: Whether or not the input is enabled. A value of 0 tells Splunk that the input is enabled, and a value of 1 tells Splunk that the input is disabled. The default is 0 (enabled).

Example AD monitoring configurations

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

To index data from the top of the AD directory:

# Gather all AD data that this server can see

[default]
monitorSubtree = 1
disabled = 0

[NearestDC]
TargetDC =
startingNode =

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

[default]
monitorSubtree = 1
disabled = 0

# 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.

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

To monitor multiple domain controllers:

[default]
monitorSubtree = 1
disabled = 0

# 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.

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

[SecondTargetDC]
targetDC = pri02.eng.ad.splunk.com
Sample AD monitoring output

When Splunk's AD monitoring utility runs, it gathers AD change events. Each change event is indexed as an event in Splunk. You can view these events as they come into Splunk in the Search app.

There are 5 types of AD change events that Splunk can index. Examples of these events are detailed below. Some of the content of these events has been obscured/altered for publication purposes.

Update event

When an AD object is changed in any way, Splunk generates this type of event. Splunk 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
displayName=stuff2
distinguishedName=CN=stuff2,CN=Computers

Object Details:
sAMAccountType=805306369
sAMAccountName=stuff2
logonCount=4216
accountExpires=9223372036854775807
objectSid=S-1-5-21-3436176729-1841096389-3700143990-1190
primaryGroupID=515
pwdLastSet=129091141316250000
lastLogon=129095398380468750
lastLogoff=0
badPasswordTime=0
countryCode=0
codePage=0
badPwdCount=0
userAccountControl=4096
objectGUID=blah
whenChanged=20100128010211.0Z
whenCreated=20081125172950.0Z
objectClass=top|person|organizationalPerson|user|computer

Event Details:
  uSNChanged=2921916
  uSNCreated=1679623
  instanceType=4

Additional Details:
  isCriticalSystemObject=FALSE
  servicePrincipalName=TERMSRV/stuff2|TERMSRV blah
dNSHostName=stuff2.splunk.com
operatingSystemServicePack=Service Pack 2
operatingSystemVersion=6.0 (6002)
operatingSystem=Windows Vista? Ultimate
```
Delete event

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

2/1/10
3:11:16.095 PM

02/01/2010 15:11:16.0954
dcName=stuff.splunk.com
admonEventType=Update
Names:
  name=SplunkTest
DEL:blah
distinguishedName=OU=SplunkTest\0ADEL:blah,CN=Deleted Objects
DEL:blah
Object Details:
  objectGUID=blah
  whenChanged=20100128233113.0Z
  whenCreated=20100128232712.0Z
  objectClass=top|organizationalUnit
Event Details:
  uSNChanged=2922895
  uSNCreated=2922846
  instanceType=4
Additional Details:
  dSCorePropagationData=20100128233113.0Z|20100128233113.0Z|20100128233113.0Z|16010108151056.0Z
  lastKnownParent=stuff
  '''isDeleted=TRUE'''

Sync event

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

Note: When you restart either Splunk or the `splunk-admon.exe` process, Splunk will log an extra 'sync' event. This is normal.

2/1/10
3:11:09.074 PM

02/01/2010 15:11:09.0748
dcName=ftw.ad.splunk.com
admonEventType=Sync
Names:
  name=NTDS Settings
distinguishedName=CN=NTDS Settings,CN=stuff,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration
  cn=NTDS Settings
  objectCategory=CN=NTDS-DSA,CN=Schema,CN=Configuration,DC=ad,DC=splunk,DC=com
When Splunk is started after configured 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.
dSCorePropagationData=OptionalProperties
extensionName=OptionalProperties
flags=OptionalProperties
fromEntry=OptionalProperties
frsComputerReferenceBL=OptionalProperties
fRSMemberReferenceBL=OptionalProperties
fSMORoleOwner=OptionalProperties
heuristics=OptionalProperties
isCriticalSystemObject=OptionalProperties
isDeleted=OptionalProperties
isPrivilegeHolder=OptionalProperties
lastKnownParent=OptionalProperties
legacyExchangeDN=OptionalProperties
managedObjects=OptionalProperties
masteredBy=OptionalProperties
memberOf=OptionalProperties
modifyTimeStamp=OptionalProperties
mS-DS-ConsistencyChildCount=OptionalProperties
mS-DS-ConsistencyGuid=OptionalProperties
msCOM-PartitionSetLink=OptionalProperties
msCOM-UserLink=OptionalProperties
msDFSR-ComputerReferenceBL=OptionalProperties
msDFSR-MemberReferenceBL=OptionalProperties
msDS-Approx-Immed-Subordinates=OptionalProperties
msDs-masteredBy=OptionalProperties
msDS-MembersForAzRoleBL=OptionalProperties
msDS-NCRepl1Cursors=OptionalProperties
msDS-NCReplInboundNeighbors=OptionalProperties
msDS-NCReplOutboundNeighbors=OptionalProperties
msDS-NonMembersBL=OptionalProperties
msDS-ObjectReferenceBL=OptionalProperties
msDS-OperationsForAzRoleBL=OptionalProperties
msDS-OperationsForAzTaskBL=OptionalProperties
msDS-Rep1AttributeMetaData=OptionalProperties
msDS-Rep1ValueMetaData=OptionalProperties
msDS-TasksForAzRoleBL=OptionalProperties
msDS-TasksForAzTaskBL=OptionalProperties
msExchADCGlobalNames=OptionalProperties
msExchALObjectVersion=OptionalProperties
msExchHideFromAddressLists=OptionalProperties
msExchInconsistentState=OptionalProperties
msExchIPAddress=OptionalProperties
msExchTurfList=OptionalProperties
msExchUnmergedAttsPt=OptionalProperties
msExchVersion=OptionalProperties
msSFU30PosixMemberOf=OptionalProperties
name=OptionalProperties
netbootSCPBL=OptionalProperties
nonSecurityMemberBL=OptionalProperties
objectGUID=OptionalProperties
objectVersion=OptionalProperties
otherWellKnownObjects=OptionalProperties
ownerBL=OptionalProperties
partialAttributeDeletionList=OptionalProperties
partialAttributeSet=OptionalProperties
possibleInferiors=OptionalProperties
proxiedObjectName=OptionalProperties
proxyAddresses=OptionalProperties
queryPolicyBL=OptionalProperties
replicatedObjectVersion=OptionalProperties
replicationSignature=OptionalProperties
replPropertyMetaData=OptionalProperties
replUpToDateVector=OptionalProperties
repsFrom=OptionalProperties
repsTo=OptionalProperties
revision=OptionalProperties
sDRightsEffective=OptionalProperties
serverReferenceBL=OptionalProperties
showInAddressBook=OptionalProperties
showInAdvancedViewOnly=OptionalProperties
siteObjectBL=OptionalProperties
structuralObjectClass=OptionalProperties
subRef=OptionalProperties
subSchemaSubEntry=OptionalProperties
systemFlags=OptionalProperties
unmergedAtts=OptionalProperties
url=OptionalProperties
uSNChanged=OptionalProperties
uSNCreated=OptionalProperties
uSNDSALastObjRemoved=OptionalProperties
USNIntersite=OptionalProperties
uSNLastObjRem=OptionalProperties
uSNSource=OptionalProperties
wbemPath=OptionalProperties
wellKnownObjects=OptionalProperties
whenChanged=OptionalProperties
whenCreated=OptionalProperties
wWWHomePage=OptionalProperties

Answers

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

Monitor Windows event log data

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 - intermediate locations that eventually get written to an event log file. Programs such as Microsoft's Event Viewer subscribe to these log channels to display events that have occurred on the system.

Splunk also supports the monitoring of Windows event log channels. It can monitor event log channels and files stored on the local machine, and it can collect logs from remote machines.

Splunk's event log monitor runs as an input processor within the splunkd service. It runs once for every event log input defined in Splunk.
Security and remote access considerations

Splunk collects event log data from remote machines using either WMI or a forwarder. Splunk recommends using a universal forwarder to send event log data from remote machines to an indexer. Review "Introducing the universal forwarder" in the Distributed Deployment Manual for information about how to install, configure and use the forwarder to collect event log data.

If you choose to install forwarders on your remote machines to collect event log 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 want Splunk to use WMI to get event log data from remote machines, then you must ensure that your network and Splunk instances are properly configured. You cannot install Splunk as the Local System user, and the user you install with determines the set of performance metrics Splunk will see. Review "Security and remote access considerations" in the "Monitor WMI Data" topic in this manual for additional information on the requirements you must satisfy in order for Splunk to collect remote data properly using WMI.

By default, access to some Windows event logs is restricted, depending on which version of Windows you're running. In particular, the Security event logs by default can only be read by members of the local Administrators or global Domain Admins groups.

Collecting event logs from a remote Windows machine

If you want Splunk to collect event logs from a remote machine, you have two choices:

- Collect the logs remotely using WMI. When you select "Remote event log collections" in Splunk Web, you are using this option.
- Install a universal forwarder on the machine from which you want to collect logs.

If you choose to collect event logs using WMI, you must install Splunk with an Active Directory domain user. Refer to "Considerations for deciding how to monitor remote Windows data" for additional information on collecting data from remote Windows machines. 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 server from which you are collecting event logs.
- Understand how the Security Description Definition Language (SDDL) (external link) works, and how to assign permissions with it.

For instructions on how to configure event log security permissions on Windows XP and Windows Server 2003/2003 R2, review this Microsoft Knowledge Base article. If you're running Windows Vista, Windows 7 or Windows Server 2008/2008 R2, use the `wevtutil` utility to set event log security.

Use Splunk Web to configure event log monitoring
Configure local event log monitoring

1. Click Manager in the upper right-hand corner of Splunk Web.
2. Under Data, click Data Inputs.
3. Click Local event log collections.
4. Click Add new to add an input.
5. Select one or more logs from the list of Available Logs and click to add to the list of Selected Logs.

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

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

6. Click Save.

Splunk adds and enables the input.

Configure remote event log monitoring

1. Click Manager in the upper right-hand corner of Splunk Web.
2. Under Data, click Data Inputs.
3. Click Remote event log collections.
4. Click Add new to add an input.
5. Enter a unique name for this collection.
6. Specify a hostname or IP address for the host from which to pull logs, and click Find logs... to get a list of logs from which to choose.

Note: Windows Vista offers many event log channels in addition to the standard set of channels defined in all versions of Windows. Depending on the CPU available to Splunk, selecting all or a large number of them can result in high load.

7. Optionally, provide a comma-separated list of additional servers from which to pull data.
8. Click Save.
Splunk adds and enables the input.

Use inputs.conf to configure event log monitoring

You can edit inputs.conf to configure event log monitoring. For more information on configuring data inputs with inputs.conf, read "Configure your inputs" in this manual.

To enable event log inputs by editing inputs.conf:

1. Copy inputs.conf from %SPLUNK_HOME%\etc\system\default to etc\system\local.
2. Use Explorer or the ATTRIB command to remove the file's "Read Only" flag.
3. Open the file and edit it to enable Windows event log inputs.
4. Restart Splunk.

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

Event log monitor configuration values

Windows event log (*.evt) files are in binary format. They can't be monitored like 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 uses the following stanzas in inputs.conf to monitor the default Windows event logs:

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

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

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

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

If you want to specify whether or not Active Directory objects like globally unique identifiers (GUIDs) and security identifiers (SIDs) are resolved for a given Windows event log channel, you can 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:

```
[WinEventLog:Security]
disabled = 0
start_from = oldest
current_only = 0
evt_resolve_ad_obj = 1
checkpointInterval = 5
```

To specify a domain controller for the domain that Splunk should bind to 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's 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
```

Specify whether to index starting at earliest or most recent event

Use the `start_from` attribute to specify whether Splunk indexes events starting at the earliest event or the most recent. By default, Splunk starts with the oldest data and indexes forward. You can change this by setting this attribute to `newest`, telling Splunk to start with the newest data, and index backward. We don't recommend changing this setting, as it results in a highly inefficient indexing process.
Use the `current_only` attribute to specify whether or not you want Splunk to index all preexisting events in a given log channel. When set to 1, Splunk indexes only new events that appear from the moment Splunk was started. When set to 0, Splunk indexes all events.

For example:

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

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.

Constraints

- As a result of API and log channel processing constraints on Windows XP and Server 2003 systems, imported .evt files from those systems will not contain the "Message" field. This means that the contents of the "Message" field will not appear in your Splunk index.
- Splunk running on Windows Vista, 7, and Server 2008/2008 R2 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.
- The language that a .evt or .evtx file will be indexed as is the primary locale/language of the Splunk computer that collects the file.

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

Note: When producing .evt or .evtx files on one system, and monitoring them on another, it's possible that not all of the fields in each event will 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.

Answers

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

Monitor WMI-based data

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

**Important:** If it is possible for you to do so, Splunk recommends the use of 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. In particular, consider a forwarder if you are collecting multiple event logs or performance counters from each host, or from very busy hosts like domain controllers. Review "Considerations for deciding how to monitor remote Windows data" in this manual for additional information.

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

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

**Security and remote access considerations**

Before attempting to use Splunk to get data using WMI providers, you'll need to meet the following prerequisites:

- Splunk must be installed as a user with permissions to perform remote connections.
- The user Splunk 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 is running Splunk.
- The computer that is running Splunk 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 server making the query and the target servers being queried must be part of the same AD domain or forest.

The Splunk user 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 in order to configure access for your Splunk user. If you don't have domain administrator access, you must find someone who can either configure Splunk user access for you or give domain administrator rights to you.

**Note:** If you installed Splunk as the Local System user, remote authentication over WMI will not work. The Local System user has no credentials to any other machines on the network. It is not possible to grant privileges to a machine's Local System account for access to another machine.

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 server 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).
Important notice regarding group memberships and resource access control lists (ACLs)

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

Configure WMI for least permissive access

If the user you've configured Splunk to run as is not a domain administrator, you must configure WMI to provide access to this user. Splunk strongly recommends granting 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, refer to "How to enable WMI access for non-administrator domain users" in the Splunk Community Wiki.

There are several levels of access you must grant to the user Splunk runs as in order for Splunk to collect data over WMI using the least-permissive method:

1. **Local Security Policy Permissions.** The Splunk role account needs the following Local Security Policy user rights assignments defined on each machine you poll:
   - 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

   **Note:** To deploy these user rights assignments domain-wide, use the Domain Security Policy (dompol.msc) Microsoft Management Console (MMC) snap-in; those rights assignments will be inherited by any member servers on the network during the next AD replication cycle (though you must restart Splunk instances on those servers 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.

2. **Distributed Component Object Model (DCOM) configuration and permissions.** DCOM must be enabled on every machine you want to poll. In addition, the Splunk 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 server you want to poll, and then add the Splunk user to the "Distributed COM Users" domain global group. There are also a number of advanced ways to ensure that the Splunk user has access to DCOM. Review "Securing a Remote WMI Connection" (http://msdn.microsoft.com/en-us/library/aa393266(VS.85).aspx) on MSDN for additional details.

3. **Performance Monitor configuration and permissions.** In order for Splunk to access remote performance objects over WMI, the user it runs as must be a member of the Performance Log Users local group on every member server you want to poll. 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 server and then assign the Splunk user to the global group.
4. WMI namespace security. The WMI namespace that Splunk accesses (most commonly Root\CIMV2) must have the proper permissions set. These permissions must be set on each server 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

These rights must be assigned to the Root namespace and all subnamespaces below it.

See the Microsoft Knowledge Base article "HOW TO: Set WMI Namespace Security in Windows Server 2003" (http://support.microsoft.com/kb/325353) in the Microsoft Knowledgebase for more information.

**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) on MSDN Blogs offers instructions on how to create a startup script that you can place inside a Group Policy Object (GPO), which will set the namespace security once the GPO is applied to the desired hosts. You can then deploy this GPO domain-wide or to one or more Organizational Units (OUs).

5. Firewall configuration. If you have a firewall enabled, you must configure it to allow access for WMI. If you are using the Windows Firewall included with Windows XP, Windows Server 2003/2003 R2, Windows Vista, Windows 7 and Windows Server 2008/2008 R2, 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**

Once you've configured WMI and set up the Splunk user for access to your domain, you should test the configuration and access to the remote machine.

**Important:** This procedure includes steps to temporarily change Splunk's 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 is run.

To test access to WMI providers:

1. Log into the machine Splunk runs on as the Splunk user.
Note: If attempting to log into a domain controller, you may have to change your domain controller security policy to assign the "Allow log on locally" policy for the designated user.

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

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

4. Determine where Splunk is storing its data by running the following command:

   > splunk show datastore-dir

Note: You'll need to authenticate into your Splunk instance in order to do this. Once you have, be sure to note where Splunk is storing its data. You'll need to remember it for later.

5. Run the following command to change where Splunk 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 by running the following command:

   > splunk restart

Note: It may take a while for Splunk to restart. This is because it's creating a new data store at the area you specified in Step 5.

7. Once Splunk has restarted, test access to WMI providers by running the following command, replacing <server> with the name of the remote server:

   > splunk cmd splunk-wmi -wql "select * from win32_service" -namespace \\<server>\root\cimv2

8. If you see data streaming back and no error messages, that means Splunk was able to connect to the WMI provider and query successfully.

9. If there is an error, a message with a reason on what caused the error will appear. Look for the error="<msg>" string in the output for clues on how to correct the problem.

After testing WMI access, be sure to point Splunk back to the correct Splunk database directory by running the following command, and then restarting Splunk:

   > splunk set datastore-dir <directory shown from Step 4>
Configure WMI-based inputs

**Note:** Beginning with version 4.2, the procedure for adding WMI-based inputs has changed. There’s no longer a **WMI collections** input available under "Data inputs" in Manager. You now access the WMI inputs through either the **Remote event log collections** or **Remote performance monitoring** data input types.

All remote data collection in Splunk on Windows is done through either WMI providers or a forwarder. Review "Considerations for deciding how to monitor remote Windows data" in this manual for additional information about remote data collection.

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, consult one of the appropriate topics in this manual:

- Configure remote Windows performance monitoring with Splunk Web
- Configure remote Windows event log monitoring

**Configure WMI-based inputs with configuration files**

Remote data collection configurations are controlled by wmi.conf. 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 `<code>wmi.conf` in `%SPLUNK_HOME%/etc/system/local/`. Only set values for the attributes you want to change for a given type of data input. Refer to About configuration files in the Admin Manual for more information about how Splunk uses configuration files.

**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 missing, Splunk assumes system defaults.

When Splunk is not able to connect to a defined WMI provider, an error is generated in splunkd.log. For example:

```
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 reconnects to a given WMI provider when an error occurs:

- **initial_backoff**: Tells Splunk how long, in seconds, to wait the first time after an error occurs before trying to reconnect to the WMI provider. The default is 5. If connection errors continue to occur, Splunk doubles the wait time until it reaches the value specified in max_backoff.
- **max_backoff**: Tells Splunk the maximum amount of time, in seconds, that it should wait between connection attempts, before invoking max_retries_at_max_backoff. The default is 20.
- **max_retries_at_max_backoff**: If the wait time between connection attempts reaches max_backoff, tells Splunk to try to reconnect to the provider this many times, every max_backoff seconds. The default is 2. If Splunk continues to encounter errors after it has made these attempts, it will give up, and won't attempt to connect to the problem provider again until it is restarted. However, it will continue to log errors such as the example shown above.
- **checkpoint_sync_interval**: Tells Splunk how long, in seconds, to wait for state data (event log checkpoint) to be written to disk. The default is 2.

**Input-specific settings**

Input-specific stanzas tell Splunk how to connect to WMI providers on remote machines to get data. They are defined by one of two attributes that specify the type of data Splunk 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 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 to expect event log data from the sources defined in the stanza.
- **Windows Query Language (WQL)**. The `wql` attribute tells Splunk to expect data from a WMI provider. When using this attribute, you must also specify a valid WQL statement. This attribute must be used when collecting performance data.

**Caution**: 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 parameters for both types of inputs are:

- **server**: A comma-separated list of servers from which to get data. If this parameter is missing, Splunk assumes that you want to connect to the local machine.
- **interval** (required): Tells Splunk 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. There is no default.
- **disabled**: Tells Splunk whether this input is enabled or disabled. Set this parameter to 1 to disable the input, and 0 to enable it. The default is 0 (enabled).
The event log-specific parameters are:

- **event_log_file**: A comma-separated list of event log channels to poll.
- **current_only**: Tells Splunk whether or not to collect events that occur only when it is running. If events are generated when Splunk is stopped, Splunk will not attempt to index those events when it is started again. Set to 1 to have Splunk collect events that occur only when it is running, and 0 to have Splunk collect all events. The default is 0 (gather all available events.)

The WQL-specific parameters are:

- **wql**: A valid WQL statement.
- **namespace** (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 don't specify a path to a remote machine, Splunk will connect 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 (http://msdn.microsoft.com/en-us/library/aa394084(VS.85).aspx).
- **current_only**: Tells Splunk whether or not an event notification query is expected. See "WQL query types: event notification versus standard" below for additional information. Set this attribute to 1 to tell Splunk to expect an event notification query, and 0 to expect a standard query. The default is 0 (standard query.)

**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 the attribute is set to 1, event notification data is expected. 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, if you want to find out when processes are spawned on a remote machine, you must use an event notification query to get that information. 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, pre-existing 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. Check the wmi.conf configuration file reference for specific details and examples.

**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
```
server = foo, bar
interval = 10
event_log_file = Application, System, Directory Service
disabled = 0

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

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

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

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

Fields for WMI data

When Splunk indexes data from WMI-based inputs, 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 sets the source type to WinEventLog:<name of log file>. For example, WinEventLog:Application.
- For WQL data, Splunk 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.

Splunk automatically defines the originating host from the data received.

Troubleshooting WMI logging

If you encounter problems receiving events through WMI providers or are not getting the results you
expect, you can enable debugging in Splunk's logging engine in order to track down the problem.

To enable debugging for WMI-based inputs, you must set two parameters:
1. Edit `log.cfg` in `%SPLUNK_HOME\etc`. Add the following parameter:

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

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

```
category.WMI=DEBUG
```

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

3. Restart Splunk:

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

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

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

5. Once Splunk has collected enough debug log data, send a diag to Splunk Support:

```
C:\Program Files\Splunk\bin> splunk diag
```

**Important:** Once you finish troubleshooting, revert back to the default settings:

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

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

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

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

```
category.WMI=ERROR
```

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

For more information on troubleshooting WMI, see "Troubleshooting WMI Issues" in the Community Wiki.

**Monitor Windows Registry data**
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 will not run. Splunk 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. An example of this is when a program remembers the last positions of open program windows. Later, when the program is run again, it will look into the Registry to read those configurations. You can learn when Registry entries are added, updated, and deleted by programs and processes on your system. When a Registry entry is changed, Splunk 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.

Why monitor the Registry?

The Registry is probably the most used, yet least understood component of Windows operation. It gets used almost constantly, with many different programs reading from and writing to it at all times. When something is not functioning as desired, 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.

The Registry's health is also very important. Splunk not only tells you when changes to the Registry are made, but also whether or not those changes were successful. If programs and processes can't write to or read from the Registry, bad things can happen to your Windows system, including a complete failure. Splunk can alert you to problems interacting with the Registry so that you can restore it from a backup and keep your system running.

Performance considerations

When you install Splunk on a Windows machine and 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.

Since the Registry plays a central role in the operation of a Windows machine, enabling both Registry paths will likely result in a lot of data for Splunk to monitor. To achieve the best performance, it is recommended to filter the amount of Registry data that Splunk indexes by using regmon-filters.conf.

Similarly, you can capture a baseline - a snapshot of the current state of your Windows Registry - when you first start Splunk, and again every time a specified amount of time has passed. The snapshot allows you to 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 may take several minutes. You can postpone taking a baseline snapshot until you've edited regmon-filters.conf and narrowed the
scope of the Registry entries to those you specifically want Splunk to monitor.

More information on regmon-filters.conf and how to use it to filter incoming Registry events is available in "Filter incoming Registry events" later on this page.

Enable Registry monitoring in Splunk Web

Splunk comes with two configured Registry monitoring inputs that are disabled by default. To enable the inputs and configure Splunk to monitor the Windows Registry:

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

2. Under **Data**, click **Data inputs**.

3. Click **Registry Monitoring**.

4. Under **Registry hive**, click on one of the two available Registry hives (**Machine keys** or **User keys**).

5. On the next page, tell Splunk whether or not you want to take a baseline snapshot of the entire Registry before monitoring Registry changes. To set a baseline, click **On** under **Baseline index**.

   **Note:** The baseline snapshot is an index of your entire Registry, at the time the snapshot is taken. Scanning the Registry to set a baseline index is a CPU-intensive process and may take some time.

6. Tell Splunk whether or not you want it to continue monitoring of the Registry after the baseline is taken by selecting **On** or **Off** under **Ongoing monitoring**.

   **Note:** If you select **Off** for **Ongoing monitoring**, Splunk will not monitor changes to the Registry after the baseline snapshot is taken. If you also select **Off** for the baseline snapshot, no data is indexed at all - it is the same as having the input disabled.

7. Optionally, choose the index you want Splunk to send Registry monitoring events to by selecting the desired index under **Index**.

8. Click **Save**.

Splunk enables the input and returns you to the **Registry monitoring** page.

**Note:** To disable either input after it has been enabled, select **Disable** under the **Status** column on the "Registry monitoring" page.

**Caution:** When the Registry monitor is running, do not stop or kill the splunk-regmon.exe process manually. Doing so can result in system instability. To stop the Registry monitor, stop the splunkd server process from either the Services control panel or the CLI.
View Registry change data

To view Registry change data that Splunk has indexed, go to the Search app and search for events with a source of WinRegistry. An example event, which is generated by Group Policy 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:

- **event_status**: The result of the registry change attempt. This should always be "(0) The operation completed successfully.". If it is not, there may be problems with the Registry that will eventually require a restore from a backup.
- **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 program listed above attempted to invoke.
- **key_path**: The Registry key path that the program listed above attempted to change.
- **data_type**: The type of Registry data that the program making the Registry change tried to get or set.
- **data**: The data that the program making the Registry change tried to read or write.

You can use Splunk's search commands and reporting features to create reports based on the incoming data, or use its alerting features to send alerts if things go wrong.

Filter incoming Registry events

Windows Registries are often extremely dynamic, generating a great many events. This can cause problems with licensing - Splunk Registry monitoring can easily generate hundreds of megabytes of data per day, potentially causing license violations.

Splunk provides a two-tiered configuration for fine-tuning the filters that are applied to the Registry event data coming into Splunk.

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

The two configuration files work in concert:
sysmon.conf contains global settings for which event types (adds, deletes, renames, and so on) to monitor, which regular expression filters from the filter rules file to use, and whether or not Windows Registry events are monitored at all.

The filter rules file (by default named regmon-filters.conf) contains the specific regular expressions you create to refine and filter the Registry hive paths you want Splunk to monitor.

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

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

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

- **proc**: A regular expression containing the path to the process or processes you want to monitor
- **hive**: A regular expression containing the hive path to the entry or entries you want to monitor. Splunk supports the root key value mappings predefined in Windows:
  - `\REGISTRY\USER\.*` maps to HKEY_USERS or HKU
  - `\REGISTRY\USER\_Classes` maps to HKEY_CLASSES_ROOT or HKCR
  - `\REGISTRY\MACHINE` maps to HKEY_LOCAL_MACHINE or HKLM
  - `\REGISTRY\MACHINE\SOFTWARE\Classes` maps to HKEY_CLASSES_ROOT or HKCR
  - `\REGISTRY\MACHINE\SYSTEM\CurrentControlSet\Hardware Profiles\Current` maps to HKEY_CURRENT_CONFIG or HKCC
  - **Note**: There is no direct mapping for HKEY_CURRENT_USER or HKCU, as Splunk's Registry monitor runs in kernel mode. However, using `\REGISTRY\USER\.*` (note the period and asterisk at the end) will generate events that contain the logged-in user's security identifier (SID).
  - Alternatively, you can specify the user whose Registry keys you wish to monitor by using `\REGISTRY\USER\<SID>`, where SID is the SID of the desired user.
- **type**: The subset of event types to monitor. Can be one or more of delete, set, create, rename, open, close, query. The values here must be a subset of the values for event_types that you set in sysmon.conf.
- **baseline**: Whether or not to capture a baseline snapshot for that particular hive path. Set to 1 for yes, and 0 for no.
- **baseline interval**: How long Splunk has to have been down before re-taking the snapshot, in seconds. The default value is 86,400 seconds (1 day).
- **disabled**: Whether or not a filter is enabled. Set to 1 to disable the filter, and 0 to enable it.
Get a baseline snapshot

When you enable Registry monitoring, you're given the option of recording a baseline snapshot of the Registry hives the next time Splunk 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 Splunk has been down for more than 24 hours since the last checkpoint, it will retake the baseline snapshot. You can customize this value for each of the filters in `regmon-filters.conf` by setting the value of `baseline_interval` This attribute is expressed in seconds.

Change the default Windows Registry input values

Review inputs.conf to see the default values for Windows Registry input. They are also shown below. You would only need to make changes to the default values if, for example, you wanted to increase or decrease the interval between when the Registry monitor scans for new changes, or change the source and/or sourcetype of events generated by the monitor.

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

To make changes to the default values, edit a copy of `inputs.conf` in `$SPLUNK_HOME/etc/system/local/`. Provide new values for only the parameters you want to change within the `[script://$SPLUNK_HOME/bin/scripts/splunk-regmon.path]` stanza. There's no need to edit the other values. For more information about how to work with Splunk configuration files, refer to "About configuration files" in the Admin Manual.

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

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

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

Real-time Windows performance monitoring

Real-time Windows performance monitoring

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 many bouts of downtime.

Splunk supports the monitoring of all Windows performance counters available to the system in real time, and includes support for both local and remote collection of performance data.
Splunk’s performance monitoring utility gives you the abilities of Performance Monitor in a web or command-line interface. Splunk uses the Performance Data Helper (PDH) API for performance counter queries on local machines, just like Performance Monitor.

The types of performance objects, counters and instances that are available to Splunk depend on the performance libraries installed on the system. Both Microsoft and third-party vendors provide libraries that contain performance counters. For additional information on performance monitoring, review "Performance Counters" (http://msdn.microsoft.com/en-us/library/aa373083%28v=VS.85%29.aspx) on MSDN.

Both full instances of Splunk and universal forwarders support local collection of performance metrics. Remote performance monitoring is available through WMI (Windows Management Instrumentation) and requires that Splunk runs as a user with appropriate Active Directory credentials.

The performance monitor input runs as a process called splunk-perfmon.exe. This process will run once for every input defined, at the interval specified in the input. You can configure performance monitoring using Splunk Web, or either perfmon.conf (for getting local performance data) or wmi.conf (for getting performance data from a remote machine).

Security and remote access considerations

Splunk gets data from remote machines using either WMI or a forwarder. Splunk recommends using a universal forwarder to send performance data from remote machines to an indexer. Review "Introducing the universal forwarder" in the Distributed Deployment Manual for information about how to install, configure and use the forwarder to collect performance metrics.

If you choose to 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 machines.

If you want Splunk to use WMI to get performance data from remote machines, then you must ensure that your network and Splunk instances are properly configured. You cannot install Splunk as the Local System user, and the user you install with determines the set of performance metrics Splunk will see. Review "Security and remote access considerations" in the "Monitor WMI Data" topic in this manual for additional information on the requirements you must satisfy in order for Splunk to collect remote data properly using WMI.

After you install Splunk with a valid user, be sure to add the user to the following groups before enabling local performance monitor inputs:

- **Performance Monitor Users** (domain group)
- **Performance Log Users** (domain group)

Enable local Windows performance monitoring

You can configure local performance monitoring either in Splunk Web, or by using configuration files.

Splunk Web is the preferred way to add performance monitoring data inputs. This is because you can make typos when using configuration files, and it's important to specify performance monitor objects
exactly as they are defined in the Performance Monitor API. See "Important information about specifying performance monitor objects in perfmon.conf" below for a full explanation.

Configure local Windows performance monitoring with Splunk Web

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

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

3. Click **Local performance monitoring**.

4. Click **New** to add an input.

5. Enter a unique, memorable name for this input.

6. Under **Available objects**, choose the performance object whose counters you wish to display.

Splunk loads the available performance counters for the selected object.

**Note:** 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.

7. Under **Counters**, choose the counters in the **Available counters** list box that you want Splunk to monitor by clicking once on them.

The selected counter moves from the **Available counters** list box to the **Selected counters** list box.

8. Under **Instances**, select the instances you want Splunk to monitor by clicking on those instances in the **Available instances** list.

The selected instance moves from the **Available instances** list box to the **Selected instances** list box.

**Note:** The "_Total" instance is a special instance, and is present for many types of performance counters. This instance is defined as 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 monitoring performance data for the "Disk Bytes/Sec" performance counter under the "PhysicalDisk" object on a system with two disks installed, the available instances displayed include one for each physical disk - "0 C:" and "1 D:" - as well as the "_Total" instance. In this case, the "_Total" instance is the average of the two physical disk instances.

9. Specify an interval, in seconds, between polls.

10. Choose the destination index for this collection.
11. Click Save.

Splunk adds and enables the input.

Configure local Windows performance monitoring with configuration files

Performance monitoring configurations are controlled by perfmon.conf. To set up performance monitoring using configuration files, create and/or edit perfmon.conf in %SPLUNK_HOME%\etc\system\local. If you haven't worked with Splunk's configuration files before, be sure to read "About configuration files" before you begin.

perfmon.conf contains one stanza, where you specify:

- **interval** (required): How often, in seconds, to poll for new data. If this attribute is not present and defined, the input will not run, as there is no default.
- **object** (required): The performance object that you wish to capture. If this attribute is not present and defined, the input will not run, as there is no default.
- **counters** (required): One or more valid performance counters that are associated with the object specified in `object`. Multiple counters are separated by 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, as there is no default.
- **instances** (at least one instance is required): One or more valid instances associated with the performance counter specified in `counters`. Multiple instances are separated by semicolons. You can specify all instances by using an asterisk (*).
- **index** (optional): The desired index to route performance counter data to. If not present, the default index is used.
- **disabled**: 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).

The following example of perfmon.conf collects performance data from the local disk on the system and places it into the 'perfmon' index:

```
# Query the PhysicalDisk performance object and gather disk access data for
# all physical drives installed in the system. Store this data in the
# "perfmon" index.
# Note: If the interval attribute is set to 0, Splunk will reset the interval
# to 1.

[Perfmon:LocalPhysicalDisk]
interval = 0
object = PhysicalDisk
counters = Disk Bytes/sec; % Disk Read Time; % Disk Write Time; % Disk Time
instances = *
disabled = 0
index = PerfMon
```

Important information about specifying performance monitor objects in perfmon.conf

When specifying values for the `object`, `counters` and `instances` attributes in perfmon.conf stanzas, be sure that those values exactly match those defined in the Performance Monitor API, including case, or the input will return incorrect data, or no data at all. If Splunk is unable to match a performance object, counter or instance value that you've specified in perfmon.conf, it will log 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.

The best way to ensure that you specify the correct objects, counters, and instances is to use Splunk Web to add performance monitor data inputs.

Enable remote Windows performance monitoring over WMI

You can configure remote performance monitoring either in Splunk Web or by using configuration files.

Caution: When collecting performance metrics over WMI, you must configure Splunk 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 running Splunk and the machine(s) Splunk collects performance data from must reside in the same AD domain or forest.

Note: WMI self-throttles by design to prevent denial of service attacks. Splunk will also throttle WMI calls it makes as an additional 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 system from which you want to collect performance metrics may be a better choice. Consult "Considerations for deciding how to monitor remote Windows data" in this manual for additional information.

Configure remote Windows performance monitoring with Splunk Web

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

2. Under Data, click Data Inputs.

3. Click Remote Performance monitoring.

4. Click New to add an input.

5. Enter a unique name for this collection.

6. Under Select target host, enter the name of a valid Windows host to query performance monitor objects from, then click "Query..."

Splunk connects to the host and gets the available performance objects.

7. In the "Available objects" drop-down, select the performance object whose counters you wish to display.

Splunk loads the available performance counters for the selected object.
Note: 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.

8. Under Counters, choose the counters in the "Available counters" list box that you want Splunk to monitor by clicking once on them.

The selected counter moves from the "Available counters" list box to the "Selected counters" list box.

9. Next, under Instances, select the instances you want Splunk to monitor by clicking on those instances in the Available instances list.

The selected instance moves from the "Available instances" list box to the "Selected instances" list box.

Note: The "Total" instance is a special instance, and is present for many types of performance counters. This instance is defined as the average of any associated instances under the same counter. Data collected for this instance can be - and oftentimes is - significantly different than for individual instances under the same counter.

For example, when monitoring performance data for the "Disk Bytes/Sec" performance counter under the "PhysicalDisk" object on a system with two disks installed, the available instances displayed include one for each physical disk - "0 C:" and "1 D:" - as well as the "Total" instance. In this case, the "Total" instance is the average of the two physical disk instances.

10. You can optionally tell Splunk to collect the same set of metrics from additional hosts by specifying those hosts, separated by commas, in the field provided.

11. Specify an interval, in seconds, between polls.

12. Optionally, choose the destination index for this collection.

By default, the "default" index is selected.

13. Click Save.

The input is added and enabled.

Configure remote Windows performance monitoring with configuration files

Remote performance monitoring configurations are controlled by wmi.conf. 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 Splunk's configuration files before, be sure to read "About configuration files" before you begin.

Caution: Splunk strongly recommends that you use Splunk Web to create remote performance monitor inputs. This is because the names of performance monitor objects, counters, and instances
must exactly match what is defined in the Performance Monitor API, including case. Splunk Web uses WMI to get the properly-formatted names, eliminating this problem.

wmi.conf contains one stanza for each remote performance monitor object that you wish to monitor. In each stanza, you specify:

Global settings

- **initial_backoff**: How long, in seconds, to wait before retrying a connection to a WMI provider when an error occurs. The default is 5. If Splunk continues to have problems connecting to the provider, then it will double the wait time between connection attempts until either it can connect, or until the wait time is greater than or equal to the integer specified in max_backoff.
- **max_backoff**: The maximum amount of time, in seconds to attempt to reconnect to a WMI provider. The default is 20 seconds.
- **max_retries_at_max_backoff**: How many times, after Splunk has reached max_backoff seconds, to attempt to reconnect to a WMI provider. The default is 2.
- **checkpoint_sync_interval**: How long, in seconds, to wait for state data to be flushed to disk. The default is 2.

Input-specific settings

- **interval**: 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.
- **server**: One or more valid servers against which you wish to monitor performance. Multiple entries are separated by commas. If this attribute is not specified, Splunk assumes you want to monitor the local machine.
- **event_log_file**: The names of one or more Windows event log channels to poll. This attribute tells Splunk that the incoming data is in event log format.

Note: Do not use the event_log_file attribute in a stanza that already contains the wql attribute.

- **wql**: A valid Windows Query Language (WQL) statement that specifies the performance object(s), counter(s), and instance(s) you wish to poll remotely. This attribute tells Splunk to expect data from a WMI provider.

Note: Do not use the wql attribute in a stanza that already contains the event_log_file attribute.

- **namespace**: The namespace in which the WMI provider you want to query resides. The value for this attribute can be either relative (Root\CIMV2) or absolute (\SERVER\Root\CIMV2), but must be relative if you specify the server attribute. Defaults to Root\CIMV2.

Note: Only use the namespace attribute in a stanza that contains the wql attribute.

- **index**: The desired index to route performance counter data to. If not present, the 'default' index is used.
- **current_only**: The characteristics and interaction of WMI-based event collections.
  - if wql is defined, this attribute tells Splunk whether or not an event notification query is expected. 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.

- `disabled`: 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. If not present, it defaults to 0.

The following example of `wmi.conf` gathers local disk and memory performance metrics and places them into the 'wmi_perfmon' index:

```
[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.

[WMI:LocalPhysicalDisk]
interval = 1
wql = select Name, DiskBytesPerSec, PercentDiskReadTime, PercentDiskWriteTime, PercentDiskTime
disabled = 0
index = wmi_perfmon

[WMI:LocalMainMemory]
interval = 10
wql = select CommittedBytes, AvailableBytes, PercentCommittedBytesInUse, Caption from Win32_PerfFormattedData_PerfOS_Memory
disabled = 0
index = wmi_perfmon
```

Additional information on WQL query statements

When building WQL queries, make sure that the queries are structurally and syntactically correct. If you don’t, 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 it is used to create performance monitor inputs.

Increased memory usage during collection of performance metrics

When collecting 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.
Other ways to get stuff in

Get data from FIFO queues

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

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

**Add a FIFO input to inputs.conf**

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

Here's the basic syntax for adding a FIFO stanza:

```mermaid
[fifo://<path>]
<attribute1> = <val1>
<attribute2> = <val2>
...
```

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

You can use the following attributes with FIFO stanzas:

**host = <string>**

- Sets the host key/field to a static value for this stanza.
- Sets the host key's initial value. The key is used during parsing/indexing, in particular to set the host field. It is also the host field used at search time.
- The `<string>` is prepended with 'host::'.
- If not set explicitly, this defaults to the IP address or fully qualified domain name of the host where the data originated.

**index = <string>**

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

**sourcetype = <string>**

- Sets the sourcetype key/field for events from this input.
• Primarily used to explicitly declare the source type for this data, as opposed to allowing it to be determined via automated methods. This is typically important both for searchability and for applying the relevant configuration for this type of data during parsing and indexing.

• Sets the sourcetype key's initial value. The key is used during parsing/indexing, in particular to set the source type field during indexing. It is also the source type field used at search time.

  • The `<string>` is prepended with 'sourcetype::'.

  • If not set explicitly, Splunk picks a source type based on various aspects of the data. There is no hard-coded default.

  • For more information about the sourcetype field, see "About default fields (host, source, sourcetype, and more)", in this manual.

source = `<string>`

  • Sets the source key/field for events from this input.

  • **Note:** Overriding the source key is generally not recommended. Typically, the input layer will provide a more accurate string to aid in problem analysis and investigation, accurately recording the file from which the data was retrieved. Please consider use of source types, tagging, and search wildcards before overriding this value.

  • The `<string>` is prepended with 'source::'.

  • Defaults to the input file path.

queue = `[parsingQueue|indexQueue]`

  • Specifies where the input processor should deposit the events that it reads.

  • Set to "parsingQueue" to apply props.conf and other parsing rules to your data.

  • Set to "indexQueue" to send your data directly into the index.

  • Defaults to `parsingQueue`.

**Monitor changes to your filesystem**

Monitor changes to your filesystem

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

Configure the file system change monitor in `inputs.conf`.

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

**How the file system change monitor works**

The file system change monitor detects changes using:
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

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

Configure the file system change monitor

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

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

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".

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

Syntax

Here is the syntax for the [fschange] stanza:

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

- The system will monitor all adds/updates/deletes to the directory and its subdirectories.
- Any change will generate an event that is indexed by Splunk.
- `<directory or file to monitor>` defaults to `$SPLUNK_HOME/etc/`.

Attributes

All attributes are optional. Here is the list of available attributes:

`index=<indexname>`

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

`recurse=<true | false>`

- If true, recurse all directories within the directory specified in `<code>fschange</code>`.
- Defaults to true.

`followLinks=<true | false>`

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

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

`pollPeriod=N`

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

`hashMaxSize=N`

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

`signedaudit=<true | false>`

- Send cryptographically signed add/update/delete events.
- Defaults to false.
- Setting to true will generate events in the `_audit` index.
- This should be set to false if you're setting the `index` attribute.
**Note:** When setting `signedaudit` to true, make sure auditing is enabled in `audit.conf`.

`fullEvent=<true | false>`

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

`sendEventMaxSize=N`

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

`sourcetype = <string>`

- Set the source type for events from this input.
- "sourcetype:" is automatically prepended to `<string>`.
- `sourcetype` is `fs_notification` by default.

`filesPerDelay = <integer>`

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

`delayInMills = <integer>`

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

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

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

**Define a filter**

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

For example:

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

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

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

Example

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

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

[filter:white: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

Get data from APIs and other remote data interfaces through scripted inputs
Get data from APIs and other remote data interfaces through scripted inputs

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

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

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

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

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

Starting with release 4.2, any stderr messages generated by scripted inputs are logged to `splunkd.log`.

Add a scripted input in Splunk Web

To add a scripted input in Splunk Web:

**A. Go to the Add New page**

You add a scripted input from the Add New page in Splunk Web. You can get there through two routes:

- Splunk Manager
- Splunk Home

It doesn’t matter which route you use to get there; the Add New page itself is the same either way.

**Via Splunk Manager:**

1. Click **Manager** in the upper right-hand corner of Splunk Web.
2. In the Data section of the Manager page, click **Data inputs**.
3. Click **Scripts**.
4. Click the **New** button to add an input.

**Via Splunk Home:**

1. Click the **Add Data** link in Splunk Home. This brings you to a page called "Data recipes".
2. Click the **Run and collect the output of a script** link to add an input.

**B. Specify the scripted input**

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

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

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

**Important:** Consult Splunk support before changing this value.

4. To access other settings, check **More settings**. A number of additional settings appear. You can usually go with the defaults for these settings. If you want to set them explicitly, here’s what they’re for:

**a.** You can change the **Host** value, if necessary.

**b.** You can set the **Source type**. **Source type** is a default field added to events. Source type is used to determine processing characteristics, such as timestamps and event boundaries. For information on overriding Splunk’s automatic source typing, see "Override automatic source type assignment" in this manual.

**c.** You can set the **Index** 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 has a number of utility indexes, which also appear in this dropdown box.

5. Click **Save**.

**Add a scripted input via inputs.conf**

You add a scripted input in **inputs.conf** by adding a **[script]** stanza.

**Syntax**

Here is the syntax for the **[script]** stanza:

```
[script://$SCRIPT]
<attribute1> = <val1>
<attribute2> = <val2>
... 
```

**Note the following:**

- **$SCRIPT** is the fully-qualified path to the location of the script.
- As a best practice, put your script in the **bin/** directory nearest the **inputs.conf** where your script is specified. For example, if you are configuring **$SPLUNK_HOME/etc/system/local/inputs.conf**, place your script in **$SPLUNK_HOME/etc/system/bin/**. If you’re working on an application in **$SPLUNK_HOME/etc/apps/$APPLICATION/**, put your script in **$SPLUNK_HOME/etc/apps/$APPLICATION/bin/**.
Attributes

All attributes are optional. Here is the list of available attributes:

`interval = <integer>|<cron schedule>`

- Indicates how often to execute the specified command. Specify either an integer value representing seconds or a valid cron schedule.
- Defaults to 60 seconds.
- When a cron schedule is specified, the script is not executed on start up.
- Splunk keeps one invocation of a script per instance. Intervals are based on when the script completes. So if you have a script configured to run every 10 minutes and the script takes 20 minutes to complete, the next run will occur 30 minutes after the first run.
- For constant data streams, enter 1 (or a value smaller than the script's interval).
- For one-shot data streams, enter -1. Setting `interval` to -1 will cause the script to run each time the splunk daemon restarts.

`index = <string>`

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

`sourcetype = <string>`

- Sets the sourcetype key/field for events from this input.
- Primarily used to explicitly declare the source type for this data, as opposed to allowing it to be determined via automated methods. This is typically important both for searchability and for applying the relevant configuration for this type of data during parsing and indexing.
- Sets the sourcetype key's initial value. The key is used during parsing/indexing, in particular to set the source type field during indexing. It is also the source type field used at search time.
- The `<string>` is prepended with 'sourcetype::'.
- If not set explicitly, Splunk picks a source type based on various aspects of the data. There is no hard-coded default.
- For more information about the sourcetype field, see "About default fields (host, source, sourcetype, and more)", in this manual.

`source = <string>`

- Sets the source key/field for events from this input.
- **Note:** Overriding the source key is generally not recommended. Typically, the input layer will provide a more accurate string to aid 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.
- The `<string>` is prepended with 'source::'.
- Defaults to the input file path.

`disabled = <true | false>`
**disabled** is a boolean value that can be set to true if you want to disable the input. ♦ Defaults to false.

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

**Example using inputs.conf**

This example shows the use of the UNIX **top** command as a data input source:

1. Create a new application directory. This example uses **scripts/**:

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

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 paramaters
```

4. Make sure the script is 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/default/**:

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

**Note:** You may need to modify **props.conf**:

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

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

```
[top]
BREAK_ONLY_BEFORE = <stuff>
```
Since there is no timestamp in the top output we need to tell Splunk to use the current time. This is done in props.conf by setting:

```
DATETIME_CONFIG = CURRENT
```

# Find more things to monitor with crawl

Use the crawl search command to search your filesystem or network for new data sources to add to your index.

You can change default crawler settings by editing crawl.conf. You can override the crawler defaults at the time that you run the `crawl` search command.

The `crawl` command produces a log of crawl activity that's stored in

```
$SPLUNK_HOME/var/log/splunk/crawl.log
```

**Change crawler defaults**

Edit `$SPLUNK_HOME/etc/system/local/crawl.conf` to change the default crawler configuration settings. You define the files and network crawlers separately, in their own stanzas.

**Syntax**

`crawl.conf` contains two stanzas: `[files]` and `[network]`, which define defaults for the files and network crawlers, respectively.

For information on the definable attributes for those stanzas and their default values, read the `crawl.conf` spec file.

**Example**

Here's an example `crawl.conf` file with settings defined for both the files and network crawlers:

```
[files]
bad_directories_list= bin, sbin, boot, mnt, proc, tmp, temp, home, mail, .thumbnails, cache, old
bad_extensions_list= mp3, mpg, jpeg, jpg, m4, mcp, mid
bad_file_matches_list= *example*, *makefile, core.*
packed_extensions_list= gz, tgz, tar, zip
collapse_threshold= 10
days_sizek_pairs_list= 3-0,7-1000, 30-10000
big_dir_filecount= 100
index=main
max_badfiles_per_dir=100

(network]
host = myserver
subnet = 24
```
Recipes

Forwarders

**Important:** You set up inputs on a forwarder the same way you set them up on a Splunk indexer. The only difference is that the forwarder does not include Splunk Web, so you must configure inputs with either the CLI or `inputs.conf`. Before setting up the inputs, you need to deploy and configure the forwarder, as this recipe describes.

You can use Splunk forwarders to send data to indexers, called receivers. This is usually the preferred way to get remote data into an indexer.

To use forwarders, specifically universal forwarders, for getting remote data, you need to set up a forwarder-receiver topology, as well as configure the data inputs:

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

2. Use Splunk Web or the CLI to enable receiving on the instances designated as receivers. See "Enable a receiver" in the Distributed Deployment Manual.

3. Install, configure, and deploy the forwarders. Depending on your forwarding needs, there are a number of best practices deployment scenarios. See "Universal forwarder deployment overview" in the Distributed Deployment Manual for details. Some of these scenarios allow you to configure the forwarder during the installation process.

4. Specify data inputs for each universal forwarder, if you have not already done so during installation. You do this the same way you would for any Splunk instance. As a starting point, see "What Splunk can index" in this manual for guidance on configuring the different types of data inputs.

**Note:** Since the universal forwarder does not include Splunk Web, you must configure inputs through either the CLI or `inputs.conf`; you cannot configure with Splunk Manager.

5. Specify the forwarders’ output configurations, if you have not already done so during installation. You do this through the CLI or by editing the `outputs.conf` file. You get the greatest flexibility by editing `outputs.conf`. For details, see the Distributed Deployment Manual, including "Configure forwarders with outputs.conf".

6. Test the results to confirm that forwarding, along with any configured behaviors like load balancing or filtering, is occurring as expected. Go to the receiver to search on the resulting data.

For more information on forwarders, see the Distributed Deployment Manual, starting with "About forwarding and receiving". Also see "Use forwarders" in this manual.
Files and directories - local

One of Splunk's most versatile capabilities is monitoring files and directories for events. If your system generates it, Splunk can index, search, report and alert on it.

To get data from files and directories into Splunk, point Splunk at a file or directory:

1. From the Home page in Splunk Web, click Add data.

2. Under the To get started... banner, click A file or directory of files.

3. Click Next under Consume any file on this Splunk server.

4. On the Get data from files and directories page, specify the source of the data by clicking on one of the three available choices.

5. In the Full path to your data field, enter the path to the file or directory you want Splunk to monitor:

   You can usually leave the other fields blank, including the fields under the More settings option. Look here for detailed information on those fields.

6. Click Save.

7. From the Success page, click Search to start searching. You can enter any term that's in your data, or you can click on a source, source type or host to see data from the different directories within your syslog directory, the different types of data in those directories, or the different hosts that sent the syslog data in the first place.

For more information on getting data from files and directories, see "Monitor files and directories" in this manual.

Files and directories - remote

The easiest way to get your logs from remote machines into Splunk is with the universal forwarder. You set up the forwarder on the machine generating the logs and then point the forwarder at the Splunk indexer. The forwarder monitors the logs and forwards the events to the indexer, which then indexes them and makes them available for searching.

There are two main steps:

1. Set up the forwarder on the remote machine and point it at the indexer. See this recipe: "Forwarders".
2. Set up the forwarder’s inputs so that they monitor the logs. You set up the inputs on the forwarder the same as if they were on a Splunk indexer. However, the forwarder has no Splunk Web, so you must set up the inputs either with the CLI or by editing `inputs.conf` directly.

For information on setting up inputs to monitor Unix logs, see "Monitor files and directories" in this manual. For additional information on how to set up forwarders, see "Use forwarders" in this manual.

**Syslog - local**

To get local syslog data, point Splunk at a directory of syslog files:

1. From the Home page in Splunk Web, click **Add data**.

2. Under the To get started... banner, click **Syslog**.

3. Click **Next** under **Consume any syslog files or directories on this Splunk server**.

4. On the **Get data from files and directories** page, specify the source of the data by clicking on one of the three available choices.

5. In the **Full path to your data** field, enter the path to the syslog directory:
   - On *nix systems, this is usually `/var/log`.
   - On Windows, the path varies, depending on which third-party syslog daemon you have installed. By default, Windows doesn't provide a syslog facility. Instead, it relies on the Event Log service for logging.

   You can usually leave the other fields blank, including the fields under the **More settings** option. Look here for detailed information on those fields.

6. Click **Save**.

7. From the **Success** page, click **Search** to start searching. You can enter any term that?s in your data, or you can click on a source, source type or host to see data from the different directories within your syslog directory, the different types of data in those directories, or the different hosts that sent the syslog data in the first place.

   For more information on getting data from files and directories, see "Monitor files and directories" in this manual.

**Syslog - TCP**

Splunk can listen on a TCP port for data coming from the syslog service on one or more hosts. You can use Splunk to gather syslog data from these hosts for easy searching, reporting and alerting.
To get syslog data over TCP, configure Splunk to listen on a network port for incoming syslog data:

1. Go to the **Syslog** page in Splunk Web.

2. Then, choose "Next" under **Syslog data from TCP**.

3. On the next page, in the **TCP port** field, enter the TCP port on which you will accept connections from other systems running syslog.

4. You then decide whether or not to **Accept connections from all hosts**. Do so by checking either the **Yes** or **No, restrict to one host** radio buttons.

   If you select **No, restrict to one host**, another field named **Host Restriction** appears. Enter the name of one valid host on your network - Splunk will only accept connections from that computer.

5. Optionally, you can tell Splunk to override the default source value for your script, by putting a string into the **Source name override** field.

6. You can also set the sourcetype of the events generated by this source by choosing **From list** in the **Set sourcetype** drop-down, then selecting the desired choice from the **Select source type from list** drop-down.

   You will typically want to set the source type to 'syslog'.

7. Alternatively, you can choose **Manually** from "Set sourcetype," and then enter a string in the **Source type** field that appears.

   You can usually leave the other fields unchanged, including the fields under the **More settings** option.

8. Finally, click **Save**.

9. From the **Success** page, click **Search** to start searching. You can enter any term that?s in your data, or you can click on a source, source type or host to see data from the different directories within your syslog directory, the different types of data in those directories, or the different hosts that sent the syslog data in the first place.

   For more information on getting data from the network, see "Get data from TCP and UDP ports" in this manual.

**Syslog - UDP**

Splunk can listen on a UDP port for data coming from the syslog service on one or more hosts. You can use Splunk to gather syslog data from these hosts for easy searching, reporting and alerting.

To get syslog data over UDP, configure Splunk to listen for that data over a UDP port:
1. Go to the **Syslog** page in Splunk Web.

2. Then, choose "Next" under **Syslog data from UDP**.

3. On the next page, in the **UDP port** field, enter the UDP port on which you will accept connections from other systems running syslog.

   The default syslog UDP port is 514.

4. Optionally, you can tell Splunk to override the default source value for your script, by putting a string into the **Source name override** field.

5. You can set the sourcetype of the events generated by this source by choosing **From list** in the **Set sourcetype** drop-down, then selecting the desired choice from the **Select source type from list** drop-down.

   You will typically want to set the source type to 'syslog'.

6. Alternatively, you can choose **Manually** from "Set sourcetype," and then enter a string in the **Source type** field that appears.

   You can usually leave the other fields unchanged, including the fields under the **More settings** option. Look here for detailed information on these fields.

7. Finally, click **Save**.

8. From the **Success** page, click **Search** to start searching. You can enter any term that’s in your data, or you can click on a source, source type or host to see data from the different directories within your syslog directory, the different types of data in those directories, or the different hosts that sent the syslog data in the first place.

   For more information on getting data from the network, see "Get data from TCP and UDP ports" in this manual.

### Windows event logs - local

Splunk allows for fast, easy collection of Windows event logs. Whether it’s for alerting on security, or reporting on or searching for various event IDs to determine the health of your Windows systems, Splunk’s event log collection capabilities make it a snap.

To get local Windows event log data, point Splunk at your Event Log service:

1. From the **Home** page in Splunk Web, click **Add data**.

2. Under the **To get started...** banner, click **Windows event logs**.

3. Click **Next** under **Collect Windows event logs from this Splunk server**.
4. In the "Available Logs" window, click on the event log channels that you want Splunk to monitor. The log channels will appear in the "Selected Logs" window.

5. Optionally, set the destination index for this source by selecting an index from the Index drop-down box.

6. Click Save.

7. From the Success page, click Search to start searching. You can enter any term that's in your data, or you can click on a source, source type or host to see data from the events as they come into Splunk.

For more information on getting data from files and directories, see "Monitor Windows event log data" in the Getting Data In manual.

**Windows event logs - remote**

Splunk can monitor Windows event logs, both locally and remotely over WMI. Whether it's for alerting on security or reporting on or searching of various event IDs to determine the health of your Windows systems, Splunk's event log collection capabilities make it a snap.

To get remote Windows event log data, point Splunk at a remote machine's Event Log service:

1. From the Home page in Splunk Web, click Add data.

2. Under the To get started... banner, click Windows event logs.

3. Click Next under Collect Windows event logs from another machine.

4. In the Event Log collection name field, type in a unique name for the event logs you will be collecting.

5. In the Choose logs from this host field, enter the hostname for a machine on your Windows network. You can specify a short hostname, the server's fully qualified domain name, or its IP address.

6. Click Find logs? to get a list of the available event log channels on the remote machine.

7. In the Available log(s) window that appears, click once on the event log channels you want Splunk to monitor.

The log channels will appear in the Selected Logs window.

8. Optionally, you can specify additional servers to collect the same set of event logs from. Type in each of the hostnames, separating them with commas.
9. Another option is to set the destination index for this source. You can do so by selecting an index from the Index drop-down box.

10. Click Save.

11. From the Success page, click Search to start searching. You can enter any term that’s in your data, or you can click on a source, source type or host to see data from the events as they come into Splunk.

For more information on getting data from Windows event logs, see "Monitor Windows event log data" in this manual.

**Windows event logs - many remote**

There are a number of ways to gather Windows event logs from large numbers of Windows machines. One way is by using Splunk’s "Windows event logs remote" recipe to pull the logs from the machines into your Splunk instance. The other, faster, more scalable way, is to use Splunk’s universal forwarder.

You set up the forwarder on the machines that are generating the desired event logs. Then, you point the forwarder at the Splunk indexer. The forwarder monitors the desired event logs on each machine, then forwards that data to the indexer, which then indexes it and makes it available for searching.

Using the universal forwarder is the most efficient way to get event logs from a large number of remote Windows machines.

There are two main steps:

1. Set up the forwarder on the remote machine and point it at the indexer. See this recipe: "Forwarders".

2. Set up the forwarder's inputs so that they event logs that you desire. You set up the inputs on the forwarder the same as if they were on a Splunk indexer. However, the forwarder has no Splunk Web, so you’ll need to set up the inputs either with the command line interface (CLI), or by editing inputs.conf directly.

For information on setting up inputs to get Windows event logs, see "Monitor Windows event log data" in this manual. For additional information on setting up forwarders, see "Use forwarders" in this manual.

**Windows registry - local**

You can monitor changes to the Registry on Windows machines with Splunk. Whether it’s an entire hive or just one key, whether it’s an add, change, delete or even just a read - Splunk's Registry monitoring service can collect that data and allow you to search, report and alert on it.
To get local Windows registry change data, attach Splunk to your registry:

1. From the Home page in Splunk Web, click Add data.

2. Under the To get started... banner, click Windows registry.

3. Choose Next under Windows Registry data on this Splunk server.

4. On the following page, you can choose whether to query the Registry hives under the Machine keys (for hives such as HKEY_LOCAL_MACHINE or HKEY_CURRENT_CONFIG) or the User keys (for hives such as HKEY_CURRENT_USER). Choose the option that best suits the types of Registry changes you would like to have monitored.

5. On the next page, you can decide whether you want a baseline index of the Registry. You can also decide whether you wish to enable ongoing monitoring. Make your choices by selecting the appropriate radio checkboxes.

6. Optionally, you can set the destination index for this source.

7. Finally, click Save.

8. You'll be returned to the Registry Monitoring page. From here, you can enable monitoring on the other Registry hive, or click Back to Home in the upper left corner of the screen to either search through your data, or add more.

Windows Registry monitoring inputs generate a lot of data. To learn how to filter that data, or for additional information about the Registry monitor inputs, review "Monitor Windows registry data" in this manual.

**Windows registry - remote**

The easiest - and, in fact, the only - way to get your remote Windows Registry changes into Splunk is with the universal forwarder. You configure the forwarder on the machine generating the Registry data, and then point the forwarder at the Splunk indexer. The forwarder monitors the logs and forwards the events to the indexer, which then indexes them and makes them available for searching.

Using the universal forwarder is the only way to get Registry data from remote machines.

There are two main steps:

1. Set up the forwarder on the remote machine and point it at the indexer. See this recipe: "Forwarders".

2. Set up the forwarder's inputs so that they monitor the Registry. You set up the inputs on the forwarder the same as if they were on a Splunk indexer. However, the forwarder has no Splunk Web, so you'll need to set up the inputs either with the command line interface (CLI), or by editing inputs.conf directly.
For information on setting up inputs to monitor the Registry, see "Monitor Windows Registry data" in this manual. For additional information on setting up forwarders, see "Use forwarders" in this manual.

**Windows performance monitoring - local**

Splunk is the simple, web-based alternative to Performance Monitor. Whether you want to watch disk I/O, memory metrics such as free pages or commit charge, or network statistics, Splunk's collection, charting and reporting utilities increase its extensibility.

Here's how to get your performance metrics with Splunk:

1. Go to the **Windows performance data** page in Splunk Web.

2. Under **Collection name**, enter a unique name for this collection that you'll remember.

3. In the **Available objects** drop-down box, select a performance object that you would like for Splunk to monitor.

The **Available counters** window appears, containing counters that are specific to the object you just selected.

4. From the **Available counters** listbox, click once on each counter that you would like for Splunk to collect performance data for.

The desired performance counters will appear in the **Selected counters** window.

5. Next, from the **Available instances** listbox, click once on each of the desired instances for the counters selected above, that you would like for Splunk to track.

The desired instances will appear in the **Selected instances** list box.

You can usually leave the other settings as they are, though if you want to change the polling interval, you can do so by specifying it in the "Polling interval" field. Look here for detailed information on those settings.

6. Click **Save**.

7. From the **Success** page, click **Search** to start searching. You can enter any term that?s in your data, or you can click on a source, source type or host to see data from the events as they come into Splunk.

For more information on getting data from files and directories, see "Real-time Windows performance monitoring" in this manual.
Splunk is the simple, web-based alternative to Performance Monitor. Whether you want to watch disk I/O, memory metrics such as free pages or commit charge, or network statistics, Splunk’s collection, charting and reporting utilities increase its extensibility. And, like Performance Monitor, you can monitor machines remotely.

Here’s how to get your performance metrics with Splunk:

2. From there, locate Windows event logs from another machine and click Next.
3. Under Collection name, enter a unique name for this collection that you’ll remember.
4. In the Select target host field, enter the hostname for a machine on your Windows network.
   You can specify a short hostname, the server’s fully qualified domain name, or its IP address.
5. Click Query? to get a list of the available performance objects on the remote machine.
6. In the Available objects drop-down box, select a performance object that you would like for Splunk to monitor.
   The Available counters window appears, containing counters that are specific to the object you just selected.
7. From the Available counters listbox, click once on each counter that you would like for Splunk to collect performance data.
   The desired performance counters appear in the Selected counters window.
8. Next, from the Available instances listbox, click once on each of the desired instances for the counters selected above, that you would like for Splunk to track.
   The desired instances will appear in the Selected instances list box.
9. Optionally, you can specify additional servers from which to collect the same set of performance metrics. Type in each of the hostnames into the field, separating them with commas.
   You can usually leave the other settings as they are, though if you want to change the polling interval, you can do so by specifying it in the "Polling interval" field. Look here for detailed information on those settings.
10. Click Save.
11. From the Success page, click Search to start searching. You can enter any term that's in your data, or you can click on a source, source type or host to see data from the events as they come into Splunk.

For more information on getting performance monitor data from remote machines, see "Monitor WMI data" in the Getting Data In manual.

**Windows performance - many remote**

There are a number of ways to cull Windows performance metrics from large numbers of Windows machines. One way is by using Splunk’s "Windows performance remote" recipe to pull the data from the machines into your Splunk instance, one machine at a time. The other, more scalable way, is to use Splunk's universal forwarder.

You set up the forwarder on the machines that are generating the performance metrics. Then, you point the forwarder at the Splunk indexer. The forwarder monitors the desired performance counters on the machine, then forwards that data to the indexer, which then indexes it and makes it available for searching.

Using the universal forwarder is the most efficient way to get performance metrics from remote Windows machines.

There are two main steps:

1. Set up the forwarder on the remote machine and point it at the indexer. See this recipe: "Forwarders".

2. Set up the forwarder’s inputs so that they monitor the performance metrics you desire. You set up the inputs on the forwarder the same as if they were on a Splunk indexer. However, the forwarder has no Splunk Web, so you'll need to set up the inputs either with the command line interface (CLI), or by editing `perfmon.conf` directly.

For information on setting up inputs to gather performance metrics, see "Real-time Windows performance monitoring" in this manual. For additional information on setting up forwarders, see "Use forwarders" in this manual.

**Windows Active Directory**

You can collect any kind of Active Directory change data with Splunk.

Do you want or need to know who's been changing passwords, adding user or machine accounts, or delegating authority to Group Policy objects? All of that information is at your fingertips with Splunk's Active Directory monitor. What’s more, you can choose which part of the AD you want to scan for changes - from one node to the entire AD forest.
Note: In order to monitor any part of Active Directory, at a minimum you'll need to run Splunk as a user with read permissions to the Active Directory schema.

To get Active Directory data, introduce Splunk to your Active Directory:

1. From the Home page in Splunk Web, click Add data.

2. Under the Choose how you want Splunk to consume your data banner, click Monitor an Active Directory schema.

3. In the AD monitor name field, enter a unique name that you'll remember.

4. In the Target Domain Controller field, enter the host name of a domain controller on your network. Or, leave this field blank, and Splunk will look for the nearest available domain controller, and bind to it.

5. Optionally, in the Starting Node field, type in the Active Directory node that Splunk should begin monitoring from. Or, leave this field blank, and Splunk will begin monitoring from the highest part of the Active Directory tree that it has access to.

6. Check the Monitor subtree box to have Splunk monitor all child nodes under the node you specified in Step 5 (or, the top of the AD tree if no starting node was specified). Leave the box unchecked if you only wish to monitor the specified starting node.

7. Optionally, you can specify the destination index for this source.

8. Finally, click Save.

9. From the Success page, click Search to start searching. You can enter any term that's in your data, or you can click on a source, source type or host to see data from the Active Directory events as they come into Splunk.

For more information on getting data from files and directories, see "Monitor Windows event log data" in this manual.

Unix logs - local

Unix logs - local

To get data from Unix logs into Splunk, point Splunk at a file, or a directory containing Unix logs:

1. From the Home page in Splunk Web, click Add data.

2. Under the To get started... banner, click Unix/Linux logs and metrics.

3. Click Next under Consume Unix/Linux logs and metrics from this Splunk server.

4. On the Get data from files and directories page, specify the source of the data by clicking on one of the three available choices.
5. In the Source field, enter the path to the file or directory you want Splunk to monitor:

You can usually leave the other fields blank, including the fields under the More settings option. Look here for detailed information on those fields.

6. Click Save.

7. From the Success page, click Search to start searching. You can enter any term that's in your data, or you can click on a source, source type or host to see data from the different directories within your syslog directory, the different types of data in those directories, or the different hosts that sent the syslog data in the first place.

For more information on getting data from files and directories, see "Monitor files and directories" in this manual.

**FSChange - local**

Splunk can be configured to monitor changes to your local file system as they occur. To do this, edit inputs.conf and enable the fschange scripted input. Once configured, Splunk will dutifully monitor any and all changes to the file system you specify.

More information on how to enable and configure the fschange input can be found at "Monitor changes to your filesystem" in this manual.

**WebSphere - local**

To get data from WebSphere logs into Splunk, point Splunk at a file, or a directory containing WebSphere logs:

1. From the Home page in Splunk Web, click Add data.

2. Under the To get started... banner, click WebSphere logs, metrics and other data.

3. Click Next under Consume WebSphere server logs from this Splunk server.

4. On the Get data from files and directories page, specify the source of the data by clicking on one of the three available choices.

5. In the Source field, enter the path to the file or directory you want Splunk to monitor:

You can usually leave the other fields blank, including the fields under the More settings option. Look here for detailed information on those fields.

6. Click Save.
7. From the Success page, click Search to start searching. You can enter any term that?s in your data, or you can click on a source, source type or host to see data from the different directories within your syslog directory, the different types of data in those directories, or the different hosts that sent the syslog data in the first place.

For more information on getting data from files and directories, see "Monitor files and directories" in this manual.

**IIS logs - local**

To get data from Internet Information Server (IIS) web server logs into Splunk, point Splunk at a file, or a directory containing IIS web server logs:

1. From the Home page in Splunk Web, click Add data.

2. Under the To get started... banner, click IIS logs.

3. Click Next under Consume IIS logs and metrics from this Splunk server.

4. On the Get data from files and directories page, specify the source of the data by clicking on one of the three available choices.

5. In the Source field, enter the path to the file or directory you want Splunk to monitor:

You can usually leave the other fields blank, including the fields under the More settings option. Look here for detailed information on those fields.

6. Click Save.

7. From the Success page, click Search to start searching. You can enter any term that?s in your data, or you can click on a source, source type or host to see data from the different directories within your syslog directory, the different types of data in those directories, or the different hosts that sent the syslog data in the first place.

For more information on getting data from files and directories, see "Monitor files and directories" in this manual.

**Apache logs - local**

To get data from Apache web server logs into Splunk, point Splunk at a file, or a directory containing Apache logs:

1. From the Home page in Splunk Web, click Add data.

2. Under the To get started... banner, click Apache logs.
3. Click **Next** under **Consume Apache logs on this Splunk server**.

4. On the **Get data from files and directories** page, specify the source of the data by clicking on one of the three available choices.

5. In the **Source** field, enter the path to the file or directory you want Splunk to monitor:

   You can usually leave the other fields blank, including the fields under the **More settings** option. Look here for detailed information on those fields.

6. Click **Save**.

7. From the **Success** page, click **Search** to start searching. You can enter any term that's in your data, or you can click on a source, source type or host to see data from the different directories within your Apache log directory, the different types of data in those directories, or the different hosts that sent the Apache log data in the first place.

   For more information on getting data from files and directories, see "Monitor files and directories" in this manual.

**JMS/JMX - write your own**

You can write custom Java Message Service (JMS) or Java Management Extension (JMX) scripts and then set up Splunk to run the scripts at regular intervals to gather data about your Java MBeans, VMs or other J2EE-based technical services.

First, write and test your JMS or JMX script. Once that's done, place it in $SPLUNK_HOME/bin/scripts on the Splunk instance you want to monitor your Java environment.

Then, point Splunk at this script:

1. From the **Home** page in Splunk Web, click **Add data**.

2. Under the **To get started...** banner, click **WebSphere logs, metrics and other data**.

3. Click **Next** under **Collect messages from JMS**.

4. On the **Add New** page, specify the name of your script, including any required arguments.

5. Tell Splunk how often you want to run the script by specifying the desired interval in the **Interval** field.

6. Optionally, you can tell Splunk to override the default source value for your script, by putting a string into the **Source name override** field.

7. You can also set the sourcetype of the events generated by this script by choosing **From list** in the **Set sourcetype** drop-down, then selecting the desired choice from the **Select source type from list**
drop-down. Or, choose **Manually** from "Set sourcetype," and then enter a string in the **Source type** field that appears.

You can usually leave the other fields unchanged, including the fields under the **More settings** option. Look here for detailed information on these fields.

8. Click **Save**.

9. From the **Success** page, click **Search** to start searching. You can enter any term that’s in your data, or you can click on a source, source type or host to see data from the different directories within your syslog directory, the different types of data in those directories, or the different hosts that sent the syslog data in the first place.

For more information on getting data from scripts into Splunk, see "Get data from APIs and other remote data interfaces through scripted inputs" in this manual.
Configure event processing

Overview of event processing

Events are records of activity in log files, stored in Splunk indexes. They are primarily what Splunk indexes. Events provide information about the systems that produce the log files. The term event data refers to the contents of a Splunk index.

Here's a sample event:

```
172.26.34.223 -- [01/Jul/2005:12:05:27 -0700] "GET /trade/app?action=logout HTTP/1.1" 200 2953
```

When Splunk indexes events, it:

- Configures character set encoding.
- Configures linebreaking for multi-line events.
- Identifies event timestamps (and applies timestamps to events if they do not exist).
- Extracts a set of useful standard fields such as host, source, and sourcetype.
- Improves data compression with segmentation.
- Dynamically assigns metadata to events, if specified.
- Anonymizes data if specified through sed or through configuration files.

For an overview of the Splunk indexing process, see the Indexing with Splunk chapter of the Admin manual.

Configure character set encoding

Splunk allows you to configure character set encoding for your data sources. Splunk has built-in character set specifications to support internationalization of your Splunk deployment. Splunk supports 71 languages (including 20 that aren't UTF-8 encoded). You can retrieve a list of Splunk's valid character encoding specifications by using the `iconv -l` command on most *nix systems.

Splunk attempts to apply UTF-8 encoding to your sources by default. If a source doesn't use UTF-8 encoding or is a non-ASCII file, Splunk will try to convert data from the source to UTF-8 encoding unless you specify a character set to use by setting the CHARSET key in props.conf.

**Note:** If a source's character set encoding is valid, but some characters from the specification are not valid in the encoding you specify, Splunk escapes the invalid characters as hex values (for example: "\xF3").
Supported character sets

Splunk supports an extremely wide range of character sets, including such key ones as UTF-8, UTF-16LE, Latin-1, BIG5, and SHIFT-JIS. See "Comprehensive list of supported character sets" at the end of this topic for the exhaustive list.

Here's a short list of some of the main character sets that Splunk supports, along with 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>ARMSCLII-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>
<tr>
<td>Russian</td>
<td>KOI8-R</td>
</tr>
<tr>
<td>Slovak</td>
<td>CP1250</td>
</tr>
<tr>
<td>Slovenian</td>
<td>ISO-8859-2</td>
</tr>
<tr>
<td>Thai</td>
<td>TIS-620</td>
</tr>
<tr>
<td>Ukrainian</td>
<td>KOI8-U</td>
</tr>
<tr>
<td>Vietnamese</td>
<td>VISCII</td>
</tr>
</tbody>
</table>

Manually specify a character set

Manually specify a character set to apply to an input by setting the CHARSET key in props.conf:

```
[spec]
CHARSET=<string>
```

For example, if you have a host that is generating data in Greek (called "GreekSource" in this example) and that uses ISO-8859-7 encoding, set CHARSET=ISO-8859-7 for that host in props.conf:

```
[host::GreekSource]
CHARSET=ISO-8859-7
```

Note: Splunk will only parse character encodings that have UTF-8 mappings. Some EUC-JP characters do not have a mapped UTF-8 encoding.
Automatically specify a character set

Splunk can automatically detect languages and proper character sets using its sophisticated character set encoding algorithm.

Configure Splunk to automatically detect the proper language and character set encoding for a particular input by setting CHARSET=AUTO for the input in props.conf. For example, if you want Splunk to automatically detect character set encoding for the host "my-foreign-docs", set CHARSET=AUTO for that host in props.conf:

[host::my-foreign-docs]
CHARSET=AUTO

If Splunk doesn't recognize a character set

If you want to use an encoding that Splunk doesn't recognize, train Splunk to recognize the character set by adding a sample file to the following directory:

$SPLUNK_HOME/etc/ngram-models/_<language>_<encoding>.txt

Once you add the character set specification file, you must restart Splunk. After you restart, Splunk can recognize sources that use the new character set, and will automatically convert them to UTF-8 format at index time.

For example, if you want to use the "vulcan-ISO-12345" character set, copy the specification file to the following path:

/SPLUNK_HOME/etc/ngram-models/_vulcan-ISO-12345.txt

Comprehensive list of supported character sets

The common character sets described earlier are just the tip of Splunk's CHARSET iceberg. Splunk actually supports a long list of character sets and aliases, identical to the list supported by the *nix iconv utility. Here's 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)
• latin-7 (aka, ISO-8859-13, ISO-IR-179, L7)
• ISO-8859-11
• roman-8 (aka, HP-ROMAN8, R8, CSHPROMAN8)
• KOI8-R (aka, CSKOI8R)
• KOI8-U
• KOI8-T
• GEORGIAN-ACADEMY
• GEORGIAN-PS
• ARMSGII-8
• MACINTOSH (aka, MAC, MACROMAN, CSMACINTOSH) [Note: these MAC* charsets are for MacOS 9; OS/X uses unicode]
• MACGREEK
• MACCYRILLIC
• MACUKRAINE
• MACCENTRALEUROPE
• MACTURKISH
• MACCROATIAN
• MACICELAND
• MACROMANIA
• MACHEBREW
• MACTHAI
• NEXTSTEP
• CP850 (aka, 850, IBM850, CSPC850MULTILINGUAL)
• 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-GB-2312)
• GB_2312-80 (aka, CHINESE, ISO-IR-58, CSISO58GB231280)
• SHFT-JIS (aka, MS_KANJI, SJIS, CSSHIFTJIS)
• JISX0201-1976 (aka, JIS_X0201, X0201, CSSHALFWIDTHKATAKANA)
• 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)

Note: Splunk ignores punctuation and case when matching CHARSET, so, for example, "utf-8", "UTF-8", and "utf8" are all considered identical.

Configure linebreaking for multi-line events

Configure linebreaking for multi-line events

Overview of multi-line events and event linebreaking

Some events are made up of more than one line. Splunk handles most of these kinds of events correctly by default, but there are cases of multi-line events that Splunk doesn't recognize properly by default. These require special configuration to change Splunk's default linebreaking behavior.

Multi-line event linebreaking and segmentation limitations

Splunk does apply limitations to extremely large events when it comes to linebreaking and segmentation:

- **Lines over 10,000 bytes**: Splunk breaks lines over 10,000 bytes into multiple lines of 10,000 bytes each when it indexes them. It appends the field `meta::truncated` to the end of each truncated section. However, Splunk still groups these lines into a single event.
- **Segmentation for events over 100,000 bytes**: Splunk only displays the first 100,000 bytes of an event in the search results. Segments after those first 100,000 bytes of a very long line are still searchable, however.
- **Segmentation for events over 1,000 segments**: Splunk displays the first 1,000 individual segments of an event as segments separated by whitespace and highlighted on mouseover. It
Configuration

Many event logs have a strict one-line-per-event format, but some do not. Usually, Splunk can automatically figure out the event boundaries. However, if event boundary recognition is not working as desired, you can set custom rules by configuring props.conf.

To configure multi-line events, examine the format of the events. Determine a pattern in the events to set as the start or end of an event. Then, edit $SPLUNK_HOME/etc/system/local/props.conf, and set the necessary attributes for your data handling.

There are two ways to handle multi-line events:

- Break the event stream into real events. This is recommended, as it increases indexing speed significantly. Use LINE_BREAKER (see below).
- Break the event stream into lines and reassemble. This is slower but affords more robust configuration options. Use any linebreaking attribute besides LINE_BREAKER (see below).

Linebreaking general attributes

These are the props.conf attributes that affect linebreaking:

**TRUNCATE = <non-negative integer>**

- Change the default maximum line length (in bytes). Note that although this attribute is a byte measurement, Splunk rounds down line length when this attribute would otherwise land mid-character for multi-byte characters.
- Set to 0 if you never want truncation (very long lines are, however, often a sign of garbage data).
- Defaults to 10000 bytes.

**LINE_BREAKER = <regular expression>**

- Specifies a regex that determines how the raw text stream is broken into initial events, before line merging takes place (see the SHOULD_LINEMERGE attribute, below).
- The regex must contain a matching group.
- Wherever the regex matches, Splunk considers the start of the first matched group to be the end of the previous event, and considers the first matching group to be the start of the next event.
- The contents of the first matching group are ignored as event text.
- **Note:** You get a significant speed boost when you use LINE_BREAKER to delimit multiline events rather than using line merging to reassemble individual lines into events (see SHOULD_LINEMERGE, below).
- Defaults to ([\r\n]+), which means that data is broken into an event for each line, delimited by \r or \n.

**LINE_BREAKER_LOOKBEHIND = <integer>**
• 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 applies the `LINEBREAKER` regex. You may want to increase this value from its default if you are dealing with especially large or multiline events.
  • Defaults to 100.

`SHOULD_LINEMERGE = <true/false>`

• When set to true, Splunk combines several input lines into a single event, with configuration based on the attributes described below.
  • Defaults to true.

Attributes that are available only when SHOULD_LINEMERGE is set to true

When SHOULD_LINEMERGE is set to true, these additional attributes can be used to further define linebreaking behavior:

`BREAK_ONLY_BEFORE_DATE = <true/false>`

• When set to true, Splunk creates a new event if and only if it encounters a new line with a date.
  • Defaults to true.

`BREAK_ONLY_BEFORE = <regular expression>`

• When set, Splunk will create a new event if and only if it encounters a new line that matches the regular expression.
  • Defaults to empty.

`MUST_BREAK_AFTER = <regular expression>`

• When set, and the regular expression matches the current line, Splunk always creates a new event for the next input line.
  • Splunk may still break before the current line if another rule matches.
  • Defaults to empty.

`MUST_NOT_BREAK_AFTER = <regular expression>`

• When set and the current line matches the regular expression, Splunk does not break on any subsequent lines until the MUST_BREAK_AFTER expression matches.
  • Defaults to empty.

`MUST_NOT_BREAK_BEFORE = <regular expression>`

• When set and the current line matches the regular expression, Splunk does not break the last event before the current line.
  • Defaults to empty.

`MAX_EVENTS = <integer>`
• Specifies the maximum number of input lines that will be added to any event.
• Splunk will break after the specified number of lines are read.
• Defaults to 256.

Examples

Specify event breaks

```
[my_custom_sourcetype]
BREAK_ONLY_BEFORE = ^\d+\s*$
```

This example instructs Splunk to divide events in a file or stream by presuming any line that consists of all digits is the start of a new event, for any source whose source type was configured or determined by Splunk to be `sourcetype::my_custom_sourcetype`.

Merge multiple lines into a single event

The following log event contains several lines that are part of the same request. The differentiator between requests is "Path". For this example, assume that all these lines need to be shown as a single event entry.

```
{{"2006-09-21, 02:57:11.60", 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
MANUser:
Version=1&Name=&Debit=&Credit=&AccessTime=&BillDay=&Status=&Language=&Country=&Email=&EmailNotify=&Pin=&PinPayment=&PinAmount=&PinPG=&PinPGRate=&PinMenu=&", ""}}
```

To index this multiple line event properly, use the `Path` differentiator in your configuration. Add the following to your `$SPLUNK_HOME/etc/system/local/props.conf`:

```
[source::source-to-break]
SHOULD_LINEMERGE = True
BREAK_ONLY_BEFORE = Path=
```

This code tells Splunk to merge the lines of the event, and only break before the term `Path=`.

Answers

Have questions? Visit Splunk Answers and see what questions and answers the Splunk community has around multi-line event processing.

Handle event timestamps
Handle event timestamps

Look carefully at this sample event:

172.26.34.223 - - [01/Jul/2005:12:05:27 -0700] "GET /trade/app?action=logout HTTP/1.1" 200 2953

Notice the time information in the event: [01/Jul/2005:12:05:27 -0700]. This is what is known as a timestamp. Splunk uses timestamps to correlate events by time, create the histogram in Splunk Web, and set time ranges for searches. Most events contain timestamps, and in those cases where an event doesn't contain timestamp information, Splunk attempts to assign a timestamp value to the event at index time.

Most events do not require additional handling of timestamp formatting, but there are situations that require the involvement of a Splunk administrator to help set things right. In the case of some sources and distributed deployments, for example, the Splunk admin may have to reconfigure timestamp recognition and formatting. Other timestamp-handling activities that the admin might undertake include:

- Configuration of timestamp extraction for events with multiple timestamps
- Application of timestamp offsets (to correlate events from different timezones)
- Training Splunk to recognize a timestamp
- Tuning timestamp extraction performance

For more information about timestamps, see the Configure event timestamping chapter of this manual.

Extract default fields automatically

When Splunk indexes event data, it extracts by default a set of fields that are common to most events, and which are commonly used in Splunk searches and reports. These default fields include:

- **host**: Identifies the originating hostname or IP address of the network device that generated the event. Used to narrow searches to events that have their origins in a specific host.
- **source**: Identifies the filename or pathname from which the event was indexed. Used to filter events during a search, or as an argument in a data-processing command.
- **sourcetype**: Identifies the type of application, network, or device data that the event represents, such as access_log or syslog. A Splunk administrator can predefine source types, or they can be generated automatically by Splunk at index time. Use sourcetype to filter events during a search, or as an argument in a data-processing command.

For a full listing of the default fields that Splunk identifies during the indexing process, and examples of how they can be used in a search, see "Use default fields" in the User manual.

For detailed information on default field extraction, see "About default fields", and subsequent topics, in this manual.
**Improve data compression with segmentation**

**Segmentation** is what Splunk uses to break events up into searchable segments at index time, and again at search time. Segments can be classified as **major** or **minor**. To put it simply, minor segments are breaks within major segments. For example, the IP address 172.26.34.223 is, as a whole, a major segment. But this major segment can be broken down into minor segments such as 172 as well as groups of minor segments like 172.26.34.

Splunk enables a Splunk admin to define how detailed the event segmentation should be. This is important because **index-time segmentation** affects indexing and search speed, impacts disk compression, and affects your ability to use typeahead functionality. **Search-time segmentation**, on the other hand, can also affect search speed as well as your ability to create searches by selecting items from the results displayed in Splunk Web.

Index-time segmentation is set through `segmenters.conf`, while search-time segmentation is set in the search results page in Splunk Web, as described here.

For more information about "index time" and "search time," see Index time versus search time, in the Admin manual.

**Levels of event segmentation**

There are three levels of segmentation that the Splunk admin can choose from for index time and search time:

- **Inner segmentation** breaks events down into the smallest minor segments possible. For example, when an IP address such as 172.26.34.223 goes through inner segmentation, it is broken down into 172, 26, 34, and 223. Setting inner segmentation at index time leads to very efficient indexes in terms of search speed, but it also impacts indexing speed and restricts the typeahead functionality (it will only be able to typeahead at the minor segment level).

- **Outer segmentation** is the opposite of inner segmentation. Under outer segmentation only major segments are indexed. In the previous example, the IP address would not be broken down into any components. If you have outer segmentation set at index time you will be unable to search on individual pieces of the IP address without using wildcard characters. Indexes created using outer segmentation tend to be marginally more efficient than those created with full segmentation, but are not quite as efficient as those created through inner segmentation.

- **Full segmentation** is in some respects a combination of inner and outer segmentation. Under full segmentation, the IP address is indexed both as a major segment and as a variety of minor segments, including minor segment combinations like 172.26 and 172.26.34. This is the least efficient indexing option, but it provides the most versatility in terms of searching.

**Note:** By default, index-time segmentation is set to a combination of inner and outer segmentation, and search-time segmentation is set to full segmentation.

For more information about changing the segmentation level, see Configure segmentation to manage disk usage in the Admin manual.
Defining segmentation rules for specific hosts, sources, or source types

A Splunk admin can define index time and search time segmentation rules that apply specifically to events with particular hosts, sources, or sourcetypes. If you run searches that involve a particular sourcetype on a regular basis, you could use this to improve the performance of those searches. Similarly, if you typically index a large number of syslog events, you could use this feature to help decrease the overall disk space that those events take up.

For details about how to set these special segmentation rules up, see Configure custom segmentation for a host, source, or source type in the Admin manual.

Assign metadata to events dynamically

This feature allows you to dynamically assign metadata to files as they are being consumed by Splunk. Use this feature to specify source type, host, or other metadata dynamically for incoming data. This feature is useful mainly with scripted data -- either a scripted input or a pre-existing file processed by a script.

Important: Splunk does not recommend using dynamic metadata assignment with ongoing monitoring (tail) inputs. For more information about file inputs, refer to Monitor files and directories in this manual.

To use this feature, you append a single dynamic input header to your file and specify the metadata fields you want to assign values to. The metadata fields most likely to be of interest are sourcetype, host, and source. You can see the list of all available pipeline metadata fields in transforms.conf.spec.

You can use this method to assign metadata instead of editing inputs.conf, props.conf and transforms.conf.

Configure a single input file

To use this feature for an existing input file, edit the file (either manually or with a script) to add a single input header:

```text
***SPLUNK*** <metadata field>=<string> <metadata field>=<string> ...
```

- Set `<metadata field>=<string>` to a valid metadata/value pair. You can specify multiple pairs. For example, `sourcetype=log4j host=swan`.
- Add the single header anywhere in your file. Any data following the header will be appended with the attributes and values you assign until the end of the file is reached.
- Add your file to `$SPLUNK_HOME/var/spool/splunk` or any other directory being monitored by Splunk.

Configure with a script

In the more common scenario, you write a script to dynamically add an input header to your incoming data stream. Your script can also set the header dynamically based on the contents of the input file.
Anonymize data with sed

This utility allows you to anonymize your data by replacing or substituting strings in it at index time using a sed script.

Most UNIX users are familiar with sed, a Unix utility which reads a file and modifies the input as specified by a list of commands. Now, you can use sed-like syntax to anonymize your data from props.conf.

Note: Edit or create a copy of props.conf in $SPLUNK_HOME/etc/system/local.

Define the sed script in props.conf

In a props.conf stanza, use SEDCMD to indicate a sed script:

```
[<stanza_name>]
SEDCMD-<class> = <sed script>
```

The stanza_name is restricted to the host, source, or sourcetype that you want to modify with your anonymization or transform.

The sed script applies only to the _raw field at index time. Splunk currently supports the following subset of sed commands: replace (s) and character substitution (y).

Note: You need to restart Splunk to implement the changes you made to props.conf

Replace strings with regex match

The syntax for a sed replace is:

```
SEDCMD-<class> = s/<regex>/<replacement>/flags
```

- regex is a PERL regular expression.
- replacement is a string to replace the regex match and uses \n for back-references, where n is a single digit.
- flags can be either: "g" to replace all matches or a number to replace a specified match.

Example

Let's say you want to index data containing social security numbers and credit card numbers. At index time, you want to mask these values so that only the last four digits are evident in your events. Your props.conf stanza may look like this:

```
[source::.../accounts.log]
SEDCMD-accounts = s/ssn=\d{5}(\d{4})/ssn=xxxxx\1/g s/cc=(\d{4}-){3}(\d{4})/cc=xxxx-xxxx-xxxx-\2/g
```

Now, in you accounts events, social security numbers appear as ssn=xxxxx6789 and credit card numbers will appear as cc=xxxx-xxxx-xxxx-1234.
Substitute characters

The syntax for a sed character substitution is:

```
SEDCMD-<class> = y/<string1>/<string2>/
```

which substitutes each occurrence of the characters in `string1` with the characters in `string2`.

**Example**

Let's say you have a file you want to index, `abc.log`, and you want to substitute the capital letters "A", "B", and "C" for every lowercase "a", "b", or "c" in your events. Add the following to your `props.conf`:

```
[source:..../abc.log]
SEDCMD-abc = y/abc/ABC/
```

Now, if you search for `source="*/abc.log"`, you should not find the lowercase letters "a", "b", and "c" in your data at all. Splunk substituted "A" for each "a", "B" for each "b", and "C" for each "c".

**Anonymize data using configuration files**

Anonymize data using configuration files

You may want to mask sensitive personal data that goes into logs. Credit card numbers and social security numbers are two examples of data that you may not want to index in Splunk. This page shows how to mask part of confidential fields so that privacy is protected but there is enough of the data remaining to be able to use it to trace events.

This example masks all but the last four characters of fields `SessionId` and `Ticket number` in an application server log.

An example of the desired output:

```
SessionId=###########7BEA&Ticket=############96EE
```

A sample input:

```
"2006-09-21, 02:57:11.60", 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&AccountId=&AgentHost=man&AgentId=man, MANUser: Version=1&Name=&Debit=&Credit=&AccessTime=&BillDay=&Status=&Language=&Country=&Email=&EmailNotify=&Pin=&PinPayment=&PinAmount=&PinPG=&PinPGRate=&PinMenu=&",
```

**Configuration**

To mask the data, modify the `props.conf` and `transforms.conf` files in your `$SPLUNK_HOME/etc/system/local/` directory.
props.conf

Edit $SPLUNK_HOME/etc/system/local/props.conf and add the following:

```
[<spec>]
TRANSFORMS-anonymize = session-anonymizer, ticket-anonymizer
```

<spec> can be:

1. `<sourcetype>`, the source type of an event
2. `host::<host>`, where `<host>` is the host for an event
3. `source::<source>`, where `<source>` is the source for an event.

`session-anonymizer` and `ticket-anonymizer` are TRANSFORMS class names whose actions are defined in `transforms.conf`. For your data, use the class names you create in `transforms.conf`.

transforms.conf

In $SPLUNK_HOME/etc/system/local/transforms.conf, add your TRANSFORMS:

```
[session-anonymizer]
REGEX = (?m)^(.*)SessionId=\w+\(\w{4}\)[&"].*$
FORMAT = $1SessionId=########$2
DEST_KEY = _raw

[ticket-anonymizer]
REGEX = (?m)^(.*)Ticket=\w+\(\w{4}\)[&"].*$
FORMAT = $1Ticket=########$2
DEST_KEY = _raw
```

REGEX should specify the regular expression that will point to the string in the event you want to anonymize.

**Note:** The regex processor can't handle multi-line events. To get around this you need to specify in `transforms.conf` that the event is multi-line. Use the (?m) before the regular expression.

FORMAT specifies the masked values. $1$ is all the text leading up to the regex and $2$ is all the text of the event after the regex.

DEST_KEY = _raw specifies to write the value from FORMAT to the raw value in the log - thus modifying the event.
Configure event timestamping

How timestamp assignment works

Splunk uses timestamps to correlate events by time, create the timeline histogram in Splunk Web and to set time ranges for searches. Timestamps are assigned to events at index time. Most events get a timestamp value assigned to them based on information in the raw event data. If an event doesn’t contain timestamp information, Splunk attempts to assign a timestamp value to the event as it’s indexed. Splunk stores timestamp values in the _time field (in UTC time format).

Timestamp processing is one of the key steps in event processing. For more information on event processing, see Configure event processing.

Considerations when adding new data

If your data turns out to require timestamp configuration beyond what Splunk does automatically, you must re-index that data once you've configured its timestamp extraction. It's a good idea to test a new data input in a "sandbox" Splunk instance (or just a separate index) before adding it to your production Splunk instance in case you have to clean it out and re-index it a few times to get it just right.

Precedence rules for timestamp assignment

Splunk uses the following precedence to assign timestamps to events:

1. Look for a time or date in the event itself using an explicit TIME_FORMAT if provided.

   Use positional timestamp extraction for events that have more than one timestamp value in the raw data.

2. If no TIME_FORMAT is provided, or no match is found, attempt to automatically identify a time or date in the event itself.

   Use positional timestamp extraction for events that have more than one timestamp value in the raw data.

3. If an event doesn’t have a time or date, use the timestamp from the most recent previous event of the same source.

4. If no events in a source have a date, look in the source (or file) name (Must have time in the event).

5. For file sources, if no time or date can be identified in the file name, use the modification time on the file.

6. If no other timestamp is found, set the timestamp to the current system time (at the event’s index time).
Configure timestamps

Most events don’t require any special timestamp handling. For some sources and distributed deployments, you may have to configure timestamp formatting to extract timestamps from events. Configure Splunk’s timestamp extraction processor by editing props.conf. For a complete discussion of the timestamp configurations available in props.conf, see Configure timestamp recognition.

You can also configure Splunk’s timestamp extraction processor to:

- Apply timezone offsets.
- Pull the correct timestamp from events with more than one timestamp.
- Improve indexing performance.

Finally, train Splunk to recognize new timestamp formats.

**Configure timestamp recognition**

Splunk uses timestamps to correlate events by time, create the histogram in Splunk Web and to set time ranges for searches. Timestamps are assigned to events at index time.

Splunk assigns a timestamp to most events based on information in the raw event data. If an event doesn’t contain timestamp information, Splunk attempts to assign a timestamp value to the event as it’s indexed. Splunk stores timestamp values in the _time field (in UTC time format).

Most events don’t require any special timestamp handling; you can just let Splunk handle it without any configuration.

**Precedence rules for timestamp assignment**

Splunk uses the following precedence to assign timestamps to events:

1. Look for a time or date in the event itself using an explicit TIME_FORMAT if provided.

Use positional timestamp extraction for events that have more than one timestamp value in the raw data.

2. If no TIME_FORMAT is provided, or no match is found, attempt to automatically identify a time or date in the event itself.

3. If an event doesn’t have a time or date, use the timestamp from the most recent previous event of the same source.

4. If no events in a source have a time or date, look in the source (or file) name.

5. For file sources, if no time or date can be identified in the file name, use the modification time on the file.
6. If no other timestamp is found, set the timestamp to the current system time (the time at which the event is indexed by Splunk).

Configure timestamps

Most events don't require any special timestamp handling; you can just let Splunk handle it without any configuration.

For some sources and distributed deployments, you may have to configure timestamp formatting to extract timestamps from events. Configure Splunk's timestamp extraction processor by editing props.conf.

Configure how Splunk recognizes timestamps by editing props.conf. Splunk uses strftime() formatting to identify timestamp values in your events. Specify what Splunk recognizes as a timestamp by setting a strftime() format in the TIME_FORMAT= key.

**Note:** If your event has more than one timestamp, set Splunk to recognize the correct timestamp with positional timestamp extraction.

Use $SPLUNK_HOME/etc/system/README/props.conf.example as an example, or create your own props.conf. Make any configuration changes to a copy of props.conf in $SPLUNK_HOME/etc/system/local/, or your own custom application directory in $SPLUNK_HOME/etc/apps/.

Configure any of the following attributes in props.conf to set Splunk's timestamp recognition. Refer to $SPLUNK_HOME/etc/system/README/props.conf.spec for full specification of the keys.

```
[<spec>]
DATETIME_CONFIG = <filename relative to $SPLUNK_HOME>
MAX_TIMESTAMP_LOOKAHEAD = <integer>
TIME_PREFIX = <regular expression>
TIME_FORMAT = <strftime-style format>
TZ = <posix timezone string>
MAX_DAYS_AGO = <integer>
MAX_DAYS_HENCE = <integer>
MAX_DIFF_SECS_AGO = <integer>
MAX_DIFF_SECS_HENCE = <integer>
```

- `<spec>` indicates what to apply timestamp extraction to. This can be one of the following:
  - `<sourcetype>`, the sourcetype of an event.
  - `host::<host>`, where `<host>` is the host of an event.
  - `source::<source>`, where `<source>` is the source of an event.
- If an event contains data that matches the value of `<spec>`, then the timestamp rules specified in the stanza apply to that event.
- Add additional stanzas to customize timestamp recognition for any type of event.

```bash
DATETIME_CONFIG = <filename relative to $SPLUNK_HOME>
```
• Specify a file to use to configure Splunk's timestamp processor (by default Splunk uses $SPLUNK_HOME/etc/datetime.xml).
• To use a custom datetime.xml, specify the correct path to your custom file in all keys that refer to datetime.xml.
• Set DATETIME_CONFIG = NONE to prevent the timestamp processor from running. When timestamp processing is off, Splunk 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, this means that Splunk derives the event timestamp from the modification time of the input file.
• Set DATETIME_CONFIG = CURRENT to assign the current system time to each event as it's indexed.

TIME_PREFIX = <regular expression>

• When set, Splunk looks for a match for this regex in the event text 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 regex match.
• Ideally you should use a regular expression that points to the space 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.
• Defaults to empty string.

MAX_TIMESTAMP_LOOKAHEAD = <integer>

• Specify how far (how many characters) into an event Splunk should look for a timestamp.
• This constraint to timestamp extraction is applied from the point of the TIME_PREFIX-set location.
  ♦ For example, if TIME_PREFIX positions a location 11 characters into the event, and MAX_TIMESTAMP_LOOKAHEAD is set to 10, timestamp extraction will be constrained to characters 11 through 20.
• If set to 0 or -1, the length constraint for timestamp recognition is effectively disabled. This can have negative performance implications which scale with the length of input lines (or with event size when LINE_BREAKER is redefined for event splitting).
• Default is 150 characters.

TIME_FORMAT = <strptime-style format>

• Specifies a strftime() format string to extract the date.
• Unix strftime() is an industry standard for designating time formats. For more information, see the section "Enhanced strftime() support", below.
• For best results, the <strftime-style format> should describe the day of the year and the time of day.
• TIME_FORMAT starts reading after the TIME_PREFIX.
• 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.
• Default is empty.

**TZ** = <timezone identifier>

• Splunk's logic for determining a particular event's time zone is as follows:
  ♦ If the event has a timezone in its raw text (such as UTC or -08:00, use that).
  ♦ Otherwise, if TZ is set to a valid timezone string, use that. Specify a timezone setting using a value from the zoneinfo TZID database.
  ♦ Otherwise, use the timezone of the system that is running splunkd.
• For more details and examples, see "Specify timezones of timestamps", in this manual.
• Defaults to empty.

**MAX_DAYS_AGO** = <integer>

• 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 then Splunk ignores dates older than 10 days from the current date.
• Default is 2000.
• **Note:** If you have data that is more than 2000 days old, increase this setting.

**MAX_DAYS_HENCE** = <integer>

• 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, dates that are more than 3 days in the future are ignored.
• False positives are less likely with a tighter window.
• If your servers have the wrong date set or are in a timezone that is one day ahead, set this value to at least 3.
• Defaults to 2. This allows timestamp extractions that are up to a day in the future.

**MAX_DIFF_SECS_AGO** = <integer>

• If the event's timestamp is more than <integer> seconds BEFORE the previous timestamp, only accept it if it has the same exact time format as the majority of timestamps from the source.
• **IMPORTANT:** If your timestamps are wildly out of order, consider increasing this value.
• Defaults to 3600.

**MAX_DIFF_SECS_HENCE** = <integer>

• If the event's timestamp is more than <integer> seconds AFTER the previous timestamp, only accept it if it has the same exact time format as the majority of timestamps from the source.
• **IMPORTANT:** If your timestamps are wildly out of order, or if you have logs that are written less than once a week, consider increasing this value.
• Defaults to 604800 (one week).
**Enhanced strftime() support**

Configure timestamp parsing in props.conf with the `TIME_FORMAT=` key. Splunk implements an enhanced version of Unix `strftime()` that supports additional formats (allowing for microsecond, millisecond, any time width format, and some additional time formats for compatibility). See the table below for a list of the additionally supported `strftime()` formats.

In previous versions, Splunk parsed timestamps using only the standard Linux `strftime()` conversion specifications. Now, in addition to standard Unix `strftime()` formats, Splunk's `strftime()` implementation supports recognition of the following date-time formats:

- `%N` For GNU date-time nanoseconds. Specify any sub-second parsing by providing the width: `%3N = milliseconds, %6N = microseconds, %9N = nanoseconds.
- `%Q,%q` For milliseconds, microseconds for Apache Tomcat. `%Q` and `%q` can format any time resolution if the width is specified.
- `%I` For hours on a 12-hour clock format. If `%I` appears after `%S` or `%s` (like "%H:%M:%S.%l") it takes on the log4cpp meaning of milliseconds.
- `%+` For standard UNIX date format timestamps.
- `%v` For BSD and OSX standard date format.
- `%z, %::z, %:::z` GNU libc support.
- `%o` For AIX timestamp support (%o used as an alias for `%Y`).
- `%p` The locale's equivalent of AM or PM. (Note: there may be none.)

**strptime() format expression examples**

Here are some sample date formats with the `strftime()` expressions that handle them:

<table>
<thead>
<tr>
<th>Date Format</th>
<th>strftime() 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>
<tr>
<td>Jan 24, 2003</td>
<td>%b %d, %Y</td>
</tr>
<tr>
<td>January 24, 2003</td>
<td>%B %d, %Y</td>
</tr>
<tr>
<td>q</td>
<td>25 Feb '03 = 2003-02-25</td>
</tr>
</tbody>
</table>

**Note:** Splunk does not currently recognize non-English month names in timestamps. If you have an app that's writing non-English month names to log files, reconfigure the app to use numerical months, if possible.

**Examples**

Your data might contain an easily recognizable timestamp to extract such as:

```
...FOR: 04/24/07 PAGE 01...
```

The entry in `props.conf` is:

```
[host::foo]
TIME_PREFIX = FOR:
TIME_FORMAT = %m/%d/%y
```

125
Your data might contain other information that Splunk parses as timestamps, for example:

...1989/12/31 16:00:00 ed May 23 15:40:21 2007...

Splunk extracts the date as Dec 31, 1989, which is not useful. In this case, configure `props.conf` to extract the correct timestamp from events from `host::foo`:

```
[host::foo]
TIME_PREFIX = \d{4}/\d{2}/\d{2} \d{2}:\d{2}:\d{2} \w+\s
TIME_FORMAT = %b %d %H:%M:%S %Y
```

This configuration assumes that all timestamps from `host::foo` are in the same format. Configure your `props.conf` stanza to be as granular as possible to avoid potential timestamping errors.

Configure timestamps in other ways

You can also configure Splunk's timestamp extraction processor to:

- Apply timezone offsets.
- Pull the correct timestamp from events with more than one timestamp.
- Improve indexing performance.
- Train Splunk to recognize new timestamp formats.

In addition, you can use your browser's locale setting to configure how the browser formats Splunk timestamps. For information on the setting the browser locale, see User language and locale.

Answers

Have questions? Visit Splunk Answers and see what questions and answers the Splunk community has around timestamp recognition and configuration.

**Improve Splunk's ability to recognize timestamps**

Improve Splunk's ability to recognize timestamps

Splunk recognizes most timestamps by default. For more information read How timestamps work. If Splunk doesn't recognize a particular timestamp, you can use the `train dates` command to teach Splunk the pattern. The output of `train dates` is a regular expression that you can add to `datetime.xml` and `props.conf` to configure the unique timestamp extraction.

The `train` command lets you interactively teach Splunk new patterns for timestamps, fields, and sourcetypes. for more information about `train` and the different arguments you can use with it, go to `$SPLUNK_HOME/bin` and refer to the `train` help page:

```
./splunk help train
```

**Important:** Use `train dates` only when you can't configure the timestamp with `props.conf`. 

126
Steps to configure timestamps with train dates

To teach Splunk a new timestamp pattern, complete the following steps:

1. Copy a sampling of your timestamp data into a plain text file.

Splunk learns the pattern of the timestamp based on the patterns in this text file.

2. Run the train dates command.

This feature is interactive. When prompted, provide the path to the text file containing your timestamp data. The command produces a regular expression for your timestamp.

3. Create a custom datetime.xml.

Copy the output of the train command into a copy of datetime.xml file.

**Note:** The default datetime.xml file is located in $SPLUNK_HOME/etc/datetime.xml. Do not modify this file; instead, copy the default datetime.xml into a custom application directory in $SPLUNK_HOME/etc/apps/ or $SPLUNK_HOME/etc/system/local/. Refer to the topic about applications in the Admin manual for more information.

4. Edit your local props.conf.

Include the path to your custom datetime.xml file in the relevant stanzas.

```
./splunk [command]
```

**Run the train dates command**

The train command is an interactive CLI tool. For Splunk to learn a new date format, you need to explicitly provide a file and pattern. Afterwards, Splunk returns a string for you to add to datetime.xml.

1. To begin training Splunk to recognize a new timestamp, go to $SPLUNK_HOME/bin and type:

   ./splunk train dates

Splunk prompts you for an action:

```
What operation do you want to perform? (default=learn)
```

Enter choice: [Learn]/[Test]/[Quit] >

The default action is **Learn**.

2. To perform the training operation, type "L", "l", or "learn". Click enter.

Splunk prompts you to give it the sample file you want to use to train it:
Enter full filename from which to learn dates > sampling.txt

3. Enter the path of the file on your Splunk server (this step doesn't allow tab-complete).

Splunk displays the first line of your sample and asks you to teach it the values for the timestamp:

---------------------------
Interactively learning date formats.
---------------------------
INSTRUCTIONS: If a sample line does not have a timestamp, hit Enter. If it does have a timestamp, enter the timestamp separated by commas in this order: month, day, year, hour, minute, second, ampm, timezone. Use a comma as a placeholder for missing values. For example, for a sample line like this "[Jan/1/08 11:56:45 GMT] login", the input should be: "Jan, 1, 08, 11, 56, 45, , GMT" (note missing AM/PM). Spaces are optional.
SAMPLE LINE 1:
Tue Jul 10 21:23:06 PDT 2007 Received Trade 330 with detail user: user3456 date: date: 10Jul200721:23:06 action: sell 3583 MNAG @ 42
---------------------------
Enter timestamp values as: month, day, year, hour, minute, second, ampm, timezone. > 7, 10, 2007, 9, 23, 06, pm, PDT

4. Enter values for month, day, year, hour, minute, second, ampm, and timezone (as shown above). This trains Splunk to recognize the values you enter as the designated portions of the timestamp.

If the values are sufficient, Splunk displays:

Learned pattern.

If you are satisfied that the timestamps formats have been learned, hit control-c.

5. After you hit control-c, Splunk displays:

Patterns Learned.
It is highly recommended that you make changes to a copy of the default datetime.xml file. For example, copy "/Applications/splunk/etc/datetime.xml" to "/Applications/splunk/etc/system/local/datetime.xml". In that custom file, add the below timestamp definitions, and add the pattern names to timePatterns and datePatterns list. For more details, see http://www.splunk.com/doc/latest/admin/TrainTimestampRecognition

<define name="trainwreck_1_date" extract="day,litmonth,year,">
  <text><![CDATA[:\d+\s\w+\s(\d+)\s(\w+)\s(\d+)]]></text>
</define>
<define name="trainwreck_1_time" extract="hour,minute,second,ampm,">
  <text><![CDATA[(\d+):(\d+):(\d+)\s(\w+)]]></text>
</define>

What operation do you want to perform? (default=learn)

Enter choice: [Learn]/Test/Quit > q

6. Check the output.
Create a custom datetime.xml

After running train, Splunk outputs a string describing the new timestamp pattern.

In your custom datetime.xml file:

1. Paste the string returned from train before the <timePatterns> and <datePatterns> stanzas.

2. Add <use name="define name"/> for both <timePatterns> and <datePatterns> with the string defined as the <define name="string">.

Example:

For the following train dates output:

```xml
<define name="_utcepoch" extract="utcepoch">
  <text><![CDATA[(((?<=^|\[s","=\(\[\]{})(?:1[01]|9)\d{8}|^@[\{da-fA-F}\{16,24}\}(?:\d{3})?(?![]\[]\[]\[]\[)]]></text>
</define>
```

The modified datetime.xml file might look something like:

```xml
<define name="_utcepoch" extract="utcepoch">
  <text><![CDATA[(((?<=^|\[s","=\(\[\]{})(?:1[01]|9)\d{8}|^@[\{da-fA-F}\{16,24}\}(?:\d{3})?(?![]\[]\[]\[]\[)]]></text>
</define>
<timePatterns>
  <use name="_time"/>
  <use name="_hmtime"/>
  <use name="_hmtime"/>
  <use name="_dottime"/>
  <use name="_combdatetime"/>
  <use name="_utcepoch"/>
</timePatterns>
<define name="_utcepoch" extract="utcepoch">
  <text><![CDATA[(((?<=^|\[s","=\(\[\]{})(?:1[01]|9)\d{8}|^@[\{da-fA-F}\{16,24}\}(?:\d{3})?(?![]\[]\[]\[]\[)]]></text>
</define>
<datePatterns>
  <use name="_usdate"/>
  <use name="_isodate"/>
  <use name="_eurodate"/>
  <use name="_bareurlitdate"/>
  <use name="_orddate"/>
  <use name="_combdatetime"/>
  <use name="_masheddate"/>
  <use name="_masheddate2"/>
  <use name="_utcepoch"/>
</datePatterns>
```
Edit your local props.conf

To apply your custom timestamp, Splunk needs to know where to find your new datetime.xml.

Modify props.conf to:

1. Add a DATETIME_CONFIG key to the timestamp configuration stanzas.

2. Set the value of DATETIME_CONFIG to the path of your custom datetime.xml.

**Note:** See all of the keys you can set in a stanza to configure timestamp recognition.

**Example:**

This example applies a custom datetime.xml to events from the host, "london".

```
[host::london]
DATETIME_CONFIG = /etc/system/local/datetime.xml
```

You can set custom timestamp extraction patterns for any host, source, or sourcetype by editing props.conf.

**Configure timestamp assignment for events with multiple timestamps**

If an event contains more than one recognizable timestamp, you can tell Splunk to use a particular timestamp. This is especially useful when indexing events that contain syslog host-chaining data.

Configure positional timestamp extraction by editing props.conf.

**Configure positional timestamp extraction in props.conf**

Configure Splunk to recognize a timestamp anywhere in an event by adding TIME_PREFIX = and MAX_TIMESTAMP_LOOKAHEAD = keys to a [<spec>] stanza in props.conf. Set a value for MAX_TIMESTAMP_LOOKAHEAD to tell Splunk how far into an event to look for the timestamp. Set a regular expression value for TIME_PREFIX to tell Splunk what pattern of characters to look for to indicate the beginning of the timestamp.

When TIME_PREFIX is set, Splunk scans the event text for a match to its regex in the event text before it tries to extract a timestamp. Splunk's timestamping algorithm only looks for a timestamp in the text following the end of the first regex match. So if TIME_PREFIX is set to abc123, only the text following the first occurrence of abc123 is used for timestamp extraction.

This also sets the start point for MAX_TIMESTAMP_LOOKAHEAD; it will start after the matched portion of text in the TIME_PREFIX regex. For example, if TIME_PREFIX matches text in the first 11 characters of the event and the timestamp you want to extract is always within the next 20-30 characters, you may want to set MAX_TIMESTAMP_LOOKAHEAD=30 just to be safe and ensure that
the timestamp is located and extracted correctly.

**Note:** Use `$SPLUNK_HOME/etc/system/README/props.conf.example` as an example, or create your own `props.conf`. Make any configuration changes to a copy of `props.conf` in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`.

**Example:** If an event looks like:

```
1989/12/31 16:00:00 ed May 23 15:40:21 2007 ERROR UserManager - Exception thrown Ignoring unsupported search for eventtype: /doc sourcetype="access_combined" NOT eventtypetag=bot
```

To identify the timestamp: **May 23 15:40:21 2007**

Configure `props.conf`:

```
[source::/Applications/splunk/var/spool/splunk]
TIME_PREFIX = \d{4}/\d{2}/\d{2} \d{2}:\d{2}:\d{2} \w+\s
MAX_TIMESTAMP_LOOKAHEAD = 44
```

What this configuration does is instruct Splunk to locate events that match the first timestamp construction, but *ignore* that timestamp in favor of a timestamp that might occur within the following 44 characters (a number it gets from the `MAX_TIMESTAMP_LOOKAHEAD` attribute). The second timestamp in the event is the one you want, and Splunk will find it because it always lands within that 44 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 needed for the timestamp you want to extract. In this example `MAX_TIMESTAMP_LOOKAHEAD` is optimized to look 44 characters into the event.

### Specify timezones of timestamps

Specify timezones of timestamps

If you’re indexing data from different timezones, use timezone offsets to ensure that they’re correctly correlated when you search. You can configure timezones based on the host, source, or source type of an event.

Configure timezones in `props.conf`. By default, Splunk applies timezones using these rules, in the following order:

1. Use the timezone in raw event data (for example, PST, -0800).
2. Use `TZ` if it is set in a stanza in `props.conf` and the event matches the host, source, or source type specified by a stanza.
3. Use the timezone of the Splunk server that indexes the event.

**Note:** If you change the timezone setting in the OS Splunk is running on, you must restart Splunk for it to pick up the change.
Specify time zones in props.conf

Use $SPLUNK_HOME/etc/system/README/props.conf.example as an example, or create your own props.conf. Make any configuration changes to a copy of props.conf in $SPLUNK_HOME/etc/system/local/, or your own custom application directory in $SPLUNK_HOME/etc/apps/.

Configure time zones by adding a `TZ =` key to a timestamp configuration stanza for a host, source, or sourcetype in props.conf. The Splunk `TZ =` key recognizes zoneinfo TZID's (See all the timezone TZ ID's in the zoneinfo (TZ) database). Set a `TZ =` value to a TZID of the appropriate timezone for any host, source, or source type. The TZ for a host, source, or source type should be set to the timezone of the events coming from that host, source, or sourcetype.

Note that the timezone of the indexer is not configured in Splunk. As long as the time is set correctly on the host OS of the indexer, offsets to event timezones will be calculated correctly.

**Examples**

Events are coming to this indexer from New York City (in the US/Eastern timezone) and Mountain View, California (US/Pacific). To correctly handle the timestamps for these two sets of events, the `props.conf` for the indexer needs the timezone to be specified as US/Eastern and US/Pacific respectively.

The first example sets the timezone of events from host names that match the regular expression `nyc.*` with the US/Eastern timezone.

```
[host::nyc*]
TZ = US/Eastern
```

The second example sets the timezone of events from sources in the path `/mnt/ca/...` with the US/Pacific timezone.

```
[source::/mnt/ca/...]
TZ = US/Pacific
```

**zoneinfo (TZ) database**

The zoneinfo database is a publicly maintained database of timezone values.

- UNIX versions of Splunk rely on a TZ database included with the UNIX distribution you're installing on. Most UNIX distributions store the database in the directory: `/usr/share/zoneinfo`.
- Solaris versions of Splunk store TZ information in this directory: `/usr/share/lib/zoneinfo`.
- Windows versions of Splunk ship with a copy of the TZ database.

Refer to the zoneinfo (TZ) database for values you can set as `TZ =` in `props.conf`. 

132
Tune timestamp recognition for better indexing performance

Tune Splunk’s timestamp extraction by editing props.conf. Adjust how far Splunk’s timestamp processor looks into events, or turn off the timestamp processor to make indexing faster.

**Note:** Use $SPLUNK_HOME/etc/system/README/props.conf.example as an example, or create your own props.conf. Make any configuration changes to a copy of props.conf in $SPLUNK_HOME/etc/system/local/, or your own custom application directory in $SPLUNK_HOME/etc/apps/. For more information on configuration files in general, see how configuration files work.

Adjust timestamp lookahead

Timestamp lookahead determines how far (how many characters) into an event the timestamp processor looks for a timestamp. Adjust how far the timestamp processor looks by setting a value (the number of characters) for the `MAX_TIMESTAMP_LOOKAHEAD` key in any timestamp stanza.

**Note:** You can set `MAX_TIMESTAMP_LOOKAHEAD` to different values for each timestamp stanza.

The default number of characters that the timestamp processor looks into an event is 150. Set `MAX_TIMESTAMP_LOOKAHEAD` to a lower value to speed up how fast events are indexed. You should do this if your timestamps occur in the first part of your event.

If your events are indexed in real time, increase Splunk’s overall indexing performance by turning off timestamp recognition. Set `DATETIME_CONFIG = CURRENT` to cause Splunk to not look into events for their timestamps and instead assign the current system time (at the time of indexing) to each event.

**Example:**

This example tells the timestamp processor to look 20 characters into events from source foo.

```
[source::foo]
MAX_TIMESTAMP_LOOKAHEAD = 20
...
```

Disable timestamp determination

Turn off the timestamp processor entirely to significantly improve indexing performance. Turn off timestamp processing for events matching a host, source, sourcetype specified by a timestamp stanza by adding a `DATETIME_CONFIG` key to a stanza and setting the value to `NONE`. When timestamp processing is off, Splunk 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 (such as monitor) this means that Splunk derives the event timestamp from the modification time of the input file.

Example:

This example turns off timestamp extraction for events that come from the source `foo`.

```
[source::foo]
DATETIME_CONFIG = NONE
...
Configure indexed field extraction

About default fields (host, source, sourcetype, and more)

About default fields (host, source, sourcetype, and more)

If you've read the "Configure event processing" chapter in this manual, you know that Splunk automatically extracts a number of default fields for each event it processes prior to indexing. These default fields include index, which identifies the index in which the related event is located, linecount, which describes the number of lines the related event contains, and timestamp, which describes the point in time at which an event occurred. (As discussed in the "Configure event timestamping" chapter, you have a number of options when it comes to changing the manner in which sets of events are timestamped.)

Note: For a complete list of the default fields that Splunk identifies for each event prior to indexing, see "Use default fields" in the User manual.

This chapter focuses mainly on three important default fields: host, source, and sourcetype. Splunk identifies host, source, and sourcetype values for each event it processes. This chapter explains how Splunk does this, and shows you how you can override the automatic assignment of host and sourcetype values for events when it is necessary to do so.

This chapter also shows you how to have Splunk extract additional, custom fields at index time. It's important to note that this practice is strongly discouraged, however. Adding to the list of indexed fields can negatively impact indexing and search speed. In addition, it may require you to reindex your entire dataset in order to have those fields show up for previously indexed events. It's best to extract fields at search time whenever possible. For more information, see "Index time versus search time" in the Admin manual.

For more information about index-time field extraction, see "Configure index-time field extraction" in this manual.

Defining host, source, and sourcetype

The host, source, and sourcetype fields are defined as follows:

- **host** - An event's host value is typically the hostname, IP address, or fully qualified domain name of the network host from which the event originated. The host value enables you to easily locate data originating from a specific device. For an overview of the methods Splunk provides for the override of automatic host assignment, see the "About hosts" section of this topic.
- **source** - The source of an event is the name of the file, stream, or other input from which the event originates. For data monitored from files and directories, the value of source is the full path, such as /archive/server1/var/log/messages.0 or /var/log/. The value of source for network-based data sources is the protocol and port, such as UDP:514.
- **sourcetype** - The source type of an event is the format of the data input from which it originates, such as access_combined or cisco_syslog. For an overview of how Splunk sets the source type value and the ways you can override automatic sourcetyping, see the "Override automatic source type assignment" topic in this manual.
Under what conditions should you override host and sourcetype assignment?

Much of the time, Splunk can automatically identify host and sourcetype values that are both correct and useful. But situations do come up that require you to intervene in this process and provide override values.

You may want to change your default host assignment when:

- you are bulk-loading archive data that was originally generated from a different host and you want those events to have that host value.
- your data is actually being forwarded from a different host (the forwarder will be the host unless you specify otherwise).
- you are working with a centralized log server environment, which means that all of the data received from that server will have the same host even though it originated elsewhere.

You may want to change your default sourcetype assignment when:

- you want to give all event data coming through a particular input or from a specific source the same source type, for tracking purposes.
- you want to apply source types to specific events coming through a particular input, such as events that originate from a discrete group of hosts, or even events that are associated with a particular IP address or userid.

There are also steps you can take to expand the range of source types that Splunk automatically recognizes, or to simply rename source types. See the "Source type field overview" section, below, for more information.

About hosts

An event's host field value is the name of the physical device from which the event originates. Because it is a default field, which means that Splunk assigns it to every event it indexes, you use it to search for all events that have been generated by a particular host.

The host value can be an IP address, device hostname, or a fully qualified domain name, depending on whether the event was received through a file input, network input, or the computer hosting the instance of Splunk.

How Splunk assigns the host value

If no other host rules are specified for a source, Splunk assigns host a default value that applies to all data coming from inputs on a given Splunk server. The default host value is the hostname or IP address of the network host. When Splunk is running on the server where the event occurred (which is the most common case) this is correct and no manual intervention is required.

For more information, see "Set a default host for a Splunk server" in this manual.

Set a default host for a file or directory input

If you are running Splunk on a central log archive, or you are working with files forwarded from other hosts in your environment, you may need to override the default host assignment for events coming
from particular inputs.

There are two methods for assigning a host value to data received through a particular input. You can define a static host value for all data coming through a specific input, or you can have Splunk dynamically assign a host value to a portion of the path or filename of the source. The latter method can be helpful when you have a directory structure that segregates each host's log archive in a different subdirectory.

For more information, see "Set a default host for a file or directory input" in this manual.

Override default host values based on event data

You may have a situation that requires you to override host values based on event data. For example, if you work in a centralized log server environment, you may have several host servers that feed into that main log server. The central log server is called the reporting host. The system where the event occurred is called the originating host (or just the host). In these cases you need to define rules that override the automatic host assignments for events received from that centralized log host and replace them with distinct originating host values.

For more information, see "Override default host values based on event data" in this manual.

Tag host values

Tag host values to aid in the execution of robust searches. Tags enable you to cluster groups of hosts into useful, searchable categories.

For more information, see "About tags and aliases" in the Knowledge Manager manual.

About source types

Any common data input format can be a source type. Most source types are log formats. For example, the list of common source types that Splunk automatically recognizes includes:

- **access_combined**, for NCSA combined format HTTP Web server logs.
- **apache_error**, for standard Apache Web server error logs.
- **cisco_syslog**, for the standard syslog produced by Cisco network devices including PIX firewalls, routers, ACS, etc., usually via remote syslog to a central log host.
- **websphere_core**, which is a core file export from WebSphere.

**Note:** For a longer list of source types that Splunk automatically recognizes, see "List of pretrained sourcetypes" in this manual.

sourcetype is the name of the source type field. You can use the sourcetype field to find similar types of data from any source type. For example, you could search sourcetype=weblogic_stdout to find all of your WebLogic server events even when WebLogic is logging from more than one domain (or "host," in Splunk terms).
Source vs source type

The source is the name of the file, stream, or other input from which a particular event originates. For data monitored from files and directories, the value of source is the full path, such as /archive/server1/var/log/messages.0 or /var/log/. The value of source for network-based data sources is the protocol and port, such as UDP:514.

Events with the same source type can come from different sources. For example, say you’re monitoring source=/var/log/messages and receiving direct syslog input from udp:514. If you search sourcetype=linux_syslog, Splunk will return events from both of those sources.

Methods Splunk uses for source type assignment and their precedence

Splunk employs a variety of methods to assign source types to event data at index time. As it processes event data, Splunk steps through these methods in a defined order of precedence. It starts with hardcoded source type configurations in `inputs.conf` and `props.conf`, moves on to rule-based source type association, and then works through methods like automatic source type recognition and automatic source type learning. This range of methods enables you to configure how Splunk applies source type values to specific kinds of events, while letting Splunk assign source type values to the remaining events automatically.

The following list discusses these methods in the order that Splunk typically uses them to assign source types to event data at index time:

1. Explicit source type specification based on the data input, as configured in `inputs.conf` stanzas:

   \[\text{monitor://$PATH}\]
   sourcetype=$SOURCETYPE

2. Explicit source type specification based on the data source, as configured in `props.conf` stanzas:

   \[\text{source::$SOURCE}\]
   sourcetype=$SOURCETYPE

3. Rule-based source type recognition:

   Enables Splunk to match incoming data to source types using classification rules specified in `rule::` stanzas in `props.conf`.

   \[\text{rule::$RULE_NAME}\]
   sourcetype=$SOURCETYPE
   MORE_THAN_[0-100] = $REGEX
   LESS_THAN_[0-100] = $REGEX

   For information about setting up or removing source type recognition rules, see "Configure rule-based source type recognition" in this manual.

4. Automatic source type matching:
Splunk uses automatic source type recognition to match similar-looking files and, through that, assign a source type. It calculates signatures for patterns in the first few thousand lines of any file or stream of network input. These signatures identify things like repeating word patterns, punctuation patterns, line length, and so on. When Splunk calculates a signature, it compares it to previously seen signatures. If the signature appears to be a radically new pattern, Splunk creates a new source type for the pattern.

**Note:** At this stage in the source type assignation process, Splunk just matches incoming data with source types that it has learned previously. It doesn't create new source types for unique signatures until the final stage of source typing (step 6, below).

See "List of pretrained source types" in this manual for a list of the source types that Splunk can recognize out of the box. See "Train Splunk's source type autoclassifier" for more information about expanding the list of source types that Splunk can assign through automatic source type recognition.

5. **Delayed rule-based source type association:**

This works like rule-based associations (see above), except you create a delayedrule:: stanza in props.conf. This is a useful "catch-all" for source types, in case Splunk missed any with intelligent matching (see above).

A good use of delayed rule associations is for generic versions of very specific source types that are defined earlier with rule:: in step 3, above. For example, you could use rule:: to catch event data with specific syslog source types, such as "sendmail syslog" or "cisco syslog" and then have delayedrule:: apply the generic "syslog" source type to the remaining syslog event data.

```
[delayedrule::$RULE_NAME]
  sourcetype=$SOURCETYPE
  MORE_THAN_\[0-100\] = $REGEX
  LESS_THAN_\[0-100\] = $REGEX
```

For more information about setting up or removing delayed rules for source type recognition, see "Configure rule-based source type recognition" in this manual.

6. **Automatic source type learning:**

If Splunk is unable to assign a source type for the event using the preceding methods, it creates a new source type for the event signature (see step 4, above). Splunk stores learned pattern information in sourcetypes.conf.

**Set a default host for a Splunk server**

An event's host value is the IP address, host name, or fully qualified domain name of the physical device on the network from which the event originates. Because Splunk assigns a host value at index time for every event it indexes, host value searches enable you to easily find data originating from a specific device.
Default host assignment

If you have not specified other host rules for a source (using the information in this and subsequent topics in this chapter), the default host value for an event is typically the hostname, IP address, or fully qualified domain name of the network host from which the event originated. When the event originates from the server on which Splunk is running (which is the most common case) the host assignment is correct, and there’s no need for you to change anything. However, if you data is being forwarded from a different host, or if you’re bulk-loading archive data, you may want to change the default host value for that data.

To set the default value of the host field, you can use Splunk Manager, or edit inputs.conf.

Set the default host value using Manager

Use Manager to set the default host value for a server:

1. In Splunk Web, click on the Manager link in the upper right-hand corner of the screen.
2. In Manager, click System settings under System configurations.
4. On the General settings page, scroll down to the Index settings section and change the Default host name.
5. Save your changes.

This sets the value of the host field for all events that have not received another host name.

Set the default host value using inputs.conf

This host assignment is set in inputs.conf during Splunk installation. Modify the host entry by editing $SPLUNK_HOME/etc/system/local/, or your own custom application directory in $SPLUNK_HOME/etc/apps/. (We recommend using the latter directory if you want to make it easy to transfer your data customizations to other search servers).

This is the format of the host assignment in inputs.conf:

```
host = <string>
```

- Set `<string>` to your chosen default host value. `<string>` defaults to the IP address or domain name of the host where the data originated.
- This is a shortcut for `MetaData:Host = <string>`. It sets the host of events from this input to be the specified string. Splunk automatically prepends `host::` to the value when this shortcut is used.

Restart Splunk to enable any changes you have made to inputs.conf.
Override the default host value for data received from a specific input

If you are running Splunk on a central log archive, or you are working with files copied from other hosts in the environment, you may want to override the default host assignment for a particular input on a static or dynamic basis.

For more information, see "Set a default host value for an input" in this manual.

Override the default host value using event data

If you have a centralized log host sending events to Splunk, many servers may be involved. The central log server is called the *reporting host*. The system where the event occurred is called the *originating host* (or just the host). In this case you need to define rules that set the host field value based on the information in the events themselves.

For more information, see "Override default host values based on event data" in this manual.

**Set a default host for a file or directory input**

*Set a default host for a file or directory input*

In certain situations you may want to explicitly set a host value for all data coming in to Splunk through a particular file or directory input. You can set the host statically or dynamically.

- To **statically** set the host means you're setting the same host for every event that comes to Splunk through a designated file or directory input.
- If you **dynamically** set the host value, Splunk extracts the host name from a portion of the source input using a regex or segment of the source's full directory path.

You can also assign host values to events coming through a particular file or directory input based on their source or sourcetype values (as well as other kinds of information). For more information, see "Override default host values based on event data" in this manual.

**Note:** Splunk currently does not enable the setting of default host values for event data received through TCP, UDP, or **scripted inputs**.

**Statically setting the default host value for a file or directory input**

This method applies a single default host value to each event received through a specific file or directory input.

**Note:** A static host value assignment only impacts new data coming in through the input with which it's associated. You cannot assign a default host value to data that has already been processed, split into events, and indexed.

If you need to assign a host value to data that's already been indexed, you need to tag the host value instead.
You can statically define a host for a file or directory input whenever you add a new input of that type through the "Data inputs" page of Splunk Web's Manager interface:

1. In Splunk Web, click on the **Manager** link in the upper right-hand corner of the screen.

2. In Manager, click **Data inputs** under **System configurations**.

3. On the Data inputs page, select **Files & Directories** to go to the list page for that input type.

4. On the Files & directories page, you can either click the name of an input that you want to update, or click **New** to create a new file or directory input.

5. Once you're on the detail page for the file or directory input, select the **Constant value** option from the **Set host** dropdown.

6. Enter the static host value for the input in the **Host field value** field.

7. Save your changes.

For more information about inputs and input types, see "What Splunk can monitor" in this manual.

**Via configuration files**

Edit inputs.conf to specify a host value for a monitored file or directory input. Include a `host =` attribute within the appropriate stanza.

```
[monitor://<path>]
host = $YOUR_HOST
```

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

For more information about inputs and input types, see "What Splunk can monitor" in this manual.

**Example of static host value assignment for an input**

This example covers any events coming in from `/var/log/httpd`. Any events coming from this input will receive a **host value** of `webhead-1`.

```
[monitor:///var/log/httpd]
host = webhead-1
```

**Dynamically setting the default host value for a file or directory input**

Use this method if you want to dynamically extract the host value for a file or directory input, either from a segment of the source input path, or from a regular expression. For example, if you want to
index an archived directory and the name of each file in the directory contains relevant host information, you can use Splunk to extract this information and assign it to the host field.

**Via SplunkWeb**

Start by following the steps for setting up a static host assignment via Splunk Web, above. However, when you get to the **Set host** dropdown list on the input details page for a file or directory input, choose one of the following two values:

- **Regex on path** - Choose this option if you want to extract the host name via a regular expression. Enter the regex for the host you want to extract in the **Regular expression** field.
- **Segment in path** - Choose this option if you want to extract the host name from a segment in your data source's path. Enter the segment number in the **Segment #** field. For example, if the path to the source is `/var/log/[host server name]` and you want the third segment (the host server name) to be the host value, enter `3` into the **Segment #** field.

**Note:** For a primer on regular expression syntax and usage, see Regular-Expressions.info. You can test regexes by using them in searches with the rex search command. Splunk also maintains a list of useful third-party tools for writing and testing regular expressions.

**Via configuration files**

You can set up dynamic host extraction rules when you are configuring `inputs.conf`. Edit `inputs.conf` in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For more information on configuration files in general, see "About configuration files" in the Admin manual.

Add `host_regex = <regular expression>` to override the host field with a value extracted using a regular expression.

```plaintext
[monitor://<path>]
host_regex = $YOUR_REGEX
```

The regular expression extracts the host value from the filename of each input. The first capturing group of the regex is used as the host.

**Note:** If the regex fails to match, the default `host =` attribute is set as the host.

**Important:** For a primer on regular expression syntax and usage, see Regular-Expressions.info. You can test regexes by using them in searches with the rex search command. Splunk also maintains a list of useful third-party tools for writing and testing regular expressions.

Define a `host_segment = <integer>` instead of a `host_regex` if you want to override the host field with a value extracted using a segment of the data source path. For example, if the path to the source is `/var/log/[host server name]` and you want the third segment (the host server name) to be the host value, your input stanza would look like:

```plaintext
[monitor://var/log/]
```
host_segment = 3

**Note:** If the `<integer>` value is not an integer, or is less than 1, Splunk sets the default `host =` attribute as the host.

**Note:** You cannot simultaneously specify a `host_regex` and `host_segment`.

**Examples of dynamic host assignment for an input**

This example uses regex on the file path to set the host:

```
[monitor://var/log]
host_regex = /var/log/(\w+)
```

With that regex, all events from `/var/log/foo.log` are given the a `host` value of `foo`.

This example uses the segment of the data source filepath to set the host:

```
[monitor://apache/logs/]
host_segment = 3
```

It sets the `host` value to the third segment in the path `apache/logs`.

**Override default host values based on event data**

**Override default host values based on event data**

Splunk assigns default host names to your events based on data in those events. This topic shows you how to override specific default host assignments when these default assignments are incorrect.

**Configuration**

To set up host value overrides based on event data, you need to edit `transforms.conf` and `props.conf`. Edit these files in `$SPLUNK_HOME/etc/system/local/`, or your own custom application directory in `$SPLUNK_HOME/etc/apps/`. For more information about configuration files in general, see "About configuration files" in the Admin manual.

**Edits to transforms.conf**

Add your custom stanza to `$SPLUNK_HOME/etc/system/local/transforms.conf`. Configure your stanza as follows:

```
[$UNIQUE_STANZA_NAME]
DEST_KEY = MetaData:Host
REGEX = $YOUR_REGEX
FORMAT = host::$1
```

Fill in the stanza name and the regex fields with the correct values for your data.

Leave `DEST_KEY = MetaData:Host` to write a value to the `host::` field. `FORMAT = host::$1` writes the `REGEX` value into the `host::` field.
**Note:** Name your stanza with a unique identifier (so it is not confused with an existing stanza in $SPLUNK_HOME/etc/system/default/transforms.conf).

**Edits to props.conf**

Create a stanza in $SPLUNK_HOME/etc/system/local/props.conf to map the transforms.conf regex to the source type in props.conf.

```
[<spec>]
TRANSFORMS-$name=$UNIQUE_STANZA_NAME
```

- `<spec>` can be:
  - `<sourcetype>`, the sourcetype of an event
  - `host::<host>`, where `<host>` is the host for an event
  - `source::<source>`, where `<source>` is the source for an event
- `$name` is whatever unique identifier you want to give to your transform.
- `$UNIQUE_STANZA_NAME` must match the stanza name of the transform you just created in transforms.conf.

**Note:** Optionally add any other valid attribute/value pairs from props.conf when defining your stanza. This assigns the attributes to the `<spec>` you have set. For example, if you have custom line-breaking rules to set for the same `<spec>`, append those attributes to your stanza.

**Example**

Here is a set of events from the houseness.log file. They contain the host in the third position.

```
41602046:53 accepted fflanda
41602050:29 accepted rhallen
41602052:17 accepted fflanda
```

Create a regex to extract the host value and add it to a new stanza in $SPLUNK_HOME/etc/system/local/transforms.conf:

```
[houseness]
DEST_KEY = MetaData:Host
REGEX = \s(\w*)$
FORMAT = host::$1
```

Now, link your transforms.conf stanza to $SPLUNK_HOME/etc/system/local/props.conf so your transforms are called. Optionally add any additional attribute/value pairs from props.conf as needed.

The transform above works with the following stanza in props.conf:

```
[source::.../houseness.log]
TRANSFORMS-rhallen=houseness
SHOULD_LINEMERGE = false
```

The above stanza has the additional attribute/value pair `SHOULD_LINEMERGE = false`. This specifies that Splunk should create new events at a newline.
The events now appear in SplunkWeb as the following:

<table>
<thead>
<tr>
<th>Time</th>
<th>Host</th>
<th>Source Type</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>6/22/09</td>
<td>416082652:17 accepted fflanda</td>
<td>source</td>
<td>source_rhallen.log</td>
</tr>
<tr>
<td>6/22/09</td>
<td>416082658:29 accepted rhallen</td>
<td>source</td>
<td>source_rhallen.log</td>
</tr>
<tr>
<td>6/22/09</td>
<td>416082661:53 accepted fflanda</td>
<td>source</td>
<td>source_rhallen.log</td>
</tr>
</tbody>
</table>

**Handle incorrectly-assigned host values**

At some point, you may discover that the host value for some of your events might be set incorrectly for some reason. For example, you might be scraping some Web proxy logs into a directory directly on your Splunk server and add that directory as an input to Splunk without remembering to override the value of the host field, causing all those events to think their original host value is the same as your Splunk host.

If something like that happens, here are your options, in order of complexity:

- Delete and reindex the entire index
- Use a search to delete the specific events that have the incorrect host value and reindex those events
- Tag the incorrect host values with a tag, and search with that
- Set up a static field lookup to look up the host, map it in the lookup file to a new field name, and use the new name in searches
- Alias the host field to a new field (such as temp_host), set up a static field lookup to look up the correct host name using the name temp_host, then have the lookup overwrite the original host with the new lookup value (using the OUTPUT option when defining the lookup)

Of these options, the last option will look the nicest if you can't delete and reindex the data, but deleting and reindexing the data will always give the best performance.

**Override automatic source type assignment**

You can override automatic source type assignment for event data that comes from specific inputs, or which has a particular source.

*Note:* While source type assignment by input seems like a simple way to handle things, it isn't very granular—when you use it Splunk gives all event data from an input the same source type, even if they actually have different sources and hosts. If you want to bypass automatic source type assignment in a more targeted manner, arrange for Splunk to assign source types according to the event data source.
Override automated source type matching for an input

Use these instructions to override automated source type assignment and explicitly assign a single source type value to data coming from a specific input such as `/var/log/`.

**Note:** This only affects new data coming in *after* the override is set up. To correct the source types of events that have already been indexed, create a tag for the source type instead.

**Through Splunk Web**

When you define a data input in Manager, you can set a source type value that Splunk applies to all incoming data from that input. Manager gives you the option of picking a source type from a list or entering a unique source type value of your own.

To select a source type for an input, click the **Manager** link to go to the Splunk Manager page, select **Data inputs**, and then drill down to the details page of the input for which you want to define a source type. On the details page, check the **More settings** box, if it's not already selected.

Under the **Source type** heading, you'll see three dropdown choices for setting the source type:

- **Automatic.** With this default setting, Splunk automatically selects a source type for the data.

- **From list.** You can use a dropdown list to choose from a set of common pretrained source types.

- **Manual.** If the source type you want isn't in the dropdown list but still belongs to the set of Splunk's pretrained source types, you can enter its value manually. You can also manually enter your own source type. For information on Splunk's pretrained source types, see the "List of pretrained sourcetypes" topic in this manual.

**Pick a source type from a dropdown list**

On the details page for the input that you're choosing a source type for, select **From list** from the **Set the source type** dropdown list. Then choose a source type from the **Select source type from list** dropdown list that next appears.

Save your input settings. After that point, Splunk will assign the source type that you've selected to all events it indexes for that input.

**Note:** The dropdown list includes just the most common source types. For the complete list of available pretrained source types, see "List of pretrained sourcetypes".

**Manually enter a source type**

You can manually enter a source type for data that Splunk receives from a particular input. This can be either one of Splunk's pretrained source types or a source type of your own.

On the details page for the input that you're defining a source type for, select **Manual** from the **Set the source type** dropdown list. Then enter a source type in the **Source type** field that next appears.
Save your input settings. After that point, Splunk will assign the source type that you've specified to all events it indexes for that input.

Through configuration files

When you configure inputs in inputs.conf, you can set a source type as well. Edit inputs.conf in $SPLUNK_HOME/etc/system/local/, or your own custom application directory in $SPLUNK_HOME/etc/apps/. For more information on configuration files in general, see "About configuration files" in the Admin manual.

**Note:** This only affects new data coming in *after* your configuration change. If you want to correct the source types of events that have already been indexed, create a tag for the source type instead.

Include a **sourcetype=** attribute within the appropriate stanza in inputs.conf. For example:

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

This example sets any events coming from your TCP input on port 9995 as **sourcetype=log4j**.

**Override automatic source type matching for a source**

Use these instructions to override automated source type matching and explicitly assign a single source type value to data coming from a specific source.

Use these instructions to assign a source type based on a source through props.conf. Edit props.conf in $SPLUNK_HOME/etc/system/local/, or your own custom application directory in $SPLUNK_HOME/etc/apps/. For more information on configuration files in general, see "About configuration files".

**Important:** If you are forwarding data to one or more receivers, and want to set up an override of automatic source type matching for a specific source, you must set it up on the props.conf file for the forwarder. If you set it up on the receiver, the override will not take effect.

**Note:** This only impacts new data coming in *after* your configuration change. If you want to correct the source types of events that have already been indexed, create a tag for the source type instead.

Learn more about props.conf.

Through configuration files

Add a stanza for your source in props.conf. In the stanza, identify the source path, using regex syntax for flexibility if necessary. Then identify the source type by including a **sourcetype=** attribute. For example:

```plaintext
[source::..../var/log/anaconda.log(\d+)?]
sourcetype=anaconda
```
This example sets any events from sources containing the string /var/log/anaconda.log followed by any number of numeric characters to sourcetype=anaconda.

Splunk recommends that your stanza source path regexes (such as [source::.../web/....log]) be as specific as possible. It is HIGHLY recommended that you not have the regex end in "...". For example, don’t do this:

[source::/home/fflanda/...] sourcetype=mytype

This is dangerous. The above example tells Splunk to process gzip files in /home/fflanda as mytype files rather than gzip files.

It would be much better to write:

[source::/home/fflanda/...log(\d+)?$] sourcetype=mytype

Note: For a primer on regular expression syntax and usage, see Regular-Expressions.info. You can test regexes by using them in searches with the rex search command. Splunk also maintains a list of useful third-party tools for writing and testing regular expressions.

Advanced source type overrides

Advanced source type overrides

This topic shows you how to configure Splunk to override sourcetypes on a per-event basis. It includes an example that demonstrates the use of transforms.conf in tandem with props.conf to override sourcetypes for events associated with a specific host, and goes on to show how you can do this for event data coming from a particular input or source.

For more information about performing basic source type overrides for event data that comes from specific inputs, or which has a particular source, see "Override automatic source type assignment" in this manual.

Configuration

To do this you’ll set up two stanzas, one in transforms.conf, and another in props.conf. Edit these files in $SPLUNK_HOME/etc/system/local/, or your own custom application directory in $SPLUNK_HOME/etc/apps/.

transforms.conf

The transforms.conf stanza should follow this format:

[<unique_stanza_name>]
REGE = <your_regex>
FORMAT = sourcetype::<your_custom_sourcetype_value>
DEST_KEY = MetaData:Sourcetype
• `<unique_stanza_name>` should reflect that it involves a sourcetype. You'll use this name later in the `props.conf` stanza.
• `<your_regex>` is a regular expression that identifies the events that you want to apply a custom sourcetype to (such as events carrying a particular hostname or other field value).
• `<your_custom_sourcetype_value>` is the sourcetype value that you want to apply to the regex-selected events.

**Note:** For a primer on regular expression syntax and usage, see Regular-Expressions.info. You can test regexes by using them in searches with the rex search command. Splunk also maintains a list of useful third-party tools for writing and testing regular expressions.

**props.conf**

Next you create a stanza in `props.conf` that references the `transforms.conf` stanza, as follows.

```
[<spec>]
TRANSFORMS-<class> = <unique_stanza_name>
```

• `<spec>` can be:
  ♦ `<sourcetype>`, the sourcetype value of an event.
  ♦ `host::<host>`, where `<host>` is the host value for an event.
  ♦ `source::<source>`, where `<source>` is the source value for an event.
• `<class>` is any name that you want to give to your stanza to identify it. In this case you might just use "sourcetype" to identify it as a sourcetype.
• `<unique_stanza_name>` is the name of your stanza from `transforms.conf`.

**Example - Sourcetyping events originating from different hosts, indexed from a single input**

Let's say that you have a shared UDP input, UDP514. Your Splunk instance indexes a wide range of data from a number of hosts through this input. You've found that you need to apply a particular sourcetype--which, for the purposes of this example we'll call "my_log"--to data originating from three specific hosts (host1, host2, and host3) that reaches Splunk through UDP514.

To start, you can use the regex that Splunk typically uses to extract the host field for syslog events. You can find it in `system/default/transforms.conf`:

```
[syslog-host]
REGEX = :\d\d\s+(?:\d+\s+|(?:user|daemon|local\.?).\w+\s+)*\[?\(\w[\w\.-]{2,}\]\]?$\s
FORMAT = host::$1
DEST_KEY = MetaData:Host
```

You can easily modify this regex to only match events from the hostnames you want (for the purposes of this example we're calling them `host1`, `host2`, and `host3`):

```
REGEX
= :\d\d\s+(?:\d+\s+|(?:user|daemon|local\.\d+\w+\s+)*\[?\(host1|host2|host3\)\]\w\s+)\s+
```

Now you can use that modified regex in a transform that applies the `my_log` sourcetype to events that come from those three hosts:

```
[set_sourcetype_my_log_for_some_hosts]
```
Configure rule-based source type recognition

Configure rule-based source type recognition to expand the range of source types that Splunk recognizes. Splunk automatically assigns rule-based source types based on regular expressions you specify in props.conf.

You can create two kinds of rules in props.conf: rules and delayed rules. The only difference between the two is the point at which Splunk checks them during the source typing process. As it processes each string of event data, Splunk uses several methods to determine source types:

- After checking for explicit source type definitions based on the event data input or source, Splunk looks at the rule:: stanzas defined in props.conf and tries to match source types to the event data based on the classification rules specified in those stanzas.
- If Splunk is unable to find a matching source type using the available rule:: stanzas, it tries to use automatic source type matching, where it tries to identify patterns similar to source types it has learned in the past.
- When that method fails, Splunk then checks the delayedrule:: stanzas in props.conf, and tries to match the event data to source types using the rules in those stanzas.

You can set up your system so that rule:: stanzas contain classification rules for specialized source types, while delayedrule:: stanzas contain classification rules for generic source types. This way the the generic source types are applied to broad ranges of events that haven't qualified for more specialized source types. For example, you could use rule:: stanzas to catch event data with specific syslog source types, such as sendmail_syslog or cisco_syslog and then have a delayedrule:: stanza apply the generic syslog source type to remaining syslog event data.

Configuration

To set source typing rules, edit props.conf in $SPLUNK_HOME/etc/system/local/, or your own custom application directory in $SPLUNK_HOME/etc/apps/. For more information on configuration files in general, see "About configuration files" in 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 name in the body of the stanza. After the source type declaration, list the the source type assignation rules. These rules use one or more `MORE_THAN` and `LESS_THAN` statements to find patterns in the event data that fit given regular expressions by specific percentages.

**Note:** You can specify any number of `MORE_THAN` and `LESS_THAN` statements in a source typing rule stanza. All of the statements must match a percentage of event data lines before those lines can be assigned the source type in question. For example, you could define a rule that assigns a specific source type value to event data where more than 10% match one regular expression and less than 10% match another regular expression.

Add the following to `props.conf`:

```
[rule::$RULE_NAME] OR [delayedrule::$RULE_NAME]
sourcetype=$SOURCETYPE
MORE_THAN_[0-100] = $REGEX
LESS_THAN_[0-100] = $REGEX
```

The `MORE_THAN` and `LESS_THAN` numerical values refer the percentage of lines that contain the string specified by the regular expression. To match, a rule can be either `MORE_THAN` or `LESS_THAN` those percentages.

**Note:** For a primer on regular expression syntax and usage, see Regular-Expressions.info. You can test regexes by using them in searches with the rex search command. Splunk also maintains a list of useful third-party tools for writing and testing regular expressions.

Examples

The following examples come from `$SPLUNK_HOME/etc/system/default/props.conf`.

**Postfix syslog files**

```
# postfix_syslog sourcetype rule
[rule::postfix_syslog]
sourcetype = postfix_syslog
# If 80% of lines match this regex, then it must be this type
MORE_THAN_80 = ^\w{3} +\d+ \d\d:\d\d:\d\d.* postfix(/\w+)?/\[\d+\]:
```

**Delayed rule for breakable text**

```
# breaks text on ascii art and blank lines if more than 10% of lines have
# ascii art or blank lines, and less than 10% have timestamps
[delayedrule::breakable_text]
sourcetype = breakable_text
MORE_THAN_10 = (\^[\-\+---]+|\*\*\*|\*=\*|\*=\*\*|\+=\+=)|^s*$
LESS_THAN_10 = [: ]\[012]?[0-9]:[0-5][0-9]
```

**Rename source types**
Rename source types

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.

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 a search time.

To rename the source type, add the following to your source type stanza:

```
rename = <string>
```

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.

Train Splunk's source type autoclassifier

**Train Splunk's source type autoclassifier**

Use these instructions to train Splunk to recognize a new source type, or give it new samples to better recognize a pre-trained source type. Autoclassification training enables Splunk to classify future event data with similar patterns as a specific source type. This can be useful when Splunk is indexing directories that contains data with a mix of source types (such as `/var/log`). Splunk ships "pre-trained," with the ability to assign `sourcetype=syslog` to most syslog files.

**Note:** Keep in mind that source type autoclassification training applies to future event data, not event data that has already been indexed.
You can also bypass auto-classification in favor of hardcoded configurations, and just override a sourcetype for an input, or override a sourcetype for a source. Or configure rule-based source type recognition.

You can also anonymize your file using Splunk's built in anonymizer utility.

If Splunk fails to recognize a common format, or applies an incorrect source type value, you should report the problem to Splunk support and send us a sample file.

via the CLI

Here's what you enter to train source types through the CLI:

```
# splunk train sourcetype $FILE_NAME $SOURCETYPE_NAME
```

Fill in $FILE_NAME with the entire path to your file. $SOURCETYPE_NAME is the custom source type you wish to create.

It's usually a good idea to train on a few different samples for any new source type so that Splunk learns how varied a source type can be.

**List of pretrained source types**

List of pretrained source types

Splunk ships pre-trained to recognize many different source types. A number of source types are automatically recognized, tagged and parsed appropriately. Splunk also contains a significant number of pre-trained source types that are not automatically recognized but can be assigned via Splunk Web or inputs.conf.

It's a good idea to use a pre-trained source type if it matches your data, as Splunk contains optimized indexing properties for pre-trained source types. However, if your data does not fit with any pre-trained source types, Splunk can index virtually any format of data without custom properties.

Learn more about source types and how they work.

**Automatically recognized source types**

<table>
<thead>
<tr>
<th>Source type name</th>
<th>Origin</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>access_combined</strong></td>
<td>NCSA combined format http web server logs (can be generated by apache or other web servers)</td>
<td>10.1.1.43 - webdev [08/Aug/2005:13:18:16 &quot;-&quot; &quot;check_http/1.10 (nagios-plugins 1.4]]</td>
</tr>
<tr>
<td><strong>access_combined_wcookie</strong></td>
<td>NCSA combined</td>
<td>&quot;66.249.66.102.1124471045570513&quot; 59.92.110.121</td>
</tr>
<tr>
<td>log_type</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>access_common</td>
<td>NCSA common format http web server logs (can be generated by apache or other web servers), with cookie field added at end</td>
<td>10.1.1.140 - - [16/May/2005:15:01:52 -0700] &quot;GET /themes/ComBeta/images/bullet.png HTTP/1.1&quot;</td>
</tr>
<tr>
<td>apache_error</td>
<td>Standard Apache web server error log</td>
<td>[Sun Aug 7 12:17:35 2005] [error] [client /home/reba/public_html/images/bullet_image.png]</td>
</tr>
<tr>
<td>asterisk_cdr</td>
<td>Standard Asterisk IP PBX call detail record</td>
<td>Aug 24 14:08:05 asterisk[14287]: Manager 'randy' logged on from 127.0.0.1</td>
</tr>
<tr>
<td>asterisk_event</td>
<td>Standard Asterisk event log (management events)</td>
<td>Aug 24 14:48:27 WARNING[14287]: Channel extension 's' in context 'default', but</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 extension 's' in context 'default', but</td>
</tr>
<tr>
<td>asterisk_queue</td>
<td>Standard Asterisk queue log</td>
<td>NONE</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.com Aug 24 2005 00:08:49: %PIX-2-106001: Inbound TCP connection denied from IP_addr/port to 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 f</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 I27231H324760 PROC : db2fmp.exe INSTANCE: DB2 NOD Table Maintenance, db2HmonEvalStats, probe:900 STOP : 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.</td>
</tr>
</tbody>
</table>

-0700] "GET /themes/splunk_com/images/logo_splunk.png HTTP/1.1" "http://www.splunk.org/index.php/docs" "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.7.8) Gecko/20050524 Fedora/1.0.4-4 Firefox/1.0.4" "61.3.110.148.1124404439914689"
| **linux_messages_syslog** | Standard linux syslog (/var/log/messages on most platforms) | Aug 19 10:04:28 db1 sshd(pam_unix)[15979] (uid=0) |
| **linux_secure** | Linux securelog | Aug 18 16:19:27 db1 sshd[29330]: Accepted publickey for root from ::ffff:10.2.1.5 port 40892 ssh2 |
| **log4j** | Log4j standard output produced by any J2EE server using log4j | 2005-03-07 16:44:03,110 53223013 [PoolThread-0] INFO [STDOUT] got some property... |
| **mysql_error** | Standard mysql error log | 050818 16:19:29 InnoDB: Started; log seq '/usr/libexec/mysqld: ready for connections, '/var/lib/mysql/mysql.sock' port: 3306 S |
| **mysqlqd** | Standard mysql query log; also matches mysql's binary log following conversion to text | 53 Query SELECT xar_dd_itemid, xar_dd_propid, xar_dd_value FROM xar_dynamic_data WHERE xar_dd_propid IN |
| **postfix_syslog** | Standard Postfix MTA log reported via the Unix/Linux syslog facility | Mar 1 00:01:43 avas postfix/smtpd[1822]: client=host76-117.pool80180.interbusiness |
| **sendmail_syslog** | Standard Sendmail MTA log reported via the Unix/Linux syslog facility | Aug 6 04:03:32 nmrij00 sendmail[5200]: q _ctladdr=root (0/0), delay=00:00:01, xdelay=00026, relay=[101.0.0.1] [101.0.0.1] (v00F3HmX004301 Message accepted for delivery |
| **sugarcrm_log4php** | Standard Sugarcrm activity log reported using the log4php utility | 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) |
| **weblogic_stdout** | Weblogic server log in the standard native BEA format | ######<Sep 26, 2005 7:27:24 PM MDT> <Warning asiAdminServer> <ListenThread.Default> <HostName: 0.0.0.0, maps to multiple IP addresses:169.254.25.129,169.254.193.219> |
| **websphere_activity** | Websphere activity log, also often referred to as the service log | --------------------------------------------------------------- ComponentId: Application Server ProcessId: 2580 ThreadName: Non-deferrable Alarm : 3 Source com.ibm.ws.channel.framework.impl. WSChannelFrameworkImpl, MethodName: Manufacturer: IBM Product: WebSphere 6.0.1.0 o0510.18] ServerName: nd6Cell01\was1Node01\TradeServer1 TimeStamp: 2005-07-01 13:04:55.187000000 UnitOfWork: 156 |
PrimaryMessage: CHFW0020I: The Transport Channel service has stopped the Chain labeled SOAPAcceptorChain2

---

<table>
<thead>
<tr>
<th>Source type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>websphere_core</td>
<td>Corefile export from Websphere</td>
</tr>
<tr>
<td>websphere_trlog_syserr</td>
<td>Standard Websphere system error log in IBM's native tr log format</td>
</tr>
<tr>
<td>websphere_trlog_sysout</td>
<td>Standard Websphere system out log in IBM's native tr log format; similar to the log4j server log for Resin and Jboss, same format as the system error log but containing lower severity and informational events</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>
</tr>
</tbody>
</table>

---

Pre-trained source types

This list contains both automatically recognized source types and pre-trained source types that are not automatically recognized.

<table>
<thead>
<tr>
<th>Category</th>
<th>Source type(s)</th>
</tr>
</thead>
</table>

157
### Application servers
- log4j, log4php, weblogic_stdout, websphere_activity, websphere_core, websphere_trlog

### Databases
- mysqld, mysqld_error, mysqld_bin

### E-mail
- exim_main, exim_reject, postfix_syslog, sendmail_syslog, procmail

### Operating systems
- linux_messages_syslog, linux_secure, linux_audit, linux_bootlog, anaconda, anaconda_syslog, osx_asl, osx_crashreporter, osx_crash_log, osx_install, osx_secure, osx_daily, osx_weekly, osx_monthly, osx_window_server, windows_snare_syslog, dmesg, ftp, ssl_error, syslog, sar, rpmpkgs

### Network
- novell_groupwise, tcp

### Printers
- cups_access, cups_error, spooler

### Routers and firewalls
- cisco_cdr, cisco_syslog, clavister

### VoIP
- asterisk_cdr, asterisk_event, asterisk_messages, asterisk_queue

### Webservers
- access_combined, access_combined_wcookie, access_common, apache_error, iis

### Miscellaneous
- snort

---

**Finding out how a pre-trained source type is configured to work**

To find out what configuration information Splunk is using to index a given source type, you can use the `btool` utility to list out the properties. For more information on using `btool`, refer to "Command line tools for use with Support's direction" in the Admin manual.

The following example shows how to list out the configuration for the `tcp` source type:

```bash
$ 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
```
Specify source type settings in props.conf

There are source type specific settings in props.conf. Specify settings for a source type using the following attribute/value pairs. Add a sourcetype stanza to props.conf in $SPLUNK_HOME/etc/system/local/, or your own custom application directory in $SPLUNK_HOME/etc/apps/. For more information on configuration files, see "About configuration files" in the Admin manual.

Note: The following attribute/value pairs can only be set for a stanza that begins with [<$SOURCETYPE>]:

invalid_cause = <string>

- Can only be set for a [<sourcetype>] stanza.
- Splunk will not index any data with invalid_cause set.
- Set <string> to "archive" to send the file to the archive processor (specified in unarchive_cmd).
- Set to any other string to throw an error in the splunkd.log if running Splunklogger in debug mode.
- Defaults to empty.

unarchive_cmd = <string>

- Only called if invalid_cause is set to "archive".
- <string> specifies the shell command to run to extract an archived source.
- Must be a shell command that takes input on stdin and produces output on stdout.
- Defaults to empty.

LEARN_MODEL = <true/false>

- For known sourcetypes, the fileclassifier will add a model file to the learned directory.
- To disable this behavior for diverse sourcetypes (such as sourcecode, where there is no good exemplar to make a sourcetype) set LEARN_MODEL = false.
- More specifically, set LEARN_MODEL to false if you can easily classify your source by its name or a rule and there's nothing gained from trying to analyze the content.
- Defaults to empty.

maxDist = <integer>

- Determines how different a sourcetype model may be from the current file.
- The larger the value, the more forgiving.
- For example, if the value is very small (e.g., 10), then files of the specified sourcetype should not vary much.
- A larger value indicates that files of the given sourcetype vary quite a bit.
Configure index-time field extractions

Caution: We do not recommend that you add custom fields to the set of default fields that Splunk automatically extracts and indexes at index time, such as timestamp, punct, host, source, and sourcetype. Adding to this list of fields can negatively impact indexing performance and search times, because each indexed field increases the size of the searchable index. Indexed fields are also less flexible—whenever you make changes to your set of fields, you must re-index your entire dataset. For more information, see "Index time versus search time" in the Admin manual.

With those caveats, there are times when you may find a need to change or add to your indexed fields. For example, you may have situations where certain search-time field extractions are noticeably impacting search performance. This can happen, for example, if you commonly search a large event set with expressions like foo!=bar or NOT foo=bar, and the field foo nearly always takes on the value bar.

Conversely, you may want to add an indexed field if the value of a search-time extracted field exists outside of the field more often than not. For example, if you commonly search only for foo=1, but 1 occurs in many events that do not have foo=1, you may want to add foo to the list of fields extracted by Splunk at index time.

In general, you should try to extract your fields at search time. For more information see "Create search-time field extractions" in the Knowledge Manager manual.

Define additional indexed fields

Define additional indexed fields by editing props.conf, transforms.conf and fields.conf.

Add a regex stanza for the new field to transforms.conf

Follow this format when you define an index-time field transform in transforms.conf (Note: Some of these attributes, such as LOOKAHEAD and DEST_KEY, are only required for certain use cases):

```
[<unique_transform_stanza_name>]
REGEX = <regular_expression>
FORMAT = <your_custom_field_name>::$1
```
• The `<unique_stanza_name>` is required for all transforms, as is the REGEX.

• REGEX is a regular expression that operates on your data to extract fields.
  ♦ Name-capturing groups in the REGEX are extracted directly to fields, which means that you don’t have to specify a FORMAT for simple field extraction cases.
  ♦ If the REGEX extracts both the field name and its corresponding value, you can use the following special capturing groups to skip specifying the mapping in the FORMAT attribute:

    _KEY_<string>, _VAL_<string>

• For example, the following are equivalent:

Using FORMAT:

REGEX = ([a-z]+)=(^[a-z]+)
FORMAT = $1::$2

Not using FORMAT:

REGEX = (?<KEY_1>[a-z]+)=(?<VAL_1>[a-z]+)

• FORMAT is optional. Use it to specify the format of the field/value pair(s) that you are extracting, including any field names or values that you want to add. You don’t need to specify the FORMAT if you have a simple REGEX with name-capturing groups.
• 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")
    or

    FORMAT = $1::$2 (where the REGEX extracts both the field name and the field value)
However you can also set up index-time field extractions that create concatenated fields:

```
FORMAT = ipaddress::$1.$2.$3.$4
```

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 foo$bar
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 256. You may want to increase your **LOOKAHEAD** value if you have events with line lengths longer than 256 characters.

**Note:** For a primer on regular expression syntax and usage, see Regular-Expressions.info. You can test regexes by using them in searches with the rex search command. Splunk also maintains a list of useful third-party tools for writing and testing regular expressions.

**Note:** The capturing groups in your regex must identify field names that use ASCII characters (a-zA-Z0-9_-.). International characters will not work.

**Link the new field to props.conf**

To **props.conf**, add the following lines:

```
[<spec>]
TRANSFORMS=<value> = <unique_stanza_name>
```

- `<spec>` can be:
  - `<sourcetype>`, the sourcetype of an event.
  - `host::<host>`, where `<host>` is the host for an event.
  - `source::<source>`, where `<source>` is the source for an event.
  - **Note:** You can use regex-type syntax when setting the `<spec>`. Also, source and sourcetype stanzas match in a case-sensitive manner while host stanzas do not. For more information, see the props.conf spec file.
- `<value>` is any value you want, to give your attribute its name-space.
- `<unique_stanza_name>` is the name of your stanza from transforms.conf.

**Note:** For index-time field extraction, **props.conf** uses TRANSFORMS-<class>, as opposed to EXTRACT-<value>, which is used for configuring search-time field extraction.

**Add an entry to fields.conf for the new field**

Add an entry to **fields.conf** for the new indexed field:

```
[[your_custom_field_name]]
INDEXED=true
```

- `<your_custom_field_name>` is the name of the custom field you set in the unique stanza that you added to transforms.conf.
- Set INDEXED=true to indicate that the field is indexed.

**Note:** If a field of the same name is extracted at search time, you **must** set INDEXED=false for the field. In addition, you **must also** set INDEXED_VALUE=false if events exist that have values of that field that are not pulled out at index time, but which are extracted at search time.

For example, say you’re performing a simple `<field>::1234` extraction at index time. This could
work, but you would have problems if you also implement a search-time field extraction based on a
regex like $A(\d+)B$, where the string $A1234B$ yields a value for that field of 1234. This would turn up
events for 1234 at search time that Splunk would be unable to locate at index time with the
$<field>::1234$ extraction.

Restart Splunk for your changes to take effect

Changes to configuration files such as $props.conf$ and $transforms.conf$ won’t take effect until
you shut down and restart Splunk.

How Splunk builds indexed fields

Splunk builds indexed fields by writing to _meta. Here’s how it works:

- _meta is modified by all matching transforms in $transforms.conf$ that contain either $DEST_KEY$
  = _meta or $WRITE_META = true$.
- Each matching transform can overwrite _meta, so use $WRITE_META = true$ to append
  _meta.
  - If you don't use $WRITE_META$, then start your $FORMAT$ with $\$$0\$.  
- After _meta is fully built during parsing, Splunk interprets the text in the following way:
  - The text is broken into units; each unit is separated by whitespace.
  - Quotation marks (" ") group characters into larger units, regardless of whitespace.
  - Backslashes (\) immediately preceding quotation marks disable the grouping
    properties of quotation marks.
  - Backslashes preceding a backslash disable that backslash.
  - Units of text that contain a double colon (:) are turned into extracted fields. The text on
    the left side of the double colon becomes the field name, and the right side becomes
    the value.

Note: Indexed fields with regex-extracted values containing quotation marks will generally not work,
and backslashes may also have problems. Fields extracted at search time do not have these
limitations.

Here’s an example of a set of index-time extractions involving quotation marks and backslashes

to disable quotation marks and backslashes.

```plaintext
WRITE_META = true
FORMAT = field1::value field2::"value 2" field3::"a field with a \" quotation mark" field4::"a ends with a backslash"
```

When Splunk creates field names

When Splunk creates field names, it applies the following rules to all extracted fields, whether they
are extracted at index-time or search-time, by default or through a custom configuration:

1. All characters that are not in a-z,A-Z, and 0-9 ranges are replaced with an underscore (_).
2. All leading underscores are removed. In Splunk, leading underscores are reserved for internal
variables.
Index-time field extraction examples

Here are a set of examples of configuration file setups for index-time field extractions.

Define a new indexed field

This basic example creates an indexed field called `err_code`.

**transforms.conf**

In `transforms.conf` add:

```
[netscreen-error]
REGEX = device_id=\[\w+\](?<err_code>[^:\]+)
FORMAT = err_code::"$1"
WRITE_META = true
```

This stanza takes `device_id=` followed with a word within brackets and a text string terminating with a colon. The source type of the events is `testlog`.

Comments:

- The `FORMAT =` line contains the following values:
  - `err_code::` is the name of the field.
  - `$1` refers to the new field written to the index. It is the value extracted by `REGEX`.
- `WRITE_META = true` is an instruction to write the content of `FORMAT` to the index.

**props.conf**

Add the following lines to `props.conf`:

```
[testlog]
TRANSFORMS-netscreen = netscreen-error
```

**fields.conf**

Add the following lines to `fields.conf`:

```
[err_code]
INDEXED=true
```

Restart Splunk for your configuration file changes to take effect.

Define two new indexed fields with one regex

This example creates two indexed fields called `username` and `login_result`.

**transforms.conf**

In `transforms.conf` add:

```
[ftpd-login]
REGEX = Attempt to login by user: (.*)
```

165
This stanza finds the literal text \texttt{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:

\begin{verbatim}
2008-10-30 14:15:21 mightyhost awesomeftpd INFO Attempt to login by user: root: login FAILED.
\end{verbatim}

\begin{verbatim}
props.conf
Add the following lines to props.conf:

[ftpd-log]
TRANSFORMS-login = ftpd-login
\end{verbatim}

\begin{verbatim}
fields.conf
Add the following lines to fields.conf:

[username]
INDEXED=true

[login_result]
INDEXED=true
\end{verbatim}

Restart Splunk for your configuration file changes to take effect.

\textbf{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 \texttt{FORMAT} option.

Let's say you have the following event:

\begin{verbatim}
20100126 08:48:49 781 PACKET 078FCFD0 UDP Rcv 127.0.0.0 8226 R Q [0084 A NOERROR] A (4)www(8)google(3)com(0)
\end{verbatim}

Now, what you want to do is get \texttt{(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 \texttt{www.google.com}. How do you achieve this?

\begin{verbatim}
transforms.conf
You would start by setting up a transform in transforms.conf named \texttt{dnsRequest}:

[dnsRequest]
REGEX = UDP[^\(]+\(\d\)+\(\d\)+\(\w\)+\(\d\)+\(\d\)+\(\w\)+
FORMAT = dns_requestor::$1.$2.$3
\end{verbatim}

This transform defines a custom field named **dns_requestor**. It uses its \texttt{REGEX} to pull out the three segments of the **dns_requestor** value. Then it uses \texttt{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 file headers at index time**

Extract fields from file headers at index time

Certain data sources and source types, such as CSV and MS Exchange log files, can have headers that contain field information. You can configure Splunk to automatically extract these fields during index-time event processing.

For example, a legacy CSV file—which is essentially a static table—could have a header row like

```
name, location, message, "start date"
```

which behaves like a series of column headers for the values listed afterwards in the file.

**Note:** Automatic header-based field extraction doesn't impact index size or indexing performance because it occurs during source typing (before index time).

**How automatic header-based field extraction works**

When you enable automatic header-based field extraction for a specific source or source type, Splunk scans it for header field information, which it then uses for field extraction. If a source has the necessary header information, Splunk extracts fields using delimiter-based key/value extraction.

Splunk does this at index time by changing the source type of the incoming data to `[original_sourcetype]`-N, where N is a number). Next, it creates a stanza for this new source
type in props.conf, defines a delimiter-based extraction rule for the static table header in transforms.conf, and then ties that extraction rule to the new source type back in its new props.conf stanza. Finally, at search time, Splunk applies field transform to events from the source (the static table file).

You can use fields extracted by Splunk for filtering and reporting just like any other field by selecting them from the fields sidebar in the Search view (select Pick fields to see a complete list of available fields).

Note: Splunk records the header line of a static table in a CSV or similar file as an event. To perform a search that gets a count of the events in the file without including the header event, you can run a search that identifies the file as the source while explicitly excluding the comma delimited list of header names that appears in the event. Here’s an example:

source=/my/file.csv NOT "header_field1,header_field2,header_field3,..." | stats count

Enable automatic header-based field extraction

Enable automatic header-based field extraction for any source or source type by editing props.conf. Edit this file in $SPLUNK_HOME/etc/system/local/, or your own custom application directory in $SPLUNK_HOME/etc/apps/<app_name>/local.

Note: If you are using Splunk in a distributed environment, be sure to place the props.conf and transforms.conf files that you update for header-based field extraction on your search head, not the indexer.

For more information on configuration files in general, see "About configuration files" in the Admin manual.

To turn on automatic header-based field extraction for a source or source type, add CHECK_FOR_HEADER=TRUE under that source or source type’s stanza in props.conf.

Example props.conf entry for an MS Exchange source:

```
[MSExchange]
CHECK_FOR_HEADER=TRUE
...
```

OR

```
[source::C:\\Program Files\\Exchsrvr\\ServerName.log]
sourcetype=MSExchange

[MSExchange]
CHECK_FOR_HEADER=TRUE
```

Set CHECK_FOR_HEADER=FALSE to turn off automatic header-based field extraction for a source or source type.
Important: Changes you make to props.conf (such as enabling automatic header-based field extraction) won’t take effect until you restart Splunk.

Note: CHECK_FOR_HEADER must be in a source or source type stanza.

Changes Splunk makes to configuration files

If you enable automatic header-based field extraction for a source or source type, Splunk adds stanzas to copies of transforms.conf and props.conf in $SPLUNK_HOME/etc/apps/learned/local/ when it extracts fields for that source or source type.

Important: Don’t edit these stanzas after Splunk adds them, or the related extracted fields won’t work.

transforms.conf

Splunk creates a stanza in transforms.conf for each source type with unique header information matching a source type defined in props.conf. Splunk names each stanza it creates as [AutoHeader-N], where N is an integer that increments sequentially for each source that has a unique header ([AutoHeader-1], [AutoHeader-2],..., [AutoHeader-N]). Splunk populates each stanza with transforms that the fields (using header information).

Here’s the transforms.conf entry that Splunk would add for the MS Exchange source. It was enabled for automatic header-based field extraction in the preceding example:

```plaintext
...[AutoHeader-1]
FIELDS="time", "client-ip", "cs-method", "sc-status"
DELIMS=" "
...
```

props.conf

Splunk then adds new sourcetype stanzas to props.conf for each source with a unique name, fieldset, and delimiter. Splunk names the stanzas as [yoursource-N], where yoursoure is the source type configured with automatic header-based field extraction, and N is an integer that increments sequentially for each transform in transforms.conf.

For example, say you’re indexing a number of CSV files. If each of those files has the same set of header fields and with the same delimiter in transforms.conf, Splunk maps the events indexed from those files to a source type of csv-1 in props.conf. But if that batch of CSV files also includes a couple of files with unique sets of fields and delimiters, Splunk gives the events it indexes from those files source types of csv-2 and csv-3, respectively. Events from files with the same source, fieldset, and delimiter in transforms.conf will have the same source type value.

Note: If you want to enable automatic header-based field extraction for a particular source, and you have already manually specified a source type value for that source (either by defining the source type in Splunk Web or by directly adding the source type to a stanza in inputs.conf), be aware that setting CHECK_FOR_HEADER=TRUE for that source allows Splunk to override the source type value you’ve set for it with the source types generated by the automatic header-based field extraction.
process. This means that even though you may have set things up in `inputs.conf` so that all csv files get a source type of csv, once you set `CHECK_FOR_HEADER=TRUE`, Splunk overrides that source type setting with the incremental source types described above.

Here's the source type that Splunk would add to `props.conf` to tie the transform to the MS Exchange source mentioned earlier:

```
[MSExchange-1]
TRANSFORMS-AutoHeader = AutoHeader-1
...
```

**Note about search and header-based field extraction**

Use a wildcard to search for events associated with source types that Splunk generated during header-based field extraction.

For example, a search for `sourcetype="yoursource"` looks like this:

```
sourcetype=yoursource*
```

**Examples of header-based field extraction**

These examples show how header-based field extraction works with common source types.

**MS Exchange source file**

This example shows how Splunk extracts fields from an MS Exchange file using automatic header-based field extraction.

This sample MS Exchange log file has a header containing a list of field names, delimited by spaces:

```
# Message Tracking Log File
# Exchange System Attendant Version 6.5.7638.1
# Fields: time client-ip cs-method sc-status
14:13:11 10.1.1.9 HELO 250
14:13:13 10.1.1.9 MAIL 250
14:13:19 10.1.1.9 RCPT 250
14:13:29 10.1.1.9 DATA 250
14:13:31 10.1.1.9 QUIT 240
```

Splunk creates a header and transform in `transforms.conf`:

```
[AutoHeader-1]
FIELDS="time", "client-ip", "cs-method", "sc-status"
DELIMS=" "
```

Note that Splunk automatically detects that the delimiter is a whitespace.

Splunk then ties the transform to the source by adding this to the source type stanza in `props.conf`:  
170
Splunk automatically extracts the following fields from each event:

14:13:11 10.1.1.9 HELO 250

- time="14:13:11" client-ip="10.1.1.9" cs-method="HELO" sc-status="250"

14:13:13 10.1.1.9 MAIL 250

- time="14:13:13" client-ip="10.1.1.9" cs-method="MAIL" sc-status="250"

14:13:19 10.1.1.9 RCPT 250

- time="14:13:19" client-ip="10.1.1.9" cs-method="RCPT" sc-status="250"

14:13:29 10.1.1.9 DATA 250

- time="14:13:29" client-ip="10.1.1.9" cs-method="DATA" sc-status="250"

14:13:31 10.1.1.9 QUIT 240

- time="14:13:31" client-ip="10.1.1.9" cs-method="QUIT" sc-status="240"

CSV file

This example shows how Splunk extracts fields from a CSV file using automatic header-based field extraction.

Example CSV file contents:

foo,bar,anotherfoo,anotherbar
100,21,this is a long file,nomore
200,22,wow,o rly?
300,12,ya rly!,no wai!

Splunk creates a header and transform in `transforms.conf` (located in: `$SPLUNK_HOME/etc/apps/learned/transforms.conf`):

```plaintext
# Some previous automatic header-based field extraction
[AutoHeader-1]
...
# source type stanza that Splunk creates
[AutoHeader-2]
FIELDS="foo", "bar", "anotherfoo", "anotherbar"
DELIMS="","
```
Note that Splunk automatically detects that the delim is a comma.

Splunk then ties the transform to the source by adding this to a new source type stanza in `props.conf`:

```plaintext
[CSV-1]
REPORT-AutoHeader = AutoHeader-2
...
```

Splunk extracts the following fields from each event:

100,21,this is a long file,nomore

- `foo="100"` bar="21" anotherfoo="this is a long file"
  anotherbar="nomore"

200,22,wow,o rly?

- `foo="200"` bar="22" anotherfoo="wow" anotherbar="o rly?"

300,12,ya rly!,no wai!

- `foo="300"` bar="12" anotherfoo="ya rly!" anotherbar="no wai!"

**Answers**

Have questions? Visit Splunk Answers and see what questions and answers the Splunk community has around extracting fields.
**Improve the data input process**

**Test your inputs**

**Test your inputs**

Before adding new inputs to your production index, it’s 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 aren’t appearing the way you need them to, you can keep working with the test index until you’re happy with the results. When things start looking good, you can edit the inputs to point to your main index instead.

**Use a test index**

To learn how to create and use custom indexes, read "Set up multiple indexes" in the Admin manual. There are a few basic steps, described in detail in that topic:

1. Create the test index, using Splunk Web or the CLI or by editing `indexes.conf` directly. See "Set up multiple 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 searches on the "main" index.) Use the `index=` command:

   ```shell
   index=test_index
   ```

   **Note**: When searching a test index for events coming in from your newly created input, Splunk recommends that you use the `Real-time > All time(real-time)` time range for the field picker. 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 Manager > 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 searches on the "main" index.

Use persistent queues to help prevent data loss

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 is running at a faster rate than the forwarder or indexer can process, to a point where the queue is maxed out, undesired consequences will occur. In the case of UDP, data will drop off the queue and get lost. For other input types, the application generating the data will get 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.
Important: Persistent queues help prevent data loss if Splunk gets backed up. They are not a panacea, however. You can still lose data if Splunk crashes. For example, some input data is held in the in-memory queue, as well as in the persistent queue files. The in-memory data can get lost if there’s a crash. Similarly, data that’s in the parsing or indexing pipeline but that has not yet been written to disk can get lost in the event of a crash.

Note: In 4.2, the persistent queue capability has been re-implemented, in a much improved fashion. It is now a feature of data inputs and is therefore configured in inputs.conf. It is not related in any way to the previous, deprecated persistent queue capability, which was configured through outputs.conf.

When can you use persistent queues?

Persistent queueing 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 input, including Windows scripted inputs

Persistent queues are not available for these input types:

- Monitor
- Batch
- File system change monitor
- Windows event log data
- 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:

`queueSize = <integer>(KB|MB|GB)`
* Max size of the in-memory input queue.
* Defaults to 500KB.

`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]
queueSize=50KB
persistentQueueSize=100MB

Troubleshoot the input process

Troubleshoot the input process

Not finding the events you're looking for?

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

Note: When you add inputs by editing inputs.conf, Splunk may not immediately recognize them. Splunk looks for inputs every 24 hours, starting from the time it was last restarted. This means that if you add a new stanza to monitor a directory or file, it could take up to 24 hours for Splunk 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 by using the add command in the CLI.

Troubleshoot your tailed files

You can use the FileStatus REST endpoint to get the status of your tailed files. For example:

https://serverhost:8089/services/admin/inputstatus/TailingProcessor:FileStatus

Troubleshoot monitor inputs

For a variety of information on dealing with monitor input issues, see this article in the Community Wiki: Troubleshooting Monitor Inputs.

Can't find data coming from a forwarder?

Make sure the forwarder is functioning properly and is visible to the indexer. You can use the Deployment Monitor app to troubleshoot Splunk topologies and get to the root of any forwarder issues. See "About the deployment monitor" for details.