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

The Splunk platform, through a universal or heavy forwarder, can accept events from scripts that you provide.

Scripted input is useful combined with some Windows and *nix command-line tools, such as `ipconfig`, `iostat`, `netstat`, `top`, and so on. You can use scripted input to get data from APIs, other remote data interfaces, and message queues. You can then use commands like `vmstat` and `iostat` on that data to generate metrics and status data. On Windows platforms, you can enable text-based scripts, such those in Perl and Python, with an intermediary Windows batch (.bat) or PowerShell (.ps1) file.

You can configure scripted inputs from the Settings menu in Splunk Web on Splunk Enterprise, or by editing the inputs.conf configuration file on a universal or heavy forwarder.

When a scripted input launches a script, that script inherits the Splunk Enterprise or universal forwarder environment. The only environment variable that might cause problems with scripts and script output generation is the library path, most commonly known as `LD_LIBRARY_PATH` on Linux, Solaris, and FreeBSD. When you use scripted inputs on a Splunk platform instance, clear any environment variables that can affect the operation of a script.

Splunk Enterprise and the universal forwarder log any messages that scripted inputs send to the `stderr` I/O channel to `splunkd.log`.

Prerequisites

To add a scripted input, you must first write an input. To learn how to write scripted inputs, see Scripted input examples for Splunk Cloud Platform or Splunk Enterprise on the Splunk developer portal.

Add a scripted input in Splunk Web

On a Splunk Enterprise or heavy forward instance, follow these high-level steps to add a scripted input in Splunk Web:

1. Go to the Add New page.
2. Select the input source.
3. (Optional) Specify input settings.
4. Review your choices.

Go to the Add Data page

To get to the Add Data page using Settings, follow these steps:

1. In Splunk Web, click **Settings**.
2. Click **Data Inputs**.
3. Click **Scripts**.
4. Click **New** to add an input.

To get to the Add Data page using the Splunk Web homepage, follow these steps:

1. In Splunk Web, click the **Add Data**.
2. Click **Monitor** to monitor a script on the local machine, or click **Forward** to forward data from a script on a remote machine.
   Splunk Web displays the **Add Data - Select Source** page.
3. In the left pane, select **Scripts**.

Forwarding data from scripted inputs requires additional setup.

Select the input source

1. In the **Script Path** drop-down list, select the path where the script resides.
   Splunk Web updates the page to include the **Script Name** drop-down list.
2. In the **Script Name** drop-down list, select the script that you want to run.
   Splunk Web updates the page to populate the **Command** field with the script name.
3. In the **Command** field, add any arguments needed to invoke the script.
4. In the **Interval** field, enter the amount of time, in seconds, that Splunk Enterprise waits before invoking the script.
5. (Optional) In the **Source Name Override** field, enter a new source name to override the default source value, if necessary.
6. Click **Next**.

Specify input settings

You can specify the application context, default host value, and index on the **Input Settings** page. All of these parameters are optional. For more about setting the host value, see About hosts.

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

1. Select the source type for the script. Click **Select** to pick from the list of available source types on the local machine, or click **Manual** to enter the name of a source type.
2. Select the appropriate **Application context** for this input.
3. Set the **Host** name. You have several choices for this setting.
4. Set the **Index** that Splunk Enterprise will send data to. Unless you defined multiple indexes to handle different types of events, leave the value as **default**. In addition to indexes for user data, Splunk Enterprise has multiple utility indexes, which also appear in this drop-down list.
5. Click **Review**.

Review your choices

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

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

Splunk Web displays the Success page.

Add a scripted input with the inputs.conf configuration file

You add a scripted input in the inputs.conf file by adding a `[script]` stanza within that file. You can do this on Splunk Enterprise or the universal forwarder, and then forward that information to Splunk Cloud Platform.

Syntax

The syntax for the `[script]` stanza appears as follows, where `$SCRIPT` is the full path to the location of the script:

```
[script://$SCRIPT]
<attrbute1> = <val1>
<attrbute2> = <val2>
...
```

`$SCRIPT` can also be a file path that ends in the `.path` suffix. This special suffix lets you use the stanza to point to another command or script that exists anywhere on the host file system. See Use the .path suffix to reference external scripts. The file that you refer to in the stanza must follow the location restrictions described in the following section, Where to place the scripts for scripted inputs.

Where to place the scripts for scripted inputs

The script that you refer to in `$SCRIPT` can reside in only one of the following places on the host file system:

- $SPLUNK_HOME/etc/system/bin
- $SPLUNK_HOME/etc/apps/<your_app>/bin
- $SPLUNK_HOME/bin/scripts

As a best practice, put your script in the `bin/` directory that is nearest to the inputs.conf file that calls your script on the host file system. For example, if you configure $SPLUNK_HOME/etc/system/local/inputs.conf, place your script in $SPLUNK_HOME/etc/system/bin/. If you work on an application in $SPLUNK_HOME/etc/apps/$APPLICATION/, put your script in $SPLUNK_HOME/etc/apps/$APPLICATION/bin/.

Attributes

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

Attribute	Description	Default
`interval = <number>|<cron schedule>`	How often to run the specified command. Specify either an integer value representing seconds or a valid cron schedule. When you specify a `cron schedule`, the script does not run at start-up, but rather at the times that the cron schedule defines. Splunk Enterprise keeps one invocation of a script per instance. Intervals are based on when the script completes. If you configure a script to run every 10 minutes and the script takes 20 minutes to complete, the next run occurs 30 minutes after the first run. For constant data streams, enter 1 or a value smaller than the script interval. For one-shot data streams, enter -1. Setting `interval` to -1 causes the script to run each time at start-up.	60 seconds
`index = <string>`		

Attribute	Description	Default
	The index where events from this input are stored. Splunk Enterprise prepends the `<string>` with `index::`. For more information about the index field, see How indexing works in the *Managing Indexers and Clusters of Indexers* manual.	`main`, or whatever you have set as your default index.
`sourcetype = <string>`	Sets the `sourcetype` field for events from this input. The `<string>` is prepended with `sourcetype::`. Explicitly declares the source type for this data, as opposed to letting it be determined automatically. This is important both for searchability and for applying the relevant formatting for this type of data during parsing and indexing. Sets the `sourcetype` key initial value. Splunk Enterprise uses this key during parsing and indexing, particularly to set the `sourcetype` field during indexing. It also uses the `sourcetype` field at search time.	There is no hard-coded default. Splunk Enterprise picks a source type based on various aspects of the data.
`source = <string>`	Sets the `source` key for events from this input. **Caution:** Do not override the source key unless absolutely necessary. Typically, the input layer provides a more accurate string to aid in problem analysis and investigation, accurately recording the file from which the data was retrieved. Consider use of source types, tagging, and search wildcards before overriding this value. Splunk Enterprise prepends `<string>` with `source::`.	The input file path
`disabled = <true | false>`	Whether or not the input will run. Set to true if you want to disable the input.	`false`

Run scripts continuously

If you want the script to run continuously, write the script to never exit and set it on a short interval. This helps to ensure that if a problem occurs, the script restarts. Splunk Enterprise keeps track of scripts it spawned and shuts them down on exit.

Use a wrapper script

As a best practice, write a wrapper script for scripted inputs that use commands with arguments. In some cases, the command can contain special characters that the scripted input escapes when it validates text that you enter in Splunk Web. Updates to a previously configured input will then fail to save.

When validating text, Splunk Enterprise escapes characters that can't be in paths, such as the equal sign (=) and semicolon (;). For example, the following scripted input is not correctly saved when you edit it in Splunk Web because the scripted input escapes the equal (=) sign in the parameter to the myUtil.py utility:

```
[script://$SPLUNK_HOME/etc/apps/myApp/bin/myUtil.py file=my_datacsv]
disabled = false
```

To avoid this problem, write a wrapper script that contains the scripted input, or use the special `.path` argument for the scripted input stanza name. For information on writing wrapper scripts, see Create custom data inputs for Splunk Cloud Platform or Splunk Enterprise in the *Splunk Developer Guide*.

When you update scripted inputs by editing inputs.conf directly, this validation does not occur.

Use the .path suffix to reference external scripts

As an alternative to writing a wrapper script, you can configure the scripted input to reference a script or executable that is anywhere on the host file system.

The script that you refer to can have a single line that calls the script or executable that you want. You can use this file to call a runtime environment that is outside of the Splunk Enterprise environment. For example, if you have both Splunk Enterprise, which comes with Python, and a second installation of Python on the same host, you can use the `.path` method to refer to the second Python installation.

Follow these steps to refer to external scripts with the .path suffix:

1. Use Splunk Web or edit inputs.conf and specify a scripted input stanza with a script name that ends in `.path`. For example:

   ```
   [script://myfile.path]
   disabled = 0
   ```
2. Place the file that you reference in the stanza in the appropriate directory, as described in Where to place the scripts for scripted inputs.
3. Edit the file to specify the script or executable you want. For example:
   ```
   /path/to/myscript -arg1 arg -arg2 arg
   ```

Examples of scripted inputs with inputs.conf

The following examples configure various scripted inputs with inputs.conf.

Unix top command

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

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

   ```
   $ mkdir $SPLUNK_HOME/etc/apps/scripts
   ```
2. Create a `bin/` directory. All scripts must be run out of a `bin/` directory inside your application directory.

   ```
   $ mkdir $SPLUNK_HOME/etc/apps/scripts/bin
   ```
3. Create a script within the `bin/` directory. This example uses a small shell script `top.sh`.

   ```
   $ #!/bin/sh
   top -bn 1  # linux only - different OSes have different parameters
   ```
4. Make the script executable.

   ```
   chmod +x $SPLUNK_HOME/etc/apps/scripts/bin/top.sh
   ```
5. Test that the script works by running it with the shell.

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

The script sends one `top` output.

6. Add the script entry to inputs.conf in $SPLUNK_HOME/etc/apps/scripts/local/.

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

7. By default, Splunk Enterprise breaks the single `top` entry into multiple events, so you might need to modify props.conf to fix this issue. If necessary, edit props.conf and configure the server to break only before something that doesn't 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, you must tell Splunk Enterprise to use the current time. Set the following parameter in props.conf:

```
DATETIME_CONFIG = CURRENT
```

Reference an external script with the .path stanza

The following example uses the special .path stanza setting to reference an external build of Python to run a script on your host.

1. Edit inputs.conf.

```
[script://loglogs.path]
disabled = 0
```

2. Place or create loglogs.path in $SPLUNK_HOME/etc/system/bin.
3. Edit loglogs.path to reference the external version of Python.

```
/usr/bin/python logit.py --source /opt/files/my_files --target /opt/files/my_files/processed
--logfile /opt/src/my_sources/logfiles
```

Set interval attribute to cron schedule

In the previous example, you can also set the `interval` attribute to a cron schedule by specifying strings.

For example, the following string means the script runs once an hour at the top of the hour:

```
0 * * * *
```

The following string means the script runs every 15 minutes from 9 AM until 5 PM, Monday to Friday.

```
*/15 9-17 * * 1-5
```

The following string means the script runs at 15, 35, and 55 minutes after the hour between midnight and 7 AM and again between 8 PM and midnight, on the first of every even-numbered month, such as February, April, June, and so on.

```
15,35,55 0-6,20-23 1 */2 *
```

For more information about setting cron schedules, see CRONTAB(5) in https://crontab.org on the Crontab website.