Splunk® Enterprise
Getting Data In 9.0.0

Format events for HTTP Event Collector

Generated: 7/21/2022 12:27 pm

Format events for HTTP Event Collector

The HTTP Event Collector (HEC) receives events from clients in a series of HTTP requests. Each request can contain a HEC token, a channel identifier header, event metadata, or event data, depending on whether your events are raw or have been formatted in accordance with the JavaScript Object Notation (JSON) standard. You can format events for HEC in both Splunk Cloud Platform and Splunk Enterprise.

To learn more about HEC, how it works, and how to set it up, see Set up and use the HTTP Event Collector in Splunk Web.

HEC token

Before the HTTP Event Collector can accept your data for indexing, you must authenticate to the Splunk Cloud Platform or Splunk Enterprise instance on which it runs. You do this using the token you generate when you create a new HEC input. When you use the token management endpoint on the Splunk server to generate a token, it generates the token in the form of a globally unique identifier (GUID). See Use cURL to manage HTTP Event Collector tokens, events, and services for more information. Using the token management endpoint guarantees that the token is unique.

You have several ways to authenticate to the instance: by HTTP authentication, basic authentication, or a query string.

HTTP authentication

Place the token in the authorization header of each HTTP request as follows:

```
"Authorization: Splunk <hec_token>"
```

The following example shows HTTP authentication in the context of a typical `curl` command that you use to communicate with HEC:

```
curl -H "Authorization: Splunk 12345678-1234-1234-1234-1234567890AB"
https://mysplunkserver.example.com:8088/services/collector/event -d '{"sourcetype": "my_sample_data",
"event": "http auth ftw!"}'
```

Basic authentication

To perform basic authentication into HEC, include a colon-separated user-password pair in the request after `-u`, inserting the HEC token as the `<password>: "<user>:<password>"`. The `<user>` can be any string. For example:

```
-u "x:<hec_token>"
```

The following example shows the basic authentication in context of a `curl` command:

```
curl -u "x:12345678-1234-1234-1234-1234567890AB"
https://mysplunkserver.example.com:8088/services/collector/event -d '{"sourcetype": "my_sample_data",
"event": "basic auth ftw!"}'
```

Query string

You can specify the HEC token as a query string in the URL that you specify in your queries to HEC. For example:

```
?token=<hec_token>
```

The following example shows the query string authentication in context of a `curl` command:

```
curl
https://mysplunkserver.example.com:8088/services/collector/event?token=12345678-1234-1234-1234-1234567890AB
-d '{"sourcetype": "my_sample_data", "event": "query string ftw!"}'
```

You must also enable query string authentication on a per-token basis. Open a case with Splunk Support to edit the file in the $SPLUNK_HOME/etc/apps/splunk_httpinput/local/inputs.conf file. Your tokens appear by name in this file, in the form of http://<token_name>.

Within the stanza for each token you want to enable query string authentication, have Splunk Support add or change the following setting:

```
allowQueryStringAuth = true
```

On Splunk Enterprise, you can make these configurations directly on the instance. The changes take effect after you restart the instance.

Channel identifier header

If the request to HEC includes raw events and indexer acknowledgement is enabled for the HEC token, you must include the `X-Splunk-Request-Channel` header field in the request. You must set the header field to a unique channel identifier (GUID). See About channels and sending data for more information. The following example shows a cURL command that constitutes a valid request:

```
curl https://http-inputs-<customer>.splunkcloud.com/services/collector/raw  -H "X-Splunk-Request-Channel:
FE0ECFAD-13D5-401B-847D-77833BD77131" -H "Authorization: Splunk BD274822-96AA-4DA6-90EC-18940FB2414C" -d
'<raw data string>' -v
```

Alternatively, you can set the `X-Splunk-Request-Channel` header field as a URL query parameter:

```
curl https://http-inputs-<customer>.splunkcloud.com/services/collector/raw?channel=FE0ECFAD-13D5-401B-847D
-77833BD77131 -H "Authorization: Splunk BD274822-96AA-4DA6-90EC-18940FB2414C" -d '<raw data string>' -v
```

If the token with which you're authenticating to HTTP Event Collector has indexer acknowledgement enabled, you must also include the channel identifier with your indexer status query. For more information, see Enable indexer acknowledgement.

Event metadata

The following table shows keys that you can include in event metadata. These keys are all optional. Any key-value pairs that you don't include in the event are set to values defined for the token on the Splunk platform instance.

Key	Description
`"time"`	The event time. The default time format is UNIX time format, in the format `<sec>.<ms>` and depends on your local timezone. For example, `1433188255.500` indicates 1433188255 seconds and 500 milliseconds after epoch, or Monday, June 1, 2015, at 7:50:55 PM GMT.
`"host"`	The host value to assign to the event data. This key is typically the hostname of the client from which you're sending data.
`"source"`	The source value to assign to the event data. For example, if you're sending data from an app you're developing, set this key to the name of the app.
`"sourcetype"`	The sourcetype value to assign to the event data.
`"index"`	The name of the index by which the event data is to be indexed. The index you specify here must be within the list of allowed indexes if the token has the `indexes` parameter set.
`"fields"`	The fields key isn't applicable to raw data. This key specifies a JSON object that contains a flat (not nested) list of explicit custom fields to be defined at index time. Requests containing the `"fields"` property must be sent to the `/collector/event` endpoint, or else they aren't indexed. For more information, see Indexed field extractions.

With raw events, you can configure metadata at the global level, at the token level, and at the request level using the query string. See About Event Collector tokens and Use HEC from the CLI for more information. Metadata specified within a request applies to all events that are extracted from the request.

Event data

Event data can be assigned to the `"event"` key within the JSON object in the HTTP request, or it can be raw text. The `"event"` key is at the same level within the JSON event packet as the metadata keys. Within the `"event"` key-value curly brackets, the data can be in whatever format you want: a string, a number, another JSON object, and so on.

You can batch multiple events in one event packet by combining them within the request. By batching the events, you're specifying that any event metadata within the request is to apply to all of the events contained in the request. Batching events can significantly speed performance when you need to index large quantities of data.

Example 1: Event metadata as a string contained in a JSON

The following example is of a properly formatted event metadata and event data as a string contained within a JSON object:

```
{
    "time": 1426279439, // epoch time
    "host": "localhost",
    "source": "random-data-generator",
    "sourcetype": "my_sample_data",
    "index": "main",
    "event":  "Hello world!"
}
```

Example 2: JSON object as event data

The following example is of a JSON object as the event data within a properly formatted event:

```
{
    "time": 1437522387,
    "host": "dataserver992.example.com",
    "source": "testapp",
    "event": {
        "message": "Something happened",
        "severity": "INFO"
    }
}
```

Example 3: Batched data

The following example is of batched data. The batch protocol for HTTP Event Collector involves event objects stacked one after the other, and not in a JSON array. These events, although they contain only the `"event"` and `"time"` keys, are still valid:

```
{
  "event":"event 1",
  "time": 1447828325
}

{
  "event":"event 2",
  "time": 1447828326
}
```

Example 4: cURL statement with authorization header

The following example is a simplified "hello world!" cURL statement that includes the authorization header, a destination endpoint, and simple event data. Note that the request is going to the `/services/collector/event` endpoint, which is where all JSON-formatted event requests must go:

```
curl -H "Authorization: Splunk 12345678-1234-1234-1234-1234567890AB"
https://localhost:8088/services/collector/event -d '{"event":"hello world"}'
```

Example 5: cURL statement sending raw data

The following example cURL statement demonstrates sending raw event data. Note the addition of the channel ID, which is required when sending raw event data. Also, the request is going to the `/services/collector/raw` endpoint, which is where all raw event requests must go:

```
curl http://localhost:8088/services/collector/raw -H 'Authorization: Splunk
B5A79AAD-D822-46CC-80D1-819F80D7BFB0' -H 'x-splunk-request-channel: 18654C68-B28B-4450-9CF0-6E7645CA60CA'
-d 'hello world'
```

Example 6: cURL statement as a URL parameter

Alternately, this example cURL statement passes the channel ID as a URL parameter:

```
curl http://localhost:8088/services/collector/raw?channel=18654C68-B28B-4450-9CF0-6E7645CA60CA -H
'Authorization: Splunk B5A79AAD-D822-46CC-80D1-819F80D7BFB0'  -d 'hello world'
```

For additional examples on how to format and send event data using cURL, see Use cURL to manage HTTP Event Collector tokens, events, and services.

Event parsing

The HTTP Event Collector endpoint extracts the events from the HTTP request and parses them before sending them to indexers. Because the event data formats are predetermined, the Splunk platform can parse and index your data quickly. This faster parsing results in improved data throughput and reduced event processing time compared to other methods of getting data in.

You can configure extraction rules in the props.conf configuration file. To learn more, see Configure rule-based source type recognition.

Raw event parsing

Raw event parsing is available in the current release of Splunk Cloud Platform and Splunk Enterprise 6.4.0 and higher.

HTTP Event Collector can parse raw text and extract one or more events. HEC expects that the HTTP request contains one or more events with line-breaking rules in effect. Once HEC accepts the request, it passes its events into the pipeline, which extracts the fields, such as timestamps. HEC uses a line-breaking strategy that is based on the timestamp, but you can override the strategy by setting a sourcetype in the props.conf configuration file. See Configure rule-based source type recognition.

You must include events within a single HTTP request. They cannot span multiple requests.

To accommodate raw events, use the services/collector/raw endpoint.

This endpoint requires an additional `X-Splunk-Request-Channel` header field, which you must set to a unique channel identifier (GUID). See About channels and sending data for more information. You must include a channel identifier with each HTTP request that contains raw events. The following example is of a cURL statement that constitutes a valid request:

```
curl https://http-inputs-<customer>.splunkcloud.com/services/collector/raw  -H "X-Splunk-Request-Channel:
FE0ECFAD-13D5-401B-847D-77833BD77131" -H "Authorization: Splunk BD274822-96AA-4DA6-90EC-18940FB2414C" -d
'<raw data string>' -v
```

Alternatively, the `X-Splunk-Request-Channel` header field can be sent as a URL query parameter:

```
curl https://http-inputs-<customer>.splunkcloud.com/services/collector/raw?channel=FE0ECFAD-13D5-401B-847D
-77833BD77131 -H "Authorization: Splunk BD274822-96AA-4DA6-90EC-18940FB2414C" -d '<raw data string>' -v
```

If the token with which you're authenticating to HTTP Event Collector has indexer acknowledgement enabled, you must also include the channel identifier with your indexer status query. For more information, see Enable indexer acknowledgement.

With raw events, you can configure metadata at the global level, at the token level, and at the request level using the query string. See About Event Collector tokens and Use HEC from the CLI for more information. Metadata specified within a request applies to all events that are extracted from the request.

Timestamp extraction rules are enabled at the sourcetype level to extract timestamps. The most common timestamp formats are recognized, such as the `"current-time"` key, but if no timestamp can be extracted, one is assigned based on the current time. For other metadata, you can configure extraction rules in the props.conf file. See Configure rule-based source type recognition for more information.

For more examples of cURL requests to the `services/collector/raw` endpoint, see Input endpoint examples in the Splunk Enterprise *REST API Reference Manual*.

For more information about channels, see the *About channels and sending data* in Enable indexer acknowledgement.