Event_dispatcher filter plugin

Unresolved directive in index.asciidoc - include::../../../../logstash/docs/include/plugin_header.asciidoc[]

Description

The Event_dispatcher filter works with a configuration json file provided by the user. The configuration json file allows the user to define different filter logic paths like what multiple pipelines do. Using multiple pipelines has the following limitations.

- It can not handle the case: when event flows pass through require different filter paths based on its runtime values, for example, some identity field value that can only be known (said, by a quick parsing) in runtime.
- It requires extra inputs for each added new pipeline. For example in the tcp port input case, each new added pipeline needs at least one added tcp input port. Opening too many tcp ports may not be desirable in many use cases.
- It is harder to set up and tune for performance compared to using a single pipeline.

The Event_dispatcher filter does not have the above limitations. It simply dispatches event filter processing to the filter logic defined in the configuration json file.

Content of the config json file: [list of filter_parts]

filter_part:

```
{
    "type":
        "plugin" | "filter"
    "event_filter_tags_to_process":
        list of event_filter_tag values, examples:
        ["event_filter_tag_1"],
        ["event_filter_tag_1", "event_filter_tag_2"]
    "set_done_if_executed":
        false | true, default: false
    "name" (for "plugin" type):
        name of the logstash plugin to use, examples: "csv" | "json" | ...
    "options" (for "plugin" type):
        the configuration options for the filter plugin
    "filter_parts" (for "filter" type):
        [ list of filter_parts] |
        path to a config json file containing  [ list of filter_parts]
}
```

Here is a log sample to show a typical use case, where the events may require 6 different filter logic paths to parse.

```
    level_1_tag=csv,event_log=1,2019-08-29T01:53:12Z,Amex,Giovanna Van der
Linde,Female,185.216.194.245,Industrial,Philippines,55
    level_1_tag=json,event_log=a:1 b:2 c:3
    level_1_tag=filter_1,event_log=level_1x_tag=filter_11,event_log=(kv=)a=1,b=2,c=3,d=4,e=5
    level_1_tag=filter_1,event_log=level_1x_tag=filter_11,event_log=(json){"Property 1":"value x","Property
2":"value y"}
    level_1_tag=filter_1,event_log=level_1x_tag=filter_11,event_log=(kv|)a|1,b|2,c|3,d|4,e|5
    level_1_tag=filter_1,event_log=level_1x_tag=filter_11,event_log=(kv%)a%1,b%2,c%3,d%4,e%5
```

The "Example" section at the end provides a complete configuration example for handling the events shown in the above log samples.

Note that Event_dispacther filter can handle complicated real world cases, even though the log sample is artificially made trivial just enough to show how to use it.

Event_dispatcher Filter Configuration Options

This plugin supports the following configuration options plus the [plugins-filters-event_dispatcher-common-options] described later.

Setting	Input type	Required
`config_dir`	string	Yes
`config_json_filename`	string	No
`event_filter_tag_field`	string	No
`event_filter_done_field`	string	No

Also see [plugins-filters-event_dispatcher-common-options] for a list of options supported by all filter plugins.

`config_dir`

- Value type is string

File folder path where the configuration json file is located.

`config_json_filename`

- Value type is string
- Default value is "filter.json"

Name of the configuration json file

`event_filter_tag_field`

- Value type is string
- Default value is "event_filter_tag"

Name of event_filter_tag field, the value of this field can be read and/or set by each filter_part for correctly controlling the event's filter path.

`event_filter_done_field`

- Value type is string
- Default value is "event_filter_done"

Name of event_filter_done field, the value of this field is used by each filter_part to determine if it can stop traversing all filter_parts’s "filter" methods.

Example

The following is a complete case to show how to configure Event_dispatcher to handle the sample events discussed in the description section.

logstash configuration file:

```ruby
    input {
          tcp {
            port => 9501
          }
        }


filter {
    event_filter_dispatcher {
        config_dir => "/Users/user/tmp/etc/config"
    }
}


output {
    stdout { codec => rubydebug }
}
```

"filter.json": (in the folder: /Users/user/tmp/etc/config")

```
[
  {
    "type": "plugin",
    "name": "ruby",
    "description": "Get: level_1_tag/event_log, set: event_filter_tag/message",
    "options": {
      "path": "${config_dir}/code_1.rb"
    }
  },
  {
    "type": "plugin",
    "name": "csv",
    "event_filter_tags_to_process": [
      "csv"
    ],
    "options": {
      "columns": [
        "id",
        "timestamp",
        "paymentType",
        "name",
        "gender",
        "ip_address",
        "purpose",
        "country",
        "age"
      ]
    },
    "set_event_filter_done_if_executed": true
  },
  {
    "type": "plugin",
    "name": "kv",
    "event_filter_tags_to_process": [
      "kv"
    ],
    "options": {
      "field_split": " ",
      "value_split": ":"
    },
    "set_event_filter_done_if_executed": true
  },
  {
    "type": "filter",
    "event_filter_tags_to_process": [
      "filter_1"
    ],
    "set_event_filter_done_if_executed": true,
    "filter_parts": "${config_dir}/filter_1.json"
  }
]
```

"filter_1.json":

```
[
  {
    "description": "Get: level_2_tag/event_log, set: event_filter_tag/message",
    "type": "plugin",
    "name": "ruby",
    "options": {
      "path": "${config_dir}/code_2.rb"
    }
  },
  {
    "type": "filter",
    "event_filter_tags_to_process": [
      "filter_11"
    ],
    "set_event_filter_done_if_executed": true,
    "filter_parts": "${config_dir}/filter_11.json"
  }
]
```

"filter_11.json":

```
[
  {
    "type": "plugin",
    "name": "ruby",
    "description": "Get: (json)|(kv=)|(kv==)|(kv%),set: event_filter_tag/message",
    "options": {
      "path": "${config_dir}/code_3.rb"
    }
  },
  {
    "type": "plugin",
    "name": "kv",
    "event_filter_tags_to_process": [
      "(kv=)"
    ],
    "options": {
      "field_split": ",",
      "value_split": "="
    },
    "set_event_filter_done_if_executed": true
  },
  {
    "type": "plugin",
    "name": "json",
    "event_filter_tags_to_process": [
      "(json)"
    ],
    "options": {
      "source": "message"
    },
    "set_event_filter_done_if_executed": true
  },
  {
    "type": "plugin",
    "name": "kv",
    "event_filter_tags_to_process": [
      "(kv|)"
    ],
    "options": {
      "field_split": ",",
      "value_split": "|"
    },
    "set_event_filter_done_if_executed": true
  },
  {
    "type": "plugin",
    "name": "kv",
    "event_filter_tags_to_process": [
      "(kv%)"
    ],
    "options": {
      "field_split": ",",
      "value_split": "%"
    },
    "set_event_filter_done_if_executed": true
  }
]
```

code_1.rb:

RUBY

```ruby
def filter(event)
  m = event.get("message")
  if m.start_with?("level_1_tag=")
    i1 = "level_1_tag=".length
    i2 = m.index(",")
    event.set("event_filter_tag", m[i1..i2-1])
    m1 = m[i2+1..]
    if m1.start_with?("event_log=")
      event.set("message", m1["event_log=".length..])
    end
  end
  return [event]
end
```

code_2.rb:

RUBY

```ruby
def filter(event)
  m = event.get("message")
  if m.start_with?("level_1x_tag=")
    i1 = "level_1x_tag=".length
    i2 = m.index(",")
    event.set("event_filter_tag", m[i1..i2-1])
    m1 = m[i2+1..]
    if m1.start_with?("event_log=")
      event.set("message", m1["event_log=".length..])
    end
  end
  return [event]
end
```

code_3.rb:

RUBY

```ruby
def filter(event)
  m = event.get("message")
  # e.g. m = "level_2_tag=filter_11,event_log=(kv=)a=1,b=2,c=3,d=4,e=5"
  if m.start_with?("(")
    i = m.index(")")
    event.set("event_filter_tag", m[0..i])
    event.set("message", m[i+1..])
  end
  return [event]
end
```

Last updated 2022-12-03 09:21:11 -0800