LumberMill LogParser Documentation

Release 0.5

Björn Puttmann

Jun 28, 2022

Contents

1 LumberMill **3**
- 1.1 Introduction . . . 3
- 1.2 Compatibility and Performance . . . 3
- 1.3 Installation . . . 3
- 1.4 Working modules . . . 4
  - 1.4.1 Event inputs . . . 4
  - 1.4.2 Event parsers . . . 4
  - 1.4.3 Event modifiers . . . 5
  - 1.4.4 Outputs . . . 5
  - 1.4.5 Misc modules . . . 6
  - 1.4.6 Cluster modules . . . 6
  - 1.4.7 Plugins . . . 6
- 1.5 Event flow basics . . . 6
- 1.6 Configuration example (with explanations) . . . 6
- 1.7 Configuration basics . . . 12
- 1.8 Event field notation . . . 12
  - 1.8.1 Notation in configuration fields like source_field or target_field . . . 13
  - 1.8.2 Notation in strings . . . 13
  - 1.8.3 Notation in module filters . . . 13
- 1.9 A rough sketch for using LumberMill with syslog-ng . . . 13

2 Input modules **15**
- 2.1 Beats . . . 15
- 2.2 ElasticSearch . . . 16
- 2.3 File . . . 17
- 2.4 Kafka . . . 18
- 2.5 NmapScanner . . . 18
- 2.6 RedisChannel . . . 19
- 2.7 RedisList . . . 19
- 2.8 SQS . . . 20
- 2.9 Sniffer . . . 20
- 2.10 Spam . . . 21
- 2.11 StdIn . . . 21
- 2.12 Tcp . . . 22
- 2.13 Udp . . . 22
- 2.14 UnixSocket . . . 23

2.15 ZmqTornado . . . 24

3 Parser modules **25**
3.1 Base64 . . . 25
3.2 Collectd . . . 25
3.3 Csv . . . 26
3.4 DateTime . . . 26
3.5 DomainName . . . 27
3.6 Inflate . . . 27
3.7 Json . . . 28
3.8 Line . . . 28
3.9 MsgPack . . . 29
3.10 NetFlow . . . 29
3.11 Regex . . . 30
3.12 SyslogPrival . . . 30
3.13 Url . . . 31
3.14 UserAgent . . . 31
3.15 XPath . . . 32

4 Modifier modules **33**
4.1 AddDateTime . . . 33
4.2 AddDnsLookup . . . 33
4.3 AddGeoInfo . . . 34
4.4 DropEvent . . . 35
4.5 ExecPython . . . 35
4.6 Facet . . . 36
4.7 HttpRequest . . . 36
4.8 Math . . . 37
4.9 ModifyFields . . . 38
4.10 MergeEvent . . . 41
4.11 Permutate . . . 42

5 Misc modules **43**
5.1 EventBuffer . . . 43
5.2 Cache . . . 43
5.3 SimpleStats . . . 44
5.4 Metrics . . . 45
5.5 Tarpit . . . 46
5.6 Throttle . . . 46

6 Output modules **49**
6.1 DevNull . . . 49
6.2 ElasticSearch . . . 49
6.3 File . . . 50
6.4 Graphite . . . 51
6.5 Kafka . . . 51
6.6 Logger . . . 52
6.7 MongoDb . . . 52
6.8 RedisChannel . . . 53
6.9 RedisList . . . 53
6.10 SQS . . . 54
6.11 StdOut . . . 55
6.12 Syslog . . . 55
6.13 Udp . . . 56
6.14 WebHdfs . . . 56

6.15 Zabbix . . . 57
6.16 Zmq . . . 57

7 Cluster modules **59**
7.1 Pack . . . 59
7.2 PackConfiguration . . . 60

8 Plugin modules **61**
8.1 WebGui . . . 61
8.2 WebserverTornado . . . 61

9 Indices and tables **63**

Contents:

CHAPTER 1

LumberMill

1.1 Introduction

Collect, parse and store logs with a configurable set of modules. Inspired by logstash but with a smaller memory footprint and faster startup time. Can also run multiprocessed to avoid GIL related restrictions.

1.2 Compatibility and Performance

To run LumberMill you will need Python 3.2+. For Python 2 support, please use an older version of this tool. For better performance, I heartly recommend running LumberMill with pypy. The performance gain can be up to 5-6 times events/s throughput running single processed.

1.3 Installation

via pypi

```
pip install LumberMill
```

manually

Clone the github repository to /opt/LumberMill (or any other location that fits you better :):

```
git clone https://github.com/dstore-dbap/LumberMill.git /opt/LumberMill
```

Install the dependencies with pip:

```
cd /opt/LumberMill
python setup.py install
```

Now you can give LumberMill a testdrive with:

```
wget https://raw.githubusercontent.com/dstore-dbap/LumberMill/master/conf/example-
↪stdin.conf
echo "I'm a lumberjack, and I'm okay" | lumbermill -c conf/example-stdin.conf
```

If you get a "command not found" please check your pythonpath setting. Depending on how you installed LumberMill, the executable can either be found in the bin dir of your python environment (e.g. /usr/lib64/pypy/bin/lumbermill) or in your default path (e.g. /usr/local/bin/lumbermill).

Other basic configuration examples: https://github.com/dstore-dbap/LumberMill/tree/master/conf/.

For a how-to running LumberMill, Elasticsearch and Kibana on CentOS, feel free to visit http://www.netprojects.de/collect-visualize-your-logs-with-lumbermill-and-elasticsearch-on-centos/.

1.4 Working modules

1.4.1 Event inputs

- Beats, read elastic beat inputs, e.g. filebeat.
- ElasticSearch, get documents from elasticsearch.
- File, read data from files.
- Kafka, receive events from apache kafka.
- NmapScanner, scan network with nmap and emit result as new event.
- RedisChannel, read events from redis channels.
- RedisList, read events from redis lists.
- Sniffer, sniff network traffic.
- Spam, what it says on the can - spams LumberMill for testing.
- SQS, read messages from amazons simple queue service.
- StdIn, read stream from standard in.
- Tcp, read stream from a tcp socket.
- Udp, read data from udp socket.
- UnixSocket, read stream from a named socket on unix like systems.
- ZeroMQ, read events from a zeromq.

1.4.2 Event parsers

- Base64, parse base64 data.
- Collectd, parse collectd binary protocol data.
- CSV, parse a char separated string.
- DateTime, parse a string to a dateobject and convert it to different date pattern.
- DomainName, parse a domain name or url to tld, subdomain etc. parts.
- Inflate, inflates any fields with supported compression codecs.
- Json, parse a json formatted string.

- Line, split lines at a seperator and emit each line as new event.
- MsgPack, parse a msgpack encoded string.
- Regex, parse a string using regular expressions and named capturing groups.
- SyslogPrival, parse the syslog prival value (RFC5424).
- Url, parse the query string from an url.
- UserAgent, parse a http user agent string.
- XPath, parse an XML document via an xpath expression.

1.4.3 Event modifiers

- AddDateTime, adds a timestamp field.
- AddDnsLookup. adds dns data.
- AddGeoInfo, adds geo info fields.
- DropEvent, discards event.
- ExecPython, execute custom python code.
- Facet, collect all encountered variations of en event value over a configurable period of time.
- HttpRequest, execute an arbritrary http request and store result.
- Math, execute arbitrary math functions.
- MergeEvent, merge multiple events to one single event.
- Field, some methods to change extracted fields, e.g. insert, delete, replace, castToInteger etc.
- Permutate, takes a list in the event data emits events for all possible permutations of that list.

1.4.4 Outputs

- DevNull, discards all data that it receives.
- ElasticSearch, stores data entries in an elasticsearch index.
- File, store events in a file.
- Graphite, send metrics to graphite server.
- Kafka, publish incoming events to kafka topic.
- Logger, sends data to lumbermill internal logger for output.
- MongoDb, stores data entries in a mongodb index.
- RedisChannel, publish incoming events to redis channel.
- RedisList, publish incoming events to redis list.
- StdOut, prints all received data to standard out.
- SQS, sends events to amazons simple queue service.
- Syslog, send events to syslog.
- Udp, send events to udp server.
- WebHdfs, store events in hdfs via webhdfs.

- Zabbix, send metrics to zabbix.
- Zmq, sends incoming event to zeromq.

1.4.5 Misc modules

- EventBuffer, store received events in a persistent backend until the event was successfully handled.
- Cache, use cache to store and retrieve values, e.g. to store the result of the XPathParser modul.
- SimpleStats, simple statistic module just for event rates etc.
- Metrics, Configurable fields for different metric data.
- Tarpit, slows event propagation down - for testing.
- Throttle, throttle event count over a given time period.

1.4.6 Cluster modules

- Pack, base pack module. Handles pack leader and pack member discovery.
- PackConfiguration, syncs leader configuration to pack members.

1.4.7 Plugins

- WebGui, a web interface to LumberMill.
- WebserverTornado, base webserver module. Handles all incoming requests.

1.5 Event flow basics

- an input module receives an event.
- the event data will be wrapped in a default event dictionary of the following structure: { “data”: payload, “lumbermill”: { “event_id”: unique event id, “event_type”: “Unknown”, “received_from”: ip address of sender, “source_module”: caller_class_name, } }
- the input module sends the new event to its receivers. Either by adding it to a queue or by calling the receivers handleEvent method.
- if no receivers are configured, the next module in config will be the default receiver.
- each following module will process the event via its handleEvent method and pass it on to its receivers.
- each module can have an input filter and an output filter to manage event propagation through the modules.
- output modules can not have receivers.

1.6 Configuration example (with explanations)

To give a short introduction of how LumberMill works, here is a sample configuration. Its receiving apache and nginx access logs via syslog messages from a syslog server and msgpacked events from python-beaver and stores them in an elasticsearch backend. Below, I will explain each section in more detail.

```
# Sets number of parallel LumberMill processes.
- Global:
   workers: 2

# Listen on all interfaces, port 5151.
- input.Tcp:
   port: 5151
   receivers:
    - RegexParser

# Listen on all interfaces, port 5152.
- input.Tcp:
   port: 5152
   mode: stream
   chunksize: 32768

# Decode msgpacked data.
- parser.MsgPack:
   mode: stream

# Extract fields.
- parser.RegexParser:
   source_field: data
   hot_rules_first: True
   field_extraction_patterns:
    - httpd_access_log: '(?P<remote_ip>\d+\.\d+\.\d+\.\d+)\s+(?P<identd>\w+|-)\s+(?P
↪<user>\w+|-)\s+\[(?P<datetime>\d+\/\w+\/\d+:\d+:\d+:\d+\s.\d+)\]\s+\"(?P<url>.*)\
↪"\s+(?P<http_status>\d+)\s+(?P<bytes_send>\d+)'
    - http_common_access_log: '(?P<remote_ip>\d+\.\d+\.\d+\.\d+)\s(?P<x_forwarded_for>
↪\d+\.\d+\.\d+\.\d+)\s(?P<identd>\w+|-)\s(?P<user>\w+|-)\s\[(?P<datetime>\d+\/\w+\/
↪\d+:\d+:\d+:\d+\s.\d+)\]\s\"(?P<url>.*)\"\s(?P<http_status>\d+)\s(?P<bytes_send>\d+)
↪'
    - iptables: '(?P<syslog_prival>\<\d+\>)(?P<log_timestamp>
↪\w+\s+\d+\s+\d+:\d+:\d+)\s+(?P<host>[\w\-\._]+)\s+kernel:.*?\ iptables\ (?P
↪<iptables_action>.*?)\ :\ IN=(?P<iptables_in_int>.*?)\ OUT=(?P<iptables_out_int>.*?
↪)\ SRC=(?P<iptables_src>.*?)\ DST=(?P<iptables_dst>.*?)\ LEN=(?P<iptables_len>.*?)\␣
↪.*?PROTO=(?P<iptables_proto>.*?)\ SPT=(?P<iptables_spt>.*?)\ DPT=(?P<iptables_dpt>.
↪*?)\ WINDOW=.*'
   receivers:
    - misc.SimpleStats:
       filter: $(lumbermill.event_type) != 'Unknown'
    # Print out messages that did not match
    - output.StdOut:
       filter: $(lumbermill.event_type) == 'Unknown'

# Print out some stats every 10 seconds.
- misc.SimpleStats:
   interval: 10

# Extract the syslog prival from events received via syslog.
- parser.SyslogPrival:
   source_field: syslog_prival

# Add a timestamp field.
- modifier.AddDateTime:
   format: '%Y-%m-%dT%H:%M:%S.%f'
   target_field: "@timestamp"
```

(continues on next page)

(continued from previous page)

```
# Add geo info based on the lookup_fields. The first field in <source_fields> that␣
↪yields a result from geoip will be used.
- modifier.AddGeoInfo:
   geoip_dat_path: /usr/share/GeoIP/GeoLiteCity.dat
   source_fields: [x_forwarded_for, remote_ip]
   geo_info_fields: ['latitude', 'longitude', 'country_code']

# Nginx logs request time in seconds with milliseconds as float. Apache logs␣
↪microseconds as int.
# At least cast nginx to integer.
- modifier.Math:
   filter: if $(server_type) == "nginx"
   target_field: request_time
   function: float($(request_time)) * 1000

# Map field values of <source_field> to values in <map>.
- modifier.Field:
   filter: if $(http_status)
   action: map
   source_field: http_status
   map: {100: 'Continue', 200: 'OK', 301: 'Moved Permanently', 302: 'Found', 304:
↪'Not Modified', 400: 'Bad Request', 401: 'Unauthorized', 403: 'Forbidden', 404:
↪'Not Found', 500: 'Internal Server Error', 502: 'Bad Gateway'}

# Kibana's 'bettermap' panel needs an array of floats in order to plot events on map.
- modifier.Field:
   filter: if $(latitude)
   action: merge
   source_fields: [longitude, latitude]
   target_field: geoip

# Extarct some fields from the user agent data.
- parser.UserAgent:
   source_fields: user_agent

# Parse the url into its components.
- parser.Url:
   source_field: uri
   target_field: uri_parsed
   parse_querystring: True
   querystring_target_field: params

# Store events in elastic search.
- output.ElasticSearch:
   nodes: [localhost]
   store_interval_in_secs: 5

- output.StdOut
```

Let me explain it in more detail:

```
# Sets number of parallel LumberMill processes.
- Global:
   workers: 2
```

The Global section lets you configure some global properties of LumberMill. Here the number of parallel processes

is set. In order to be able to use multiple cores with python (yay to the GIL) LumberMill can be started with multiple parallel processes. Default number of workers is CPU_COUNT - 1.

```
# Listen on all interfaces, port 5151.
- input.Tcp:
   port: 5151
   receivers:
    - RegexParser
```

Starts a tcp server listening on all local interfaces port 5151. Each module comes with a set of default values, so you only need to provide settings you need to customize. For a description of the default values of a module, refer to the README.md in the modules directory or its docstring. By default, a module will send its output to the next module in the configuration. To set a custom receiver, set the receivers value. This module will send its output directly to RegexParser.

```
# Listen on all interfaces, port 5152.
- input.Tcp:
   port: 5152
   mode: stream
   chunksize: 32768
```

Also starts a tcp server, listening on port 5152. The first tcp server uses newline as separator (which is the default) for each received event. Here, the sever reads in max. 32k of data and passes this on to the next module.

```
# Decode msgpacked data.
- parser.MsgPack:
   mode: stream
```

Decode the received data from the above tcp server in msgpack format. This can be used to e.g. handle data send via python-beaver

```
# Extract fields.
- parser.Regex:
   source_field: data
   hot_rules_first: True
   field_extraction_patterns:
    - httpd_access_log: '(?P<remote_ip>\d+\.\d+\.\d+\.\d+)\s+(?P<identd>\w+|-)\s+(?P
↪<user>\w+|-)\s+\[(?P<datetime>\d+\/\w+\/\d+:\d+:\d+:\d+\s.\d+)\]\s+\"(?P<url>.*)\
↪"\s+(?P<http_status>\d+)\s+(?P<bytes_send>\d+)'
    - http_common_access_log: '(?P<remote_ip>\d+\.\d+\.\d+\.\d+)\s(?P<x_forwarded_for>
↪\d+\.\d+\.\d+\.\d+)\s(?P<identd>\w+|-)\s(?P<user>\w+|-)\s\[(?P<datetime>\d+\/\w+\/
↪\d+:\d+:\d+:\d+\s.\d+)\]\s\"(?P<url>.*)\"\s(?P<http_status>\d+)\s(?P<bytes_send>\d+)
↪'
    - iptables: '(?P<syslog_prival>\<\d+\>)(?P<log_timestamp>
↪\w+\s+\d+\s+\d+:\d+:\d+)\s+(?P<host>[\w\-\._]+)\s+kernel:.*?\ iptables\ (?P
↪<iptables_action>.*?)\ :\ IN=(?P<iptables_in_int>.*?)\ OUT=(?P<iptables_out_int>.*?
↪)\ SRC=(?P<iptables_src>.*?)\ DST=(?P<iptables_dst>.*?)\ LEN=(?P<iptables_len>.*?)\␣
↪.*?PROTO=(?P<iptables_proto>.*?)\ SPT=(?P<iptables_spt>.*?)\ DPT=(?P<iptables_dpt>.
↪*?)\ WINDOW=.*'
   receivers:
    - misc.SimpleStats:
       filter: $(lumbermill.event_type) != 'Unknown'
    # Print out messages that did not match
    - output.StdOut:
       filter: $(lumbermill.event_type) == 'Unknown'
```

Use regular expressions to extract fields from a log event. source_field sets the field to apply the regex to. With hot_rules_first set to True, the expressions will be applied in order of their hit counts. httpd_access_log will set the

event type to "httpd_access_log" if the expression matches. Named groups are used to set the field names. Grok patterns from Logstash can also be used. In the receivers section, we can find output filters. These can be used to only send selected events to the receiving module. As to the notation of event fields in such filters, please refer to the "Event field notation" section later in this document. In this example the output filter uses the event metadata lumbermill field. This data is set by LumberMill for every event received and would look like this:

```
'lumbermill': {'event_id': '90818a85f3aa3af302390bbe77fbc1c87800',
               'event_type': 'Unknown',
               'pid': 7800,
               'received_by': 'vagrant-centos65.vagrantup.com',
               'received_from': '127.0.0.1:61430',
               'source_module': 'TcpServer'}}
```

This data is stored in a separate field to make it easier to drop it prior to store it in some backend.

```
# Print out some stats every 10 seconds.
- misc.SimpleStats:
   interval: 10
```

Prints out some simple stats every interval seconds.

```
# Extract the syslog prival from events received via syslog.
- parser.SyslogPrivalParser:
   source_field: syslog_prival
```

Parses syslog prival values to human readable ones based on RFC5424.

```
# Add a timestamp field.
- parser.AddDateTime:
   format: '%Y-%m-%dT%H:%M:%S.%f'
   target_field: "@timestamp"
```

Adds a timestamp field to the event. When you want to use kibana to view your event data, this field is required.

```
# Add geo info based on the lookup_fields. The first field in <source_fields> that␣
↪yields a result from geoip will be used.
- parser.AddGeoInfo:
   geoip_dat_path: /usr/share/GeoIP/GeoLiteCity.dat
   source_fields: [x_forwarded_for, remote_ip]
   geo_info_fields: ['latitude', 'longitude', 'country_code']
```

Adds geo information fields to the event based on ip addresses found in source_fields. The first ip address in source_fields that yields a result will be used.

```
# Nginx logs request time in seconds with milliseconds as float. Apache logs␣
↪microseconds as int.
# At least cast nginx to integer.
- parser.Math:
   filter: if $(server_type) == "nginx"
   target_field: request_time
   function: float($(request_time)) * 1000
```

As it says in the comment. Nginx and apache use different time formats for the request time field. This module lets you adjust the field to accommodate for that. Also an input filter is used here. Only matching events will be modified by this module.

```
# Map field values of <source_field> to values in <map>.
- modifier.Field:
   filter: if $(http_status)
   action: map
   source_field: http_status
   map: {100: 'Continue', 200: 'OK', 301: 'Moved Permanently', 302: 'Found', 304:
↪'Not Modified', 400: 'Bad Request', 401: 'Unauthorized', 403: 'Forbidden', 404:
↪'Not Found', 500: 'Internal Server Error', 502: 'Bad Gateway'}
```

This module shows how you can map event fields to new values. In this example numeric http status codes are mapped to human readable values.

```
# Kibana's 'bettermap' panel needs an array of floats in order to plot events on map.
- modifier.Field:
   filter: if $(latitude)
   action: merge
   source_fields: [longitude, latitude]
   target_field: geoip
```

Kibanas bettermap module expects the geodata to be found in one single field. With this module the fields longitude and latitude are merged into the geoip field.

```
# Extarct some fields from the user agent data.
- parser.UserAgent:
   source_fields: user_agent
   target_field: user_agent_info
```

Extract user agent information from the user_agent field. This module will set fields like user_agent_info.bot, user_agent_info.browser.name etc.

```
# Parse the url into its components.
- parser.Url:
   source_field: uri
   target_field: uri_parsed
   parse_querystring: True
   querystring_target_field: params
```

Extract details from the uri field. This module will set fields like uri_parsed.scheme, uri_parsed.path, uri_parsed.query etc.

```
# Store events in elastic search.
- output.ElasticSearch:
   nodes: [localhost]
   store_interval_in_secs: 5
```

Send the received events to elasticsearch servers. nodes will set the nodes to connect to.

```
- output.StdOut
```

Events received by this module will be printed out to stdout. The RegexParser module was configured to send unmatched events to this module.

The different modules can be combined in any order.

To run LumberMill you will need Python 2.5+. For better performance I recommend running LumberMill with pypy. Tested with pypy-2.0.2, pypy-2.2.1, pypy-2.3 and pypy-2.4. For IPC ZeroMq is used instead of the default multiprocessing.Queue. This resulted in nearly 3 times of the performance with multiprocessing.Queue.

1.7 Configuration basics

The configuration is stored in a yaml formatted file. Each module configuration follows the same pattern:

```
- category.SomeModuleName:
    id: AliasModuleName                    # <default: ""; type: string; is:␣
↪optional>
    filter: if $(cache_status) == "-"
    add_fields: {'my_new_field': 'my_new_value'}
    delete_fields: ['drop_this_field', 'drop_that_field']
    event_type: my_custom_type
    receivers:
     - ModuleName
     - ModuleAlias:
         filter: if $('event_type') == 'httpd_access_log'
```

- module: specifies the module name and maps to the class name of the module.
- id: use to set an alias name if you run more than just one instance of a module.
- filter: apply a filter to incoming events. Only matching events will be handled by this module.
- add_fields: if the event is handled by the module add this fields to the event.
- delete_fields: if the event is handled by the module delete this fields from the event.
- event_type: if the event is handled by the module set event_type to this value.
- receivers: ModuleName or id of the receiving modules. If a filter is provided, only matching events will be send to receiver. If no receivers are configured, the next module in config will be the default receiver.

For modules that support the storage of intermediate values in redis: * configuration['redis-client']: name of the redis client as set in the configuration. * configuration['redis-key']: key used to store the data in redis. * configuration['redis-ttl']: ttl of the stored data in redis.

For configuration details of each module refer to its docstring.

1.8 Event field notation

The following examples refer to this event data:

```
{'bytes_send': '3395',
 'data': '192.168.2.20 - - [28/Jul/2006:10:27:10 -0300] "GET /wiki/Monty_Python/?
↪spanish=inquisition HTTP/1.0" 200 3395\n',
 'datetime': '28/Jul/2006:10:27:10 -0300',
 'lumbermill': {
                'event_id': '715bd321b1016a442bf046682722c78e',
                'event_type': 'httpd_access_log',
                "received_from": '127.0.0.1',
                "source_module": 'StdIn',
  },
 'http_status': '200',
 'identd': '-',
 'remote_ip': '192.168.2.20',
 'url': 'GET /wiki/Monty_Python/?spanish=inquisition HTTP/1.0',
 'fields': ['nobody', 'expects', 'the'],
 'params':  { u'spanish': [u'inquisition']},
 'user': '-'}
```

1.8.1 Notation in configuration fields like source_field or target_field

Just use the field name. If referring to a nested dict or a list, use dots:

```
- parser.Regex:
    source_field: fields.2

- parser.Regex:
    source_field: params.spanish
```

1.8.2 Notation in strings

Use $(variable_name) notation. If referring to a nested dict or a list, use dots:

```
- output.ElasticSearch:
    index_name: 1perftests
    doc_id: $(fields.0)-$(params.spanish.0)
```

1.8.3 Notation in module filters

Use $(variable_name) notation. If referring to a nested dict, use dots:

```
- output.StdOut:
    filter: if $(fields.0) == "nobody" and $(params.spanish.0) == 'inquisition'
```

Filters

Modules can have an input filter:

```
- output.StdOut:
    filter: if $(remote_ip) == '192.168.2.20' and re.match('^GET', $(url))
```

Modules can have an output filter:

```
- parser.Regex:
    ...
    receivers:
      - output.StdOut:
          filter: if $(remote_ip) == '192.168.2.20' and $(hostname).startswith("www.")
```

1.9 A rough sketch for using LumberMill with syslog-ng

Send e.g. apache access logs to syslog (/etc/httpd/conf/httpd.conf):

```
...
CustomLog "| /usr/bin/logger -p local1.info -t apache2" common
...
```

Configure the linux syslog-ng service to send data to a tcp address (/etc/syslog-ng.conf):

```
...
destination d_lumbermill { tcp( localhost port(5151) ); };
filter f_httpd_access { facility(local1); };
log { source(s_sys); filter(f_httpd_access); destination(d_lumbermill); flags(final);}
↪;
...
```

Configure LumberMill to listen on localhost 5151(./conf/lumbermill.conf):

```
...
- input.Tcp:
    interface: localhost
    port: 5151
...
```

Work in progress.

CHAPTER 2

Input modules

2.1 Beats

Reads data from elastic beats client, i.e. filebeats, and sends it to its outputs.

interface: Ipaddress to listen on.
port: Port to listen on.
timeout: Sockettimeout in seconds.
tls: Use tls or not.
key: Path to tls key file.
cert: Path to tls cert file.
cacert: Path to ca cert file.
tls_proto: Set TLS protocol version.
max_buffer_size: Max kilobytes to in receiving buffer.

Configuration template:

```
- input.Beats:
    interface:                      # <default: ''; type: string; is: optional>
    port:                           # <default: 5151; type: integer; is: optional>
    timeout:                        # <default: None; type: None||integer; is:␣
↪optional>
    tls:                            # <default: False; type: boolean; is: optional>
    key:                            # <default: False; type: boolean||string; is:␣
↪required if tls is True else optional>
    cert:                           # <default: False; type: boolean||string; is:␣
↪required if tls is True else optional>
    cacert:                         # <default: False; type: boolean||string; is:␣
↪optional>
    tls_proto:                      # <default: 'TLSv1'; type: string; values: ['TLSv1
↪', 'TLSv1_1', 'TLSv1_2']; is: optional>
```

(continues on next page)

(continued from previous page)

```
   max_buffer_size:                 # <default: 10240; type: integer; is: optional>
   receivers:
    - NextModule
```

2.2 ElasticSearch

Get documents from ElasticSearch.

The elasticsearch module takes care of discovering all nodes of the elasticsearch cluster. Requests will the be loadbalanced via round robin.

query: The query to be executed, in json format.
search_type: The default search type just will return all found documents. If set to 'scan' it will return 'batch_size' number of found documents, emit these as new events and then continue until all documents have been sent.
field_mappings: Which fields from the result document to add to the new event.
If set to 'all' the whole document will be sent unchanged.
If a list is provided, these fields will be copied to the new event with the same field name.
If a dictionary is provided, these fields will be copied to the new event with a new field name.
E.g. if you want "_source.data" to be copied into the events "data" field, use a mapping like:
"{'_source.data': 'data'}.
For nested values use the dot syntax as described in:
http://lumbermill.readthedocs.org/en/latest/introduction.html#event-field-notation
nodes: Configures the elasticsearch nodes.
read_timeout: Set number of seconds to wait until requests to elasticsearch will time out.
connection_type: One of: 'thrift', 'http'.
http_auth: 'user:password'.
use_ssl: One of: True, False.
index_name: Sets the index name. Timepatterns like %Y.%m.%d are allowed here.
sniff_on_start: The client can be configured to inspect the cluster state to get a list of nodes upon startup. Might cause problems on hosts with multiple interfaces. If connections fail, try to deactivate this.
sniff_on_connection_fail: The client can be configured to inspect the cluster state to get a list of nodes upon failure. Might cause problems on hosts with multiple interfaces. If connections fail, try to deactivate this.
query_interval_in_secs: Get data to es in x seconds intervals. NOT YET IMPLEMENTED!!

Configuration template:

```
- input.ElasticSearch:
   query:                           # <default: '{"query": {"match_all": {}}}'; type:␣
↪string; is: optional>
   search_type:                     # <default: 'normal'; type: string; is: optional;␣
↪values: ['normal', 'scan']>
   batch_size:                      # <default: 1000; type: integer; is: optional>
   field_mappings:                  # <default: 'all'; type: string||list||dict; is:␣
↪optional;>
   nodes:                           # <type: string||list; is: required>
```

(continues on next page)

(continued from previous page)

```
   read_timeout:                    # <default: 10; type: integer; is: optional>
   connection_type:                 # <default: 'urllib3'; type: string; values: [
↪'urllib3', 'requests']; is: optional>
   http_auth:                       # <default: None; type: None||string; is:␣
↪optional>
   use_ssl:                         # <default: False; type: boolean; is: optional>
   index_name:                      # <default: 'lumbermill-%Y.%m.%d'; type: string;␣
↪is: optional>
   sniff_on_start:                  # <default: True; type: boolean; is: optional>
   sniff_on_connection_fail:        # <default: True; type: boolean; is: optional>
   query_interval_in_secs:          # <default: 5; type: integer; is: optional>
   receivers:
    - NextModule
```

2.3 File

Read data from files.

This module supports two modes:

- cat: Just cat existing files.
- tail: Follow changes in given files.

paths: An array of paths to scan for files. Can also point to a file directly.
pattern: Pattern the filenames need to match. E.g. '*.pdf', 'article*.xml' etc.
recursive: If set to true, scan paths recursively else only scan current dir.
line_by_line: If set to true, each line in a file will be emitted as single event.

|If set to false, the whole file will be send as single event. |Only relevant for <cat> mode. | **separator**: Line separator. | **mode**: Mode <cat> will just dump out the current content of a file, <tail> will follow file changes. | **sincedb_path**: Path to a sqlite3 db file which stores the file position data since last poll. | **ignore_empty**: If True ignore empty files. | **ignore_truncate**: If True ignore truncation of files. | **sincedb_write_interval**: Number of seconds to pass between update of sincedb data. | **start_position**: Where to start in the file when tailing. | **stat_interval**: Number of seconds to pass before checking for file changes. | **size_limit**: Set maximum file size for files to watch. Files exeeding this limit will be ignored. TOOD!!!

Configuration template:

```
- input.File:
   paths:                           # <type: string||list; is: required>
   pattern:                         # <default: '*'; type: string; is: optional>
   recursive:                       # <default: False; type: boolean; is: optional>
   line_by_line:                    # <default: False; type: boolean; is: optional>
   separator:                       # <default: "\\n"; type: string; is: optional>
   mode:                            # <default: 'cat'; type: string; is: optional;␣
↪values: ['cat', 'tail']>
   sincedb_path:                    # <default: '/tmp/lumbermill_file_input_sqlite.db
↪'; type: string; is: optional;>
   ignore_empty:                    # <default: False; type: boolean; is: optional;>
   ignore_truncate:                 # <default: False; type: boolean; is: optional;>
   sincedb_write_interval:          # <default: 15; type: integer; is: optional;>
```

(continues on next page)

(continued from previous page)

```
   start_position:                    # <default: 'end'; type: string; is: optional;␣
↪values: ['beginning', 'end']>
   stat_interval:                     # <default: 1; type: integer||float; is: optional;
↪>
   tail_lines:                        # <default: False; type: boolean; is: optional;>
   size_limit:                        # <default: None; type: None||integer; is:␣
↪optional;>
   multiline_regex_before:            # <default: None; type: None||integer; is:␣
↪optional;>
   multiline_regex_after:             # <default: None; type: None||integer; is:␣
↪optional;>
   encoding:                          # <default: 'utf_8'; type: string; is: optional;>
   receivers:
    - NextModule
```

2.4 Kafka

Simple kafka input.

Configuration template:

```
- input.Kafka:
   topic:                             # <type: string; is: required>
   brokers:                           # <default: ['localhost:9092']; type: list; is:␣
↪optional>
   client_id:                         # <default: 'kafka.consumer.kafka'; type: string;␣
↪is: optional>
   group_id:                          # <default: None; type: None||string; is:␣
↪optional>
   fetch_min_bytes:                   # <default: 1; type: integer; is: optional>
   auto_offset_reset:                 # <default: 'latest'; type: string; is: optional>
   enable_auto_commit:                # <default: False; type: boolean; is: optional>
   auto_commit_interval_ms:           # <default: 60000; type: integer; is: optional>
   consumer_timeout_ms:               # <default: -1; type: integer; is: optional>
   receivers:
    - NextModule
```

2.5 NmapScanner

Scan network with nmap and emit result as new event.

Configuration template:

```
- input.NmapScanner:
   network:                           # <type: string; is: required>
   netmask:                           # <default: '/24'; type: string; is: optional>
   ports:                             # <default: None; type: None||string; is:␣
↪optional>
   arguments:                         # <default: '-O -F --osscan-limit'; type: string;␣
↪is: optional>
   interval:                          # <default: 900; type: integer; is: optional>
   receivers:
    - NextModule
```

2.6 RedisChannel

Subscribes to a redis channels and passes incoming events to receivers.

channel: Name of redis channel to subscribe to.
channel_pattern: Channel pattern with wildcards (see: https://redis.io/commands/psubscribe) for channels to subscribe to.
server: Redis server to connect to.
port: Port redis server is listening on.
db: Redis db.
password: Redis password.

Configuration template:

```
- input.RedisChannel:
   channel:                          # <default: False; type: boolean||string; is:␣
↪required if channel_pattern is False else optional>
   channel_pattern:                  # <default: False; type: boolean||string; is:␣
↪required if channel is False else optional>
   server:                           # <default: 'localhost'; type: string; is:␣
↪optional>
   port:                             # <default: 6379; type: integer; is: optional>
   db:                               # <default: 0; type: integer; is: optional>
   password:                         # <default: None; type: None||string; is:␣
↪optional>
   receivers:
    - NextModule
```

2.7 RedisList

Subscribes to a redis channels/lists and passes incoming events to receivers.

lists: Name of redis lists to subscribe to.
server: Redis server to connect to.
port: Port redis server is listening on.
batch_size: Number of events to return from redis list.
db: Redis db.
password: Redis password.
timeout: Timeout in seconds.

Configuration template:

```
- input.RedisList:
   lists:                            # <type: string||list; is: required>
   server:                           # <default: 'localhost'; type: string; is:␣
↪optional>
   port:                             # <default: 6379; type: integer; is: optional>
```

(continues on next page)

(continued from previous page)

```
   batch_size:                          # <default: 1; type: integer; is: optional>
   db:                                  # <default: 0; type: integer; is: optional>
   password:                            # <default: None; type: None||string; is:␣
↪optional>
   timeout:                             # <default: 0; type: integer; is: optional>
   receivers:
    - NextModule
```

2.8 SQS

Read messages from amazon sqs service.

aws_access_key_id: Your AWS id.
aws_secret_access_key: Your AWS password.
region: The region in which to find your sqs service.
queue: Queue name.
attribute_names: A list of attributes that need to be returned along with each message.
message_attribute_names: A list of message attributes that need to be returned.
poll_interval_in_secs: How often should the queue be checked for new messages.
batch_size: Number of messages to retrieve in one call.

Configuration template:

```
- input.SQS:
   aws_access_key_id:                   # <type: string; is: required>
   aws_secret_access_key:               # <type: string; is: required>
   region:                              # <type: string; is: required; values: ['us-east-1
↪', 'us-west-1', 'us-west-2', 'eu-central-1', 'eu-west-1', 'ap-southeast-1', 'ap-
↪southeast-2', 'ap-northeast-1', 'sa-east-1', 'us-gov-west-1', 'cn-north-1']>
   queue:                               # <type: string; is: required>
   attribute_names:                     # <default: ['All']; type: list; is: optional>
   message_attribute_names:             # <default: ['All']; type: list; is: optional>
   poll_interval_in_secs:               # <default: 1; type: integer; is: optional>
   batch_size:                          # <default: 10; type: integer; is: optional>
   receivers:
    - NextModule
```

2.9 Sniffer

Sniff network traffic. Needs root privileges.

Reason for using pcapy as sniffer lib: As Gambolputty is intended to be run with pypy, every module should be compatible with pypy. Creating a raw socket in pypy is no problem but it is (up to now) not possible to bind this socket to a selected interface, e.g. socket.bind((‘lo’, 0)) will throw “error: unknown address family”. With pcapy this problem does not exist.

Dependencies: - pcapy: pypy -m pip install pcapy

Configuration template:

```
- input.Sniffer:
   interface:                          # <default: 'any'; type: None||string; is:␣
↪optional>
   packetfilter:                       # <default: None; type: None||string; is:␣
↪optional>
   promiscous:                         # <default: False; type: boolean; is: optional>
   key_value_store:                    # <default: None; type: none||string; is:␣
↪optional>
   receivers:
    - NextModule
```

2.10 Spam

Emits events as fast as possible.

Use this module to load test LumberMill. Also nice for testing your regexes.

The event field can either be a simple string. This string will be used to create a default lumbermill event dict. If you want to provide more custom fields, you can provide a dictionary containing at least a "data" field that should your raw event string.

event: Send custom event data. For single events, use a string or a dict. If a string is provided, the contents will

be put into the events data field. if a dict is provided, the event will be populated with the dict fields. For multiple events, provide a list of stings or dicts. | **sleep**: Time to wait between sending events. | **events_count**: Only send configured number of events. 0 means no limit.

Configuration template:

```
- input.Spam:
   event:                              # <default: ""; type: string||list||dict; is:␣
↪optional>
   sleep:                              # <default: 0; type: int||float; is: optional>
   events_count:                       # <default: 0; type: int; is: optional>
   receivers:
    - NextModule
```

2.11 StdIn

Reads data from stdin and sends it to its output queues.

Configuration template:

```
- input.StdIn:
   multiline:                          # <default: False; type: boolean; is: optional>
   stream_end_signal:                  # <default: False; type: boolean||string; is:␣
↪optional>
   receivers:
    - NextModule
```

2.12 Tcp

Reads data from tcp socket and sends it to its outputs. Should be the best choice perfomancewise if you are on Linux and are running with multiple workers.

interface: Ipaddress to listen on.
port: Port to listen on.
timeout: Sockettimeout in seconds.
tls: Use tls or not.
key: Path to tls key file.
cert: Path to tls cert file.
cacert: Path to ca cert file.
tls_proto: Set TLS protocol version.
mode: Receive mode, line or stream.
simple_separator: If mode is line, set separator between lines.
regex_separator: If mode is line, set separator between lines. Here regex can be used. The result includes the data that matches the regex.
chunksize: If mode is stream, set chunksize in bytes to read from stream.
max_buffer_size: Max kilobytes to in receiving buffer.

Configuration template:

```
- input.Tcp:
   interface:                       # <default: ''; type: string; is: optional>
   port:                            # <default: 5151; type: integer; is: optional>
   timeout:                         # <default: None; type: None||integer; is:␣
↪optional>
   tls:                             # <default: False; type: boolean; is: optional>
   key:                             # <default: False; type: boolean||string; is:␣
↪required if tls is True else optional>
   cert:                            # <default: False; type: boolean||string; is:␣
↪required if tls is True else optional>
   cacert:                          # <default: False; type: boolean||string; is:␣
↪optional>
   tls_proto:                       # <default: 'TLSv1'; type: string; values: ['TLSv1
↪', 'TLSv1_1', 'TLSv1_2']; is: optional>
   mode:                            # <default: 'line'; type: string; values: ['line',
↪ 'stream']; is: optional>
   simple_separator:                # <default: '\n'; type: string; is: optional>
   regex_separator:                 # <default: None; type: None||string; is:␣
↪optional>
   chunksize:                       # <default: 16384; type: integer; is: optional>
   max_buffer_size:                 # <default: 10240; type: integer; is: optional>
   receivers:
    - NextModule
```

2.13 Udp

Reads data from udp socket and sends it to its output queues.

interface: Ipaddress to listen on.
port: Port to listen on.
timeout: Sockettimeout in seconds.

Configuration template:

```
- input.Udp:
   interface:                          # <default: '0.0.0.0'; type: string; is: optional>
   port:                               # <default: 5151; type: integer; is: optional>
   timeout:                            # <default: None; type: None||integer; is:␣
↪optional>
   receivers:
    - NextModule
```

2.14 UnixSocket

Reads data from an unix socket and sends it to its output queues.

Configuration template:

```
- input.UnixSocket:
   path_to_socket:                     # <type: string; is: required>
   receivers:
    - NextModule
```

ZeroMQ —

Read events from a zeromq.

mode: Whether to run a server or client.
address: Address to connect to. Pattern: hostname:port. If mode is server, this sets the addresses to listen on.
pattern: One of 'pull', 'sub'.
hwm: Highwatermark for sending/receiving socket.

Configuration template:

```
- input.ZeroMQ:
   mode:                               # <default: 'server'; type: string; values: [
↪'server', 'client']; is: optional>
   address:                            # <default: '*:5570'; type: string; is: optional>
   pattern:                            # <default: 'pull'; type: string; values: ['pull',
↪ 'sub']; is: optional>
   topic:                              # <default: ''; type: string; is: optional>
   hwm:                                # <default: None; type: None||integer; is:␣
↪optional>
   receivers:
    - NextModule
```

2.15 ZmqTornado

Read events from a zeromq.

mode: Whether to run a server or client.
address: Address to connect to. Pattern: hostname:port. If mode is server, this sets the addresses to listen on.
pattern: One of 'pull', 'sub'.
hwm: Highwatermark for sending/receiving socket.
separator: When using the sub pattern, messages can have a topic. Set separator to split message from topic.

Configuration template:

```
- input.ZmqTornado:
   mode:                            # <default: 'server'; type: string; values: [
↪'server', 'client']; is: optional>
   address:                         # <default: '*:5570'; type: string; is: optional>
   pattern:                         # <default: 'pull'; type: string; values: ['pull',
↪ 'sub']; is: optional>
   topic:                           # <default: ''; type: string; is: optional>
   separator:                       # <default: None; type: None||string; is:␣
↪optional>
   hwm:                             # <default: None; type: None||integer; is:␣
↪optional>
   receivers:
    - NextModule
```

CHAPTER 3

Parser modules

3.1 Base64

This module will let you en/decode base64 data.

source_fields: Input fields to split. Can be a single field or a list of fields.
target_fields: event field to be filled with the new data.

Configuration template:

```
- parser.Line:
    action:                         # <default: 'decode'; type: string; values: [
↪'decode','encode']; is: optional>
    source_field:                   # <default: 'data'; type: string||list; is:␣
↪optional>
    target_field:                   # <default: 'data'; type:string; is: optional>
    keep_original:                  # <default: False; type: boolean; is: optional>
    receivers:
     - NextModule
```

3.2 Collectd

Parse collectd binary protocol data.

This module can receive binary data from the collectd network plugin.

Decode: It will parse the collectd binary data and create or replace fields in the internal data dictionary with the corresponding collectd data. Encode: Encode selected fields or all to collectd binary protocol.

Configuration template:

```
- parser.Collectd:
   action:                          # <default: 'decode'; type: string; values: [
↪'decode','encode']; is: optional>
   source_fields:                   # <default: 'data'; type: string||list; is:␣
↪optional>
   target_field:                    # <default: None; type: None||string; is:␣
↪optional>
   keep_original:                   # <default: False; type: boolean; is: optional>
   receivers:
    - NextModule
```

3.3 Csv

Parse a string as csv data.

It will parse the csv and create or replace fields in the internal data dictionary with the corresponding csv fields.

source_field: Field that contains the csv data.
escapechar: Char used to escape special characters.
skipinitialspace: When True, whitespace immediately following the delimiter is ignored. The default is False.
quotechar: A one-character string used to quote fields containing special characters, such as the delimiter or quotechar, or which contain new-line characters.
delimiter: A one-character string used to separate fields.
fieldnames: Fieldnames to be used for the extracted csv data.

Configuration template:

```
- parser.Csv:
   source_field:                    # <default: 'data'; type: string; is: optional>
   escapechar:                      # <default: '\'; type: string; is: optional>
   skipinitialspace:                # <default: False; type: boolean; is: optional>
   quotechar:                       # <default: '"'; type: string; is: optional>
   delimiter:                       # <default: '|'; type: string; is: optional>
   fieldnames:                      # <type: list; is: required>
   receivers:
    - NextModule
```

3.4 DateTime

Parse a string to a time object an back again.

For date patterns see: https://docs.python.org/2/library/datetime.html#strftime-strptime-behavior

Configuration template:

```
- parser.DateTime:
   source_field:                    # <type: string; is: required>
   source_date_pattern:             # <type: string; is: required>
   source_timezone:                 # <default: 'utc'; type: string; is: optional>
```

(continues on next page)

(continued from previous page)

```
   target_field:                    # <default: None; type: None||string; is:␣
↪optional>
   target_date_pattern:             # <type: string; is: required>
   target_timezone:                 # <default: 'utc'; type: string; is: optional>
   receivers:
    - NextModule
```

3.5 DomainName

Parse fqdn to top level domain and subdomain parts.

A string like:

“http://some.subdomain.google.co.uk”

will produce this dictionary:

‘domain_name_info’: { ‘tld’: ‘google.co.uk’, ‘domain’: ‘google’, ‘suffix’: ‘co.uk’, ‘subdomain’: ‘some.subdomain’ }

source_field: Input field to parse.
target_field: Field to update with parsed info fields.

Configuration template:

```
- parser.DomainName:
   source_field:                    # <type: string||list; is: required>
   target_field:                    # <default: 'domain_name_info'; type:string; is:␣
↪optional>
   receivers:
    - NextModule
```

3.6 Inflate

Inflate any field with supported compression codecs.

It will take the source fields and decompress them with the configured codecs. At the moment only gzip an zlib are supported.

source_fields: Single field or list of fields to decompress.
target_fields: Single field or list of fields to fill with decompressed data.
If not provided, contents of source_fields will be replaced.
compression: Compression lib to use for decompression.

Configuration template:

```
- parser.Inflate:
   source_fields:                    # <default: 'data'; type: string||list; is:␣
↪optional>
   target_fields:                    # <default: None; type: None||string||list; is:␣
↪optional>
   compression:                      # <default: 'gzip'; type: string; is: optional;␣
↪values: ['gzip', 'zlib']>
   receivers:
    - NextModule
```

3.7 Json

Json codec.

Decode: It will parse the json data in source fields and create or replace fields in the internal data dictionary with the corresponding json fields.

Encode: It will build a new list of source fields and create json of this list.

|**action**: Either encode or decode data. |**source_fields**: Input fields for de/encode. If encoding, you can set this field to 'all' to encode the complete event dict. |**target_field**: Target field for de/encode result. If decoding and target is not set, the event dict itself will be updated with decoded fields. |**keep_original**: Switch to keep or drop the original fields used in de/encoding from the event dict.

Configuration template:

```
- parser.Json:
   action:                           # <default: 'decode'; type: string; values: [
↪'decode','encode']; is: optional>
   source_fields:                    # <default: 'data'; type: string||list; is:␣
↪optional>
   target_field:                     # <default: None; type: None||string; is:␣
↪optional>
   keep_original:                    # <default: False; type: boolean; is: optional>
   receivers:
    - NextModule
```

3.8 Line

Line parser.

Decode: Will split the data in source fields and emit parts as new events. So if e.g. data field contains: message-a|message-b|message-c you can split this field by "|" and three new events will be created with message-a, message-b and message-c as payload.

The original event will be discarded.

source_field: Input field to split.
seperator: Char used as line seperator.
target_field: event field to be filled with the new data.

Configuration template:

```
- parser.Line:
   source_field:                    # <default: 'data'; type: string||list; is:␣
↪optional>
   seperator:                       # <default: '\n'; type: string; is: optional>
   target_field:                    # <default: 'data'; type:string; is: optional>
   keep_original:                   # <default: False; type: boolean; is: optional>
   receivers:
    - NextModule
```

3.9 MsgPack

Decode: It will parse the msgpack data and create or replace fields in the internal data dictionary with the corresponding json fields. Encode: Encode selected fields or all to msgpack format.

Configuration template:

```
- parser.MsgPack:
   action:                          # <default: 'decode'; type: string; values: [
↪'decode','encode']; is: optional>
   mode:                            # <default: 'line'; type: string; values: ['line',
↪'stream']; is: optional>
   source_fields:                   # <default: 'data'; type: string||list; is:␣
↪optional>
   target_field:                    # <default: None; type: None||string; is:␣
↪optional>
   keep_original:                   # <default: False; type: boolean; is: optional>
   receivers:
    - NextModule
```

3.10 NetFlow

Netflow parser

Decode netflow packets.

source_field: Input field to decode.
target_field: Event field to be filled with the new data.

Configuration template:

```
- parser.NetFlow:
   source_field:                    # <default: 'data'; type: string; is: optional>
   target_field:                    # <default: 'data'; type: string; is: optional>
   keep_original:                   # <default: False; type: boolean; is: optional>
   receivers:
    - NextModule
```

3.11 Regex

Parse a string by named regular expressions.

If regex matches, fields in the data dictionary will be set as defined in the named regular expression. Additionally the field "lumbermill.event_type" will be set containing the name of the regex. In the example below this would be "httpd_access_log".

It is also possible to define multiple regexes with the same name. This allows for different log patterns for the same log type, e.g. apache access logs and nginx access logs.

source_field: Field to apply the regex to.
mark_unmatched_as: Set <lumbermill.event_type> to this value if regex did not match.
break_on_match: Stop applying regex patterns after first match.
hot_rules_first: Apply regex patterns based on their hit count.

Configuration template:

```
- parser.Regex:
   source_field:                    # <default: 'data'; type: string; is: optional>
   mark_unmatched_as:               # <default: 'Unknown'; type: string; is: optional>
   break_on_match:                  # <default: True; type: boolean; is: optional>
   hot_rules_first:                 # <default: True; type: boolean; is: optional>
   field_extraction_patterns:       # <type: list; is: required>
    - httpd_access_log: ['(?P<httpd_access_log>.*)', 're.MULTILINE | re.DOTALL',
↪'findall']
   receivers:
    - NextModule
```

3.12 SyslogPrival

It will parse the source field in the event dictionary for the default severity and facility fields (RFC5424, http://tools.ietf.org/html/rfc5424). The source field must contain the prival with the pattern: "d+"

Numerical Facility Code

0 kernel messages 1 user-level messages 2 mail system 3 system daemons 4 security/authorization messages 5 messages generated internally by syslogd 6 line printer subsystem 7 network news subsystem 8 UUCP subsystem 9 clock daemon 10 security/authorization messages 11 FTP daemon 12 NTP subsystem 13 log audit 14 log alert 15 clock daemon (note 2) 16 local use 0 (local0) 17 local use 1 (local1) 18 local use 2 (local2) 19 local use 3 (local3) 20 local use 4 (local4) 21 local use 5 (local5) 22 local use 6 (local6) 23 local use 7 (local7)

Numerical Severity Code

0 Emergency: system is unusable 1 Alert: action must be taken immediately 2 Critical: critical conditions 3 Error: error conditions 4 Warning: warning conditions 5 Notice: normal but significant condition 6 Informational: informational messages 7 Debug: debug-level messages

Configuration template:

```
- parser.SyslogPrival:
   source_field:                    # <default: 'syslog_prival'; type: string; is:␣
↪optional>
```

(continues on next page)

(continued from previous page)

```
   map_values: False                # <default: True; type: boolean; is: optional>
   facility_mappings:               # <default: {}; type: dictionary; is: optional>
   severity_mappings:               # <default: {}; type: dictionary; is: optional>
   receivers:
    - NextModule
```

3.13 Url

Urlencode or decode an event field and extract url parameters.

If no target field is supplied, it will overrite the source field with the parsed date dictionary.

action: Either encode or decode data.
source_field: Event field to en/decode.
target_field: Event field to update with en/decode result. If not set source will be replaced.
parse_querystring: Parse url for query parameters and extract them.
querystring_target_field: Event field to update with url parameters.
querystring_prefix: Prefix string to prepend to url parameter keys.

Configuration template:

```
- parser.Url:
   action:                          # <default: 'decode'; type: string; values: [
↪'decode','encode']; is: optional>
   source_field:                    # <type: string; is: required>
   target_field:                    # <default: None; type: None||string; is:␣
↪optional>
   parse_querystring:               # <default: False; type: boolean; is: optional>
   querystring_target_field:        # <default: None; type: None||string; is:␣
↪optional>
   querystring_prefix:              # <default: None; type: None||string; is:␣
↪optional>
   receivers:
    - NextModule
```

3.14 UserAgent

Parse http user agent string

A string like:

“Mozilla/5.0 (Linux; U; Android 2.3.5; en-in; HTC_DesireS_S510e Build/GRJ90) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1”

will produce this dictionary:

‘user_agent_info’: { ‘device’: { ‘family’: u’HTC DesireS’}, ‘os’: { ‘family’: ‘Android’, ‘major’: ‘2’, ‘minor’: ‘3’, ‘patch’: ‘5’, ‘patch_minor’: None}, ‘user_agent’: { ‘family’: ‘Android’, ‘major’: ‘2’, ‘minor’: ‘3’, ‘patch’: ‘5’}}}

source_fields: Input field to parse.
target_field: field to update with parsed info fields.

Configuration template:

```
- parser.UserAgent:
   source_fields:                   # <type: string||list; is: required>
   target_field:                    # <default: 'user_agent_info'; type:string; is:␣
↪optional>
   receivers:
    - NextModule
```

3.15 XPath

Parse an xml string via xpath.

This module supports the storage of the results in an redis db. If redis-client is set, it will first try to retrieve the result from redis via the key setting. If that fails, it will execute the xpath query and store the result in redis.

Configuration template:

```
- parser.XPath:
   source_field:                    # <type: string; is: required>
   target_field:                    # <default: "gambolputty_xpath"; type: string;␣
↪is: optional>
   query:                           # <type: string; is: required>
   redis_store:                     # <default: None; type: None||string; is:␣
↪optional>
   redis_key:                       # <default: None; type: None||string; is:␣
↪optional if redis_store is None else required>
   redis_ttl:                       # <default: 60; type: integer; is: optional>
   receivers:
    - NextModule
```

CHAPTER 4

Modifier modules

4.1 AddDateTime

Add a field with a datetime.

If source_fields is not set, datetime will be based on current time. If source_fields is set, event will be searched for each source_field. If found, all source_formats will be applied, to parse the date. First successful conversion will win.

Configuration template:

```
- modifier.AddDateTime:
   source_fields:                   # <default: None; type: None||list; is: optional>
   source_formats:                  # <default: None; type: None||list; is: required␣
↪if source_fields is not None else optional>
   target_field:                    # <default: '@timestamp'; type: string; is:␣
↪optional>
   target_format:                   # <default: '%Y-%m-%dT%H:%M:%S'; type: string;␣
↪is: optional>
   receivers:
    - NextModule
```

4.2 AddDnsLookup

Add dns info for selected fields. The dns servers used are the ones configured for the system LumberMill is running on.

action: Either resolve or revers.
source_field: Source field to use for (reverse) lookups.
target_field: Target field to store result of lookup. If none is provided, the source field will be replaced.
nameservers: List of nameservers to use. If not provided, the system default servers will be used.

timeout: Timeout for lookups in seconds.

Configuration template:

```
- modifier.AddDnsLookup:
   action:                          # <default: 'resolve'; type: string; is: optional;
↪ values: ['resolve', 'reverse']>
   source_field:                    # <default: None; type: string; is: required>
   target_field:                    # <default: None; type: None||string; is:␣
↪optional>
   nameservers:                     # <default: None; type: None||string||list; is:␣
↪optional>
   timeout:                         # <default: 1; type: integer; is: optional>
   receivers:
      - NextModule
```

4.3 AddGeoInfo

Add country_code and longitude-latitude fields based on a geoip lookup for a given ip address.

Here an example of fields that the module provides: {'city': 'Hanover', 'region_name': '06', 'area_code': 0, 'time_zone': 'Europe/Berlin', 'dma_code': 0, 'metro_code': None, 'country_code3': 'DEU', 'latitude': 52.36670000000001, 'postal_code': '', 'longitude': 9.716700000000003, 'country_code': 'DE', 'country_name': 'Germany', 'continent': 'EU'}

geoip_dat_path: path to maxmind geoip2 database file.
asn_dat_path: path to maxmind ASN database file.
maxmind_locals: List of locale codes. See:
https://github.com/maxmind/GeoIP2-python/blob/master/geoip2/database.py#L59
maxmind_mode: See: https://github.com/maxmind/GeoIP2-python/blob/master/geoip2/database.py#L71
source_fields: list of fields to use for lookup. The first list entry that produces a hit is used.
target: field to populate with the geoip data. If none is provided, the field will be added directly to the event.
maxmind_info_fields: fields to add. Available fields:

- city
- postal_code
- country_name
- country_code
- continent_code
- continent
- area_code
- region_name
- longitude
- latitude
- longlat
- time_zone
- metro_code
- autonomous_system_number
- autonomous_system_organization

Configuration template:

```
- AddGeoInfo:
   geoip_dat_path:                  # <default: './assets/maxmind/GeoLite2-City.mmdb';
↪ type: string; is: optional>
   asn_dat_path:                    # <default: './assets/maxmind/GeoLite2-ASN.mmdb';␣
↪type: string; is: optional>
   maxmind_locals:                  # <default: ['en']; type: list; is: optional>
   maxmind_mode:                    # <default: 'MODE_AUTO'; type: string; is:␣
↪optional; values: ['MODE_MMAP_EXT', 'MODE_MMAP', 'MODE_FILE', 'MODE_MEMORY', 'MODE_
↪AUTO']>
   maxmind_info_fields:             # <default: None; type: None||list; is: optional>
   source_fields:                   # <default: ["x_forwarded_for", "remote_ip"];␣
↪type: list; is: optional>
   target_field:                    # <default: "geo_info"; type: string; is:␣
↪optional>
   receivers:
    - NextModule
```

4.4 DropEvent

Drop all events received by this module.

This module is intended to be used with an activated filter.

Configuration template:

```
- modifier.DropEvent
```

4.5 ExecPython

Execute python code.

To make sure that the yaml parser keeps the tabs in the source code, ensure that the code is preceded by a comment. E.g.:

- ExecPython:

source: | # Useless comment... try: imported = math except NameError: import math event['request_time'] = math.ceil(event['request_time'] * 1000)

imports: Modules to import, e.g. re, math etc.
code: Code to execute.
debug: Set to True to output the string that will be executed.

Configuration template:

```
- modifier.ExecPython:
   imports:                         # <default: []; type: list; is: optional>
   source:                          # <type: string; is: required>
   debug:                           # <default: False; type: boolean; is: optional>
```

(continues on next page)

(continued from previous page)

```
   receivers:
    - NextModule
```

4.6 Facet

Collect different values of one field over a defined period of time and pass all encountered variations on as new event after period is expired.

The "add_event_fields" configuration will copy the configured event fields into the "other_event_fields" list.

The event emitted by this module will be of type: "facet" and will have "facet_field", "facet_count", "facets" and "other_event_fields" fields set.

This module supports the storage of the facet info in an backend db (At the moment this only works for a redis backend. This offers the possibility of using this module across multiple instances of LumberMill.

source_field: Field to be scanned for unique values.
group_by: Field to relate the variations to, e.g. ip address.
backend: Name of a key::value store plugin. When running multiple instances of gp this backend can be used to synchronize events across multiple instances.
backend_ttl: Time to live for backend entries. Should be greater than interval.
add_event_fields: Fields to add from the original event to the facet event.
interval: Number of seconds to until all encountered values of source_field will be send as new facet event.

Configuration template:

```
- modifier.Facet:
   source_field:                    # <type:string; is: required>
   group_by:                        # <type:string; is: required>
   backend:                         # <default: None; type: None||string; is:␣
↪required>
   backend_ttl:                     # <default: 60; type: integer; is: optional>
   add_event_fields:                # <default: []; type: list; is: optional>
   interval:                        # <default: 5; type: float||integer; is: optional>
   receivers:
    - NextModule
```

4.7 HttpRequest

Issue an arbitrary http request and store the response in a configured field.

If the <interval> value is set, this module will execute the configured request every <interval> seconds and emits the result in a new event.

This module supports the storage of the responses in an redis db. If redis_store is set, it will first try to retrieve the response from redis via the key setting. If that fails, it will execute the http request and store the result in redis.

url: The url to grab. Can also contain templated values for dynamic replacement with event data.

socket_timeout: The socket timeout in seconds after which a request is considered failed.
get_response_header: Also get resonse headers.
target_field: Specifies the name of the field to store the retrieved data in.
interval: Number of seconds to wait before calling <url> again.
redis_store: Redis address to cache crawling results.
redis_key: The key to use for storage in redis.
redis_ttl: TTL for data in redis.

Configuration template:

```
- modifier.HttpRequest:
   url:                              # <type: string; is: required>
   socket_timeout:                   # <default: 25; type: integer; is: optional>
   get_response_header:              # <default: False; type: boolean; is: optional>
   target_field:                     # <default: "gambolputty_http_request"; type:␣
↪string; is: optional>
   interval:                         # <default: None; type: None||float||integer; is:␣
↪optional>
   redis_store:                      # <default: None; type: None||string; is:␣
↪optional>
   redis_key:                        # <default: None; type: None||string; is:␣
↪optional if redis_store is None else required>
   redis_ttl:                        # <default: 60; type: integer; is: optional>
   receivers:
    - NextModule
```

4.8 Math

Execute arbitrary math functions.

Simple example to cast nginx request time (seconds with milliseconds as float) to apache request time (microseconds as int):

- Math:

filter: if $(server_type) == "nginx" target_field: request_time function: int(float($(request_time)) * 1000)

If interval is set, the results of <function> will be collected for the interval time and the final result will be calculated via the <results_function>.

function: the function to be applied to/with the event data.
results_function: if interval is configured, use this function to calculate the final result.
interval: Number of seconds to until.
target_field: event field to store the result in.

Configuration template:

```
- modifier.Math:
   function:                         # <type: string; is: required>
   results_function:                 # <default: None; type: None||string; is:␣
↪optional if interval is None else required>
```

(continues on next page)

(continued from previous page)

```
   interval:                          # <default: None; type: None||float||integer; is:␣
↪optional>
   target_field:                      # <default: None; type: None||string; is:␣
↪optional>
   receivers:
    - NextModule
```

4.9 ModifyFields

Simple module to insert/delete/change field values.

Configuration templates:

```
# Keep all fields listed in source_fields, discard all others.
- modifier.Field:
   action: keep                       # <type: string; is: required>
   source_fields:                     # <type: list; is: required>
   receivers:
    - NextModule

# Discard all fields listed in source_fields.
- modifier.Field:
   action: delete                     # <type: string; is: required>
   source_fields:                     # <type: list; is: required>
   receivers:
    - NextModule

# Concat all fields listed in source_fields.
- modifier.Field:
   action: concat                     # <type: string; is: required>
   source_fields:                     # <type: list; is: required>
   target_field:                      # <type: string; is: required>
   receivers:
    - NextModule

# Insert a new field with "target_field" name and "value" as new value.
- modifier.Field:
   action: insert                     # <type: string; is: required>
   target_field:                      # <type: string; is: required>
   value:                             # <type: string; is: required>
   receivers:
    - NextModule

# Replace field values matching string "old" in data dictionary with "new".
- modifier.Field:
   action: string_replace             # <type: string; is: required>
   source_field:                      # <type: string; is: required>
   old:                               # <type: string; is: required>
   new:                               # <type: string; is: required>
   max:                               # <default: -1; type: integer; is: optional>
   receivers:
    - NextModule

# Replace field values in data dictionary with self.getConfigurationValue['with'].
```

(continues on next page)

(continued from previous page)

```
- modifier.Field:
   action: replace                    # <type: string; is: required>
   source_field:                      # <type: string; is: required>
   regex: ['<[^>]*>', 're.MULTILINE | re.DOTALL'] # <type: list; is: required>
   with:                              # <type: string; is: required>
   receivers:
    - NextModule

# Rename a field.
- modifier.Field:
   action: rename                     # <type: string; is: required>
   source_field:                      # <type: string; is: required>
   target_field:                      # <type: string; is: required>
   receivers:
    - NextModule

# Rename a field by regex.
- modifier.Field:
   action: rename_regex               # <type: string; is: required>
   regex:                             # <type: string; is: required>
   source_field:                      # <default: None; type: None||string; is:␣
↪optional>
   target_field_pattern:              # <type: string; is: required>
   recursive:                         # <default: True; type: boolean; is: optional>
   receivers:
    - NextModule

# Rename a field by replace.
- modifier.Field:
   action: rename_replace             # <type: string; is: required>
   old:                               # <type: string; is: required>
   new:                               # <type: string; is: required>
   source_field:                      # <default: None; type: None||string; is:␣
↪optional>
   recursive:                         # <default: True; type: boolean; is: optional>
   receivers:
    - NextModule

# Map a field value.
- modifier.Field:
   action: map                        # <type: string; is: required>
   source_field:                      # <type: string; is: required>
   map:                               # <type: dictionary; is: required>
   target_field:                      # <default: "$(source_field)_mapped"; type:␣
↪string; is: optional>
   keep_unmappable:                   # <default: False; type: boolean; is: optional>
   receivers:
    - NextModule

# Split source field to target fields based on key value pairs.
- modifier.Field:
   action: key_value                  # <type: string; is: required>
   line_separator:                    # <type: string; is: required>
   kv_separator:                      # <type: string; is: required>
   source_field:                      # <type: list; is: required>
   target_field:                      # <default: None; type: None||string; is:␣
↪optional>
```

(continues on next page)

(continued from previous page)

```
   prefix:                          # <default: None; type: None||string; is:␣
↪optional>
   receivers:
    - NextModule

# Split source field to target fields based on key value pairs using regex.
- modifier.Field:
   action: key_value_regex          # <type: string; is: required>
   regex:                           # <type: string; is: required>
   source_field:                    # <type: list; is: required>
   target_field:                    # <default: None; type: None||string; is:␣
↪optional>
   prefix:                          # <default: None; type: None||string; is:␣
↪optional>
   receivers:
    - NextModule

# Split source field to array at separator.
- modifier.Field:
   action: split                    # <type: string; is: required>
   separator:                       # <type: string; is: required>
   source_field:                    # <type: list; is: required>
   target_field:                    # <default: None; type: None||string; is:␣
↪optional>
   receivers:
    - NextModule

# Merge source fields to target field as list.
- modifier.Field:
   action: merge                    # <type: string; is: required>
   target_field:                    # <type: string; is: reuired>
   source_fields:                   # <type: list; is: required>
   receivers:
    - NextModule

# Merge source field to target field as string.
- modifier.Field:
   action: join                     # <type: string; is: required>
   source_field:                    # <type: string; is: required>
   target_field:                    # <type: string; is: required>
   separator:                       # <default: ","; type: string; is: optional>
   receivers:
    - NextModule

# Cast field values to integer.
- modifier.Field:
   action: cast_to_int              # <type: string; is: required>
   source_fields:                   # <type: list; is: required>
   receivers:
    - NextModule

# Cast field values to float.
- modifier.Field:
   action: cast_to_float            # <type: string; is: required>
   source_fields:                   # <type: list; is: required>
   receivers:
    - NextModule
```

(continues on next page)

(continued from previous page)

```
# Cast field values to string.
- modifier.Field:
   action: cast_to_str              # <type: string; is: required>
   source_fields:                   # <type: list; is: required>
   receivers:
    - NextModule

# Cast field values to boolean.
- modifier.Field:
   action: cast_to_bool             # <type: string; is: required>
   source_fields:                   # <type: list; is: required>
   receivers:
    - NextModule

# Create a hash from a field value.
# If target_fields is provided, it should have the same length as source_fields.
# If target_fields is not provided, source_fields will be replaced with the hashed␣
↪value.
# Hash algorithm can be any of the in hashlib supported algorithms.
- modifier.Field:
   action: hash                     # <type: string; is: required>
   algorithm: sha1                  # <default: "md5"; type: string; is: optional;>
   salt:                            # <default: None; type: None||string; is:␣
↪optional;>
   source_fields:                   # <type: list; is: required>
   target_fields:                   # <default: []; type: list; is: optional>
   receivers:
    - NextModule
```

4.10 MergeEvent

Merge multiple event into a single one.

In most cases, inputs will split an incoming stream at some kind of delimiter to produce events. Sometimes, the delimiter also occurs in the event data itself and splitting here is not desired. To mitigate this problem, this module can merge these fragmented events based on some configurable rules.

Each incoming event will be buffered in a queue identified by <buffer_key>. If a new event arrives and <pattern> does not match for this event, the event will be appended to the buffer. If a new event arrives and <pattern> matches for this event, the buffer will be flushed prior to appending the event. After <flush_interval_in_secs> the buffer will also be flushed. Flushing the buffer will concatenate all contained event data to form one single new event.

buffer_key: key to distinguish between different input streams

buffer_key: A key to correctly group events.

buffer_size: Maximum size of events in buffer. If size is exceeded a flush will be executed.

flush_interval_in_secs: If interval is reached, buffer will be flushed.

pattern: Pattern to match new events. If pattern matches, a flush will be executed prior to appending the event to buffer.

pattern_marks: Set if the pattern marks the start or the end of an event.

If it marks the start of an event and a new event arrives and <pattern> matches, the buffer will be flushed prior appending the event.

If it marks the end of an event and a new event arrives and <pattern> matches, the buffer will be flushed after appending the event.

glue: Join event data with glue as separator.

Configuration template:

```
- modifier.MergeEvent:
   buffer_key:                      # <default: "$(lumbermill.received_from)"; type:␣
↪string; is: optional>
   buffer_size:                     # <default: 100; type: integer; is: optional>
   flush_interval_in_secs:          # <default: 1; type: None||integer; is: required␣
↪if pattern is None else optional>
   pattern:                         # <default: None; type: None||string; is:␣
↪required if flush_interval_in_secs is None else optional>
   pattern_marks:                   # <default: 'EndOfEvent'; type: string; values: [
↪'StartOfEvent', 'EndOfEvent'];  is: optional;>
   match_field:                     # <default: "data"; type: string; is: optional>
   glue:                            # <default: ""; type: string; is: optional>
   receivers:
    - NextModule
```

4.11 Permutate

Creates successive len('target_fields') length permutations of elements in 'source_field'.

To add some context data to each emitted event 'context_data_field' can specify a field containing a dictionary with the values of 'source_field' as keys.

Configuration template:

```
- modifier.Permutate:
   source_field:                    # <type: string; is: required>
   target_fields:                   # <type: list; is: required>
   context_data_field:              # <default: ""; type:string; is: optional>
   context_target_mapping:          # <default: {}; type: dict; is: optional if␣
↪context_data_field == "" else required>
   receivers:
    - NextModule
    - NextModule
```

CHAPTER 5

Misc modules

5.1 EventBuffer

Store received events in a persistent backend until the event was successfully handled. Events, that did not get handled correctly, will be requeued when LumberMill is restarted.

At the moment only RedisStore is supported as backend.

As a technical note: This module is based on pythons garbage collection. If an event is created, a copy of the event is stored in the persistence backend. If it gets garbage collected, the event will be deleted from the backend. When used, this module forces a garbage collection every <gc_interval> seconds. This approach seemed to be the fastest and simplest with a small drawback:

IMPORTANT: It is not absolutely guaranteed, that an event will be collected, thus the event will not be deleted from the backend data. This can cause a limited amount of duplicate events being send to the sinks. With an elasticsearch sink, this should be no problem, as long as your document id stays the same for the same event data. This is also true for the default event_id.

Configuration template:

```
- misc.EventBuffer:
   backend:                          # <default: 'RedisStore'; type: string; is:␣
↪optional>
   gc_interval:                      # <default: 5; type: integer; is: optional>
   key_prefix:                       # <default: "lumbermill:eventbuffer"; type:␣
↪string; is: optional>
```

5.2 Cache

A simple wrapper around the python simplekv module.

It can be used to store results of modules in all simplekv supported backends.

When set, the following options cause RedisStore to use a buffer for setting values. Multiple values are set via the pipe command, which speeds up storage. Still this comes at a price. Buffered values, that have not yet been send to redis, will be lost when LumberMill crashes.

backend: backends supported by [simplekv](http://pythonhosted.org//simplekv/)
store_interval_in_secs: Sending data to redis in x seconds intervals.
batch_size: Sending data to redis if count is above, even if store_interval_in_secs is not reached.
backlog_size: Maximum count of values waiting for transmission. Values above count will be dropped.

Configuration template:

```
- misc.Cache:
   backend:                         # <default: 'DictStore'; type: string; values:[
↪'DictStore', 'RedisStore', 'MemcacheStore']; is: optional>
   server:                          # <default: None; type: None||string; is:␣
↪required if backend in ['RedisStore', 'MemcacheStore'] and cluster is None else␣
↪optional>
   cluster:                         # <default: None; type: None||dictionary; is:␣
↪required if backend == 'RedisStore' and server is None else optional>
   port:                            # <default: 6379; type: integer; is: optional>
   db:                              # <default: 0; type: integer; is: optional>
   password:                        # <default: None; type: None||string; is:␣
↪optional>
   socket_timeout:                  # <default: 10; type: integer; is: optional>
   charset:                         # <default: 'utf-8'; type: string; is: optional>
   errors:                          # <default: 'strict'; type: string; is: optional>
   decode_responses:                # <default: False; type: boolean; is: optional>
   unix_socket_path:                # <default: None; type: None||string; is:␣
↪optional>
   batch_size:                      # <default: None; type: None||integer; is:␣
↪optional>
   store_interval_in_secs:          # <default: None; type: None||integer; is:␣
↪optional>
   backlog_size:                    # <default: 5000; type: integer; is: optional>
```

5.3 SimpleStats

Collect and log some simple lumbermill statistic data.

Use this module if you just need some simple statistics on how many events are passing through lumbermill. Per default, statistics will just be send to stdout.

As a side note: This module inits MultiProcessStatisticCollector. As it uses multiprocessing.Manager().dict() this will start another process. So if you use SimpleStats, you will see workers + 1 processes in the process list.

Configuration template:

```
- misc.SimpleStats:
   interval:                        # <default: 10; type: integer; is: optional>
   event_type_statistics:           # <default: True; type: boolean; is: optional>
   receive_rate_statistics:         # <default: True; type: boolean; is: optional>
   waiting_event_statistics:        # <default: False; type: boolean; is: optional>
   emit_as_event:                   # <default: False; type: boolean; is: optional>
```

5.4 Metrics

Collect metrics data from events.

As a side note: This module inits MultiProcessStatisticCollector. As it uses multiprocessing.Manager().dict() this will start another process. So if you use SimpleStats, you will see workers + 1 processes in the process list.

This module keeps track of the number of times a field occured in an event during interval. So, if you want to count the http_status codes encountered during the last 10s, you would use this configuration:

```
- Mertrics:
    interval: 10
    aggregations:
        - key: http_status_%{vhost}
          value: http_status
```

After interval seconds, an event will be emitted with the following fields (counters are just examples):

Example output:

```
{'data': '',
 'field_name': 'http_status_this.parrot.dead',
 'field_counts': {'200': 5, '301': 10, '400': 5},
 'lumbermill': {'event_id': 'cef34d298fbe8ce4b662251e17b2acfb',
                'event_type': 'metrics',
                'received_from': False,
                'source_module': 'Metrics'}
 'interval': 10,
 'total_count': 20}
```

Same with buckets:

```
- Mertrics:
    interval: 10
    aggregations:
        - key: http_status_%{vhost}
          value: http_status
          buckets:
            - key: 100
              upper: 199
            - key: 200
              upper: 299
            - key: 300
              upper: 399
            - key: 400
              upper: 499
            - key: 500
              upper: 599
    percentiles:
        - key: request_time_%{vhost}
          value: request_time
          percentiles: [50, 75, 95, 99]
```

Example output:

```
{'data': '',
 'field_name': 'http_status_this.parrot.dead',
 'field_counts': {'200': 5, '300': 10, '400': 5},
```

(continues on next page)

(continued from previous page)

```
 'lumbermill': {'event_id': 'cef34d298fbe8ce4b662251e17b2acfb',
                'event_type': 'metrics',
                'received_from': False,
                'source_module': 'Metrics'}
 'interval': 10,
 'total_count': 20}
```

Configuration template:

```
- Metrics:
   interval:                        # <default: 10; type: integer; is: optional>
   aggregations:                    # <default: []; type: list; is: optional>
```

5.5 Tarpit

Send an event into a tarpit before passing it on.

Useful only for testing purposes of threading problems and concurrent access to event data.

Configuration template:

```
- misc.Tarpit:
   delay:                           # <default: 10; type: integer; is: optional>
   receivers:
    - NextModule
```

5.6 Throttle

Throttle event count over a given time period.

key: Identifies events as being the "same". Dynamic notations can be used here.
timeframe: Time window in seconds from first encountered event to last.
min_count: Minimal count of same events to allow event to be passed on.
max_mount: Maximum count of same events before same events will be blocked.
backend: Name of a key::value store plugin. When running multiple instances of gp this backend can be used to synchronize events across multiple instances.
backend_key_prefix: Prefix for the backend key.

Configuration template:

```
- misc.Throttle:
   key:                             # <type:string; is: required>
   timeframe:                       # <default: 600; type: integer; is: optional>
   min_count:                       # <default: 1; type: integer; is: optional>
   max_count:                       # <default: 1; type: integer; is: optional>
   backend:                         # <default: None; type: None||string; is:␣
↪optional>
   backend_key_prefix:              # <default: "lumbermill:throttle"; type: string;␣
↪is: optional>
```

(continues on next page)

(continued from previous page)

```
  receivers:
   - NextModule
```

CHAPTER 6

Output modules

6.1 DevNull

Just discard messages send to this module.

Configuration template:

```
- output.DevNull
```

6.2 ElasticSearch

Store the data dictionary in an elasticsearch index.

The elasticsearch module takes care of discovering all nodes of the elasticsearch cluster. Requests will the be loadbalanced via round robin.

action: Either index or update. If update be sure to provide the correct doc_id.
format: Which event fields to send on, e.g. '$(@timestamp) - $(url) - $(country_code)'.
If not set the whole event dict is send.
nodes: Configures the elasticsearch nodes.
read_timeout: Set number of seconds to wait until requests to elasticsearch will time out.
connection_type: One of: 'thrift', 'http'.
http_auth: 'user:password'.
use_ssl: One of: True, False.
index_name: Sets the index name. Timepatterns like %Y.%m.%d and dynamic values like $(bar) are allowed here.
doc_id: Sets the es document id for the committed event data.
routing: Sets a routing value (@see: http://www.elasticsearch.org/blog/customizing-your-document-routing/)
Timepatterns like %Y.%m.%d are allowed here.
ttl: When set, documents will be automatically deleted after ttl expired.

Can either set time in milliseconds or elasticsearch date format, e.g.: 1d, 15m etc.
This feature needs to be enabled for the index.
@See: http://www.elasticsearch.org/guide/en/elasticsearch/reference/current/mapping-ttl-field.html
sniff_on_start: The client can be configured to inspect the cluster state to get a list of nodes upon startup.
Might cause problems on hosts with multiple interfaces. If connections fail, try to deactivate this.
sniff_on_connection_fail: The client can be configured to inspect the cluster state to get a list of nodes upon failure.
Might cause problems on hosts with multiple interfaces. If connections fail, try to deactivate this.
store_interval_in_secs: Send data to es in x seconds intervals.
batch_size: Sending data to es if event count is above, even if store_interval_in_secs is not reached.
backlog_size: Maximum count of events waiting for transmission. If backlog size is exceeded no new events will be processed.

Configuration template:

```
- output.ElasticSearch:
   action:                          # <default: 'index'; type: string; is: optional;␣
↪values: ['index', 'update']>
   format:                          # <default: None; type: None||string; is:␣
↪optional>
   nodes:                           # <type: string||list; is: required>
   read_timeout:                    # <default: 10; type: integer; is: optional>
   connection_type:                 # <default: 'urllib3'; type: string; values: [
↪'urllib3', 'requests']; is: optional>
   http_auth:                       # <default: None; type: None||string; is:␣
↪optional>
   use_ssl:                         # <default: False; type: boolean; is: optional>
   index_name:                      # <default: 'lumbermill-%Y.%m.%d'; type: string;␣
↪is: optional>
   doc_id:                          # <default: '$(lumbermill.event_id)'; type:␣
↪string; is: optional>
   routing:                         # <default: None; type: None||string; is:␣
↪optional>
   ttl:                             # <default: None; type: None||integer||string;␣
↪is: optional>
   sniff_on_start:                  # <default: False; type: boolean; is: optional>
   sniff_on_connection_fail:        # <default: False; type: boolean; is: optional>
   store_interval_in_secs:          # <default: 5; type: integer; is: optional>
   batch_size:                      # <default: 500; type: integer; is: optional>
   backlog_size:                    # <default: 500; type: integer; is: optional>
```

6.3 File

Store all received events in a file.

file_name: absolute path to filen. String my contain pythons strtime directives and event fields, e.g. %Y-%m-%d.
format: Which event fields to use in the logline, e.g. '$(@timestamp) - $(url) - $(country_code)'
store_interval_in_secs: sending data to es in x seconds intervals.
batch_size: sending data to es if event count is above, even if store_interval_in_secs is not reached.
backlog_size: maximum count of events waiting for transmission. Events above count will be dropped.
compress: Compress output as gzip or snappy file. For this to be effective, the chunk size should not be too small.

Configuration template:

```
- output.File:
   file_name:                       # <type: string; is: required>
   format:                          # <default: '$(data)'; type: string; is: optional>
   store_interval_in_secs:          # <default: 10; type: integer; is: optional>
   batch_size:                      # <default: 500; type: integer; is: optional>
   backlog_size:                    # <default: 500; type: integer; is: optional>
   compress:                        # <default: None; type: None||string; values:␣
↪[None,'gzip','snappy']; is: optional>
```

6.4 Graphite

Send metrics to graphite server.

server: Graphite server to connect to.
port: Port carbon-cache is listening on.
formats: Format of messages to send to graphite, e.g.: ['lumbermill.stats.event_rate_$(interval)s $(event_rate)'].
store_interval_in_secs: Send data to graphite in x seconds intervals.
batch_size: Send data to graphite if event count is above, even if store_interval_in_secs is not reached.
backlog_size: Send count of events waiting for transmission. Events above count will be dropped.

Here a simple example to send http_status statistics to graphite:

```
- output.Statistics:
    interval: 10
    fields: ['http_status']

- output.Graphite:
    filter: if $(field_name) == "http_status"
    server: 127.0.0.1
    batch_size: 1
    formats: ['lumbermill.stats.http_200_$(interval)s $(field_counts.200)',
              'lumbermill.stats.http_400_$(interval)s $(field_counts.400)',
              'lumbermill.stats.http_total_$(interval)s $(total_count)']
```

Configuration template:

```
- output.Graphite:
   server:                          # <default: 'localhost'; type: string; is:␣
↪optional>
   port:                            # <default: 2003; type: integer; is: optional>
   formats:                         # <type: list; is: required>
   store_interval_in_secs:          # <default: 5; type: integer; is: optional>
   batch_size:                      # <default: 50; type: integer; is: optional>
   backlog_size:                    # <default: 50; type: integer; is: optional>
```

6.5 Kafka

Publish incoming events to kafka topic.

topic: Name of kafka topic to send data to.

brokers: Kafka brokers to connect to.

key: Key for compacted topics.

format: Which event fields to send on, e.g. '$(@timestamp) - $(url) - $(country_code)'. If not set, the whole event dict is send.

Configuration template:

```
- output.Kafka:
   topic:                           # <type: string; is: required>
   brokers:                         # <default: ['localhost:9092']; type: list; is:␣
↪optional>
   key:                             # <default: None; type: None||string; is:␣
↪optional>
   format:                          # <default: None; type: None||string; is:␣
↪optional>
```

6.6 Logger

Send data to lumbermill logger.

formats: Format of messages to send to logger, e.g.: ['############## Statistics ##############', 'Received events in $(interval)s: $(total_count)', 'EventType: httpd_access_log - Hits: $(field_counts.httpd_access_log)', 'EventType: Unknown - Hits: $(field_counts.Unknown)']

Configuration template:

```
- output.Logger:
   formats:                         # <type: list; is: required>
```

6.7 MongoDb

Store incoming events in a mongodb.

host: Mongodb server.

database: Mongodb database name.

collection: Mongodb collection name. Timepatterns like %Y.%m.%d and dynamic values like $(bar) are allowed here.

optinonal_connection_params: Other optional parameters as documented in https://api.mongodb.org/python/current/api/pymongo/mongo_client.html

format: Which event fields to send on, e.g. '$(@timestamp) - $(url) - $(country_code)'.
If not set the whole event dict is send.

doc_id: Sets the document id for the committed event data.

store_interval_in_secs: Send data to es in x seconds intervals.

batch_size: Sending data to es if event count is above, even if store_interval_in_secs is not reached.

backlog_size: Maximum count of events waiting for transmission. If backlog size is exceeded no new events will be processed.

Configuration template:

```
- output.MongoDb:
   host:                              # <default: 'localhost:27017'; type: string; is:␣
↪optional>
   database:                          # <default: 'lumbermill'; type: string; is:␣
↪optional>
   collection:                        # <default: 'lumbermill-%Y.%m.%d'; type: string;␣
↪is: optional>
   optinonal_connection_params:       # <default: {'serverSelectionTimeoutMS': 5};␣
↪type: dictionary; is: optional>
   format:                            # <default: None; type: None||string; is:␣
↪optional>
   doc_id:                            # <default: '$(lumbermill.event_id)'; type:␣
↪string; is: optional>
   store_interval_in_secs:            # <default: 5; type: integer; is: optional>
   batch_size:                        # <default: 500; type: integer; is: optional>
   backlog_size:                      # <default: 5000; type: integer; is: optional>
```

6.8 RedisChannel

Publish incoming events to redis channel.

channel: Name of redis channel to send data to.
server: Redis server to connect to.
port: Port redis server is listening on.
db: Redis db.
password: Redis password.
format: Which event fields to send on, e.g. '$(@timestamp) - $(url) - $(country_code)'. If not set, the whole event dict is send.

Configuration template:

```
- output.RedisChannel:
   channel:                           # <type: string; is: required>
   server:                            # <default: 'localhost'; type: string; is:␣
↪optional>
   port:                              # <default: 6379; type: integer; is: optional>
   db:                                # <default: 0; type: integer; is: optional>
   password:                          # <default: None; type: None||string; is:␣
↪optional>
   format:                            # <default: None; type: None||string; is:␣
↪optional>
```

6.9 RedisList

Send events to a redis lists.

list: Name of redis list to send data to.

server: Redis server to connect to.
port: Port redis server is listening on.
db: Redis db.
password: Redis password.
format: Which event fields to send on, e.g. '$(@timestamp) - $(url) - $(country_code)'. If not set the whole event dict is send.
store_interval_in_secs: Send data to redis in x seconds intervals.
batch_size: Send data to redis if event count is above, even if store_interval_in_secs is not reached.
backlog_size: Maximum count of events waiting for transmission. Events above count will be dropped.

Configuration template:

```
- output.RedisList:
   list:                            # <type: String; is: required>
   server:                          # <default: 'localhost'; type: string; is:␣
↪optional>
   port:                            # <default: 6379; type: integer; is: optional>
   db:                              # <default: 0; type: integer; is: optional>
   password:                        # <default: None; type: None||string; is:␣
↪optional>
   format:                          # <default: None; type: None||string; is:␣
↪optional>
   store_interval_in_secs:          # <default: 5; type: integer; is: optional>
   batch_size:                      # <default: 500; type: integer; is: optional>
   backlog_size:                    # <default: 500; type: integer; is: optional>
```

6.10 SQS

Send messages to amazon sqs service.

aws_access_key_id: Your AWS id.
aws_secret_access_key: Your AWS password.
region: The region in which to find your sqs service.
queue: Queue name.
format: Which event fields to send on, e.g. '$(@timestamp) - $(url) - $(country_code)'.
If not set event.data will be send es MessageBody, all other fields will be send as MessageAttributes.
store_interval_in_secs: Send data to redis in x seconds intervals.
batch_size: Number of messages to collect before starting to send messages to sqs. This refers to the internal receive buffer of this plugin. When the receive buffer is maxed out, this plugin will always send the maximum of 10 messages in one send_message_batch call.
backlog_size: Maximum count of events waiting for transmission. Events above count will be dropped.

values: ['us-east-1', 'us-west-1', 'us-west-2', 'eu-central-1', 'eu-west-1', 'ap-southeast-1', 'ap-southeast-2', 'ap-northeast-1', 'sa-east-1', 'us-gov-west-1', 'cn-north-1']

Configuration template:

```
- output.SQS:
   aws_access_key_id:                # <type: string; is: required>
   aws_secret_access_key:            # <type: string; is: required>
   region:                           # <type: string; is: required>
   queue:                            # <type: string; is: required>
   format:                           # <default: None; type: None||string; is:␣
↪optional>
   store_interval_in_secs:           # <default: 5; type: integer; is: optional>
   batch_size:                       # <default: 500; type: integer; is: optional>
   backlog_size:                     # <default: 500; type: integer; is: optional>
   receivers:
    - NextModule
```

6.11 StdOut

Print the data dictionary to stdout.

pretty_print: Use pythons pprint function.
fields: Set event fields to include in pretty print output.
format: Format of messages to send to graphite, e.g.: ['lumbermill.stats.event_rate_$(interval)s $(event_rate)'].

Configuration template:

```
- output.StdOut:
   pretty_print:                     # <default: True; type: boolean; is: optional>
   fields:                           # <default: None; type: None||list; is: optional>
   format:                           # <default: None; type: None||string; is:␣
↪optional>
```

6.12 Syslog

Send events to syslog.

format: Which event fields to send on, e.g. '$(@timestamp) - $(url) - $(country_code)'. If not set the whole event dict is send.
address: Either a server:port pattern or a filepath to a unix socket, e.g. /dev/log.
proto: Protocol to use.
facility: Syslog facility to use. List of possible values, @see:
http://epydoc.sourceforge.net/stdlib/logging.handlers.SysLogHandler-class.html#facility_names

Configuration template:

```
- output.Syslog:
   format:                           # <type: string; is: required>
   address:                          # <default: 'localhost:514'; type: string; is:␣
↪required>
```

(continues on next page)

(continued from previous page)

```
   proto:                              # <default: 'tcp'; type: string; values: ['tcp',
↪'udp']; is: optional>
   facility:                           # <default: 'user'; type: string; is: optional>
```

6.13 Udp

Send events to udp socket.

address: address:port
format: Which event fields to send on, e.g. '$(@timestamp) - $(url) - $(country_code)'. If not set the whole event dict is send.
store_interval_in_secs: Send data to redis in x seconds intervals.
batch_size: Send data to redis if event count is above, even if store_interval_in_secs is not reached.
backlog_size: Maximum count of events waiting for transmission. Events above count will be dropped.

Configuration template:

```
- output.Udp:
   address:                            # <default: 'localhost:514'; type: string; is:␣
↪required>
   format:                             # <default: None; type: None||string; is:␣
↪optional>
   store_interval_in_secs:             # <default: 5; type: integer; is: optional>
   batch_size:                         # <default: 500; type: integer; is: optional>
   backlog_size:                       # <default: 500; type: integer; is: optional>
```

6.14 WebHdfs

Store events in hdfs via webhdfs.

server: webhdfs/https node | **user**: Username for webhdfs. | **path**: Path to logfiles. String my contain any of pythons strtime directives. | **name_pattern**: Filename pattern. String my conatain pythons strtime directives and event fields. | **format**: Which event fields to send on, e.g. '$(@timestamp) - $(url) - $(country_code)'. If not set the whole event dict is send. | **store_interval_in_secs**: Send data to webhdfs in x seconds intervals. | **batch_size**: Send data to webhdfs if event count is above, even if store_interval_in_secs is not reached. | **backlog_size**: Maximum count of events waiting for transmission. Events above count will be dropped. | **compress**: Compress output as gzip file. For this to be effective, the chunk size should not be too small.

Configuration template:

```
- output.WebHdfs:
   server:                             # <default: 'localhost:14000'; type: string; is:␣
↪optional>
   user:                               # <type: string; is: required>
   path:                               # <type: string; is: required>
   name_pattern:                       # <type: string; is: required>
   format:                             # <type: string; is: required>
   store_interval_in_secs:             # <default: 10; type: integer; is: optional>
```

(continues on next page)

(continued from previous page)

```
   batch_size:                          # <default: 1000; type: integer; is: optional>
   backlog_size:                        # <default: 5000; type: integer; is: optional>
   compress:                            # <default: None; type: None||string; values:␣
↪[None,'gzip','snappy']; is: optional>
```

6.15 Zabbix

Send events to zabbix.

hostname: Hostname for which the metrics should be stored. fields: Event fields to send. field_prefix: Prefix to prepend to field names. For e.g. cpu_count field with default **lumbermill_** prefix, the Zabbix key is lumbermill_cpu_count. timestamp_field: Field to provide timestamp. If not provided, current timestamp is used. agent_conf: Path to zabbix_agent configuration file. If set to True defaults to /etc/zabbix/zabbix_agentd.conf. server: Address of zabbix server. If port differs from default it can be set by appending it, e.g. 127.0.0.1:10052. store_interval_in_secs: sending data to es in x seconds intervals. batch_size: sending data to es if event count is above, even if store_interval_in_secs is not reached. backlog_size: maximum count of events waiting for transmission. Events above count will be dropped.

Configuration template:

```
- output.Zabbix:
   hostname:                            # <type: string; is: required>
   fields:                              # <type: list; is: required>
   field_prefix:                        # <default: "lumbermill_"; type: string; is:␣
↪optional>
   timestamp_field:                     # <default: "timestamp"; type: string; is:␣
↪optional>
   agent_conf:                          # <default: True; type: boolean||string; is:␣
↪optional>
   server:                              # <default: False; type: boolean||string; is:␣
↪required if agent_conf is False else optional>
   store_interval_in_secs:              # <default: 10; type: integer; is: optional>
   batch_size:                          # <default: 500; type: integer; is: optional>
   backlog_size:                        # <default: 500; type: integer; is: optional>
```

6.16 Zmq

Sends events to zeromq.

server: Server to connect to. Pattern: hostname:port.
pattern: Either push or pub.
mode: Whether to run a server or client. If running as server, pool size is restricted to a single process.
topic: The channels topic.
hwm: Highwatermark for sending socket.
format: Which event fields to send on, e.g. '$(@timestamp) - $(url) - $(country_code)'. If not set the whole event dict is send msgpacked.
store_interval_in_secs: Send data to redis in x seconds intervals.
batch_size: Send data to redis if event count is above, even if store_interval_in_secs is not reached.
backlog_size: Maximum count of events waiting for transmission. Events above count will be dropped.

Configuration template:

```
- output.Zmq:
   server:                          # <default: 'localhost:5570'; type: string; is:␣
↪optional>
   pattern:                         # <default: 'push'; type: string; values: ['push',
↪ 'pub']; is: optional>
   mode:                            # <default: 'connect'; type: string; values: [
↪'connect', 'bind']; is: optional>
   topic:                           # <default: None; type: None||string; is:␣
↪optional>
   hwm:                             # <default: None; type: None||integer; is:␣
↪optional>
   format:                          # <default: None; type: None||string; is:␣
↪optional>
   store_interval_in_secs:          # <default: 5; type: integer; is: optional>
   batch_size:                      # <default: 500; type: integer; is: optional>
   backlog_size:                    # <default: 500; type: integer; is: optional>
```

CHAPTER 7

Cluster modules

7.1 Pack

Pack base module. Handles pack leader discovery and alive checks of pack followers.

IMPORTANT: This is just a first alpha implementation. No leader election, no failover, no sanity checks for conflicting leaders.

name: Name of the cluster. Used for auto-discovery in same network.
secret: pre shared key to en/decrypt cluster messages.
broadcast: Ipaddress for udp broadcasts.
interface: Ipaddress to listen on.
port: Port to listen on.
interval: Autodiscover interval.
pack: Set this node to be either leader or member.

Configuration template:

```
- Pack:
    name:                           # <type: string; is: required>
    secret:                         # <type: string; is: required>
    broadcast:                      # <type: string; is: required>
    interface:                      # <default: '0.0.0.0'; type: string; is: optional>
    port:                           # <default: 5252; type: integer; is: optional>
    interval:                       # <default: 10; type: integer; is: optional>
    pack:                           # <default: 'leader'; type: string; values: [
↪'leader', 'follower']; is: optional>
```

7.2 PackConfiguration

Synchronize configuration from leader to pack members. Any changes to the leaders configuration will be synced to all pack followers.

Locally configured modules of pack members will not be overwritten by the leaders configuration.

Module dependencies: ['Pack']

pack: Name of the pack module. Defaults to the standard Pack module.
ignore_modules: List of module names to exclude from sync process.
interval: Time in seconds between checks if master config did change.

Configuration template:

```
- PackConfiguration:
   pack:                            # <default: 'Pack'; type: string; is: optional>
   ignore_modules:                  # <default: []; type: list; is: optional>
   interval: 10                     # <default: 60; type: integer; is: optional>
```

CHAPTER 8

Plugin modules

8.1 WebGui

A WebGui plugin for LumberMill. At the moment this is just a stub.

Module dependencies: WebserverTornado

Configuration template:

```
tornado_webserver: Name of the webserver module.
document_root: Path to look for templates and static files.

- plugin.WebGui:
   tornado_webserver: webserver     # <default: 'WebserverTornado'; type: string; is:␣
↪optional>
   document_root: other_root        # <default: 'docroot'; type: string; is: optional>
```

8.2 WebserverTornado

A tornado based web server.

Configuration template:

```
- plugin.WebserverTornado:
   port:                            # <default: 5100; type: integer; is: optional>
   tls:                             # <default: False; type: boolean; is: optional>
   key:                             # <default: False; type: boolean||string; is:␣
↪required if tls is True else optional>
   cert:                            # <default: False; type: boolean||string; is:␣
↪required if tls is True else optional>
   document_root:                   # <default: '../assets/webserver_docroot'; type:␣
↪string; is: optional>
```

(continues on next page)

(continued from previous page)

```
   statistic_module_id:              # <default: "SimpleStats"; type: string; is:␣
↪optional>
   application_settings:             # <default: None; type: None||dict; is: optional>
```

CHAPTER 9

Indices and tables

- genindex
- modindex
- search