Events Development

# TODO

* **Uncomment //@Test and fix**
* Add support for "&quot;%I&quot; %h %u [%t] &quot;%r&quot; &quot;%q&quot; %s %b %D"
* Add support for –filter=expression. Example: --filter="original-request-status-code=400"
* **Property in-line Sorting based on Priority vs. addition order getPropertyList().** Consider property priority (and sorting) vs fixed order. Reconsider getPropertyList();
* **Insure the help is true.**“output” in-line help: *If the property is a Map, the following notation can be used:* Make sure this works.
* **Terminator disabling defect:** “describe” disables the terminator, and because the disabling logic is not well written, I get this:

@main 13:29:13,172 ERROR [EventProcessor] Output Writer failed to close the input stream

@main 13:29:16,177 WARN [ComponentBase] Output Writer did not stop in 3000 milliseconds, abandoning it ...

Investigate and fix.

* **Sampler**: rigorous testing, essential piece of functionality.
* **Sampler**: find the correct name of the statistical operation and rename. Resampling?
* **Analyze the usage of EndOfStreamEvent and ShutdownEvent** and decide if:
  + a ShutdownEvent is needed
  + if it is not needed, eliminate it
  + if it is needed, define behavior and add appropriate tests at the appropriate levels.
* Currently we take the easy way out by wrapping a HttpdLogLine in an Event – do we want to create a HttpdLogEvent?
* Fix all unit tests (including those commented out)
* Refactor InputStreamInitiator, EventProcessor and OutputStreamTerminator thread internals – there is much common behavior – unify.
  + Currently I deal with EOSListener only in OutputStreamtTerminators – it should be generic.
  + The “logics” need a base class, there is much shared behavior. Analyze what I implemented so far and factor out the common behavior.
* Currently the shutdown is initiated by inserting a ShutdownEvent in the queue. We can also interact with the Component thread directly, if we need to shut it down faster. Think about it.
* Idea: use non-blocking IO in components and read from both a data channel and a control channel. The data channel is an adapter to an InputStream.
* Unit test for resampling.
* Understand fix and document why Maven blocks at the end.
* How to handle exceptions in the logic’s process() High level view.
* Handle EndOfStream in SingleThreadedEventProcessor. Test.
* Uncomment commented out tests and make them pass.
* Need an EndOfStreamEvent
* Need a OutputStreamEvent that gets written to the output stream.
* **EndOfStreamListeners management in ComponentBase.** Analyze EndOfStreamListener usage and decide whether we need to add thread safety for the management functions.
* **Separate in ‘events’, ‘clad’, ‘httpd logs’, ‘csv’**
* **Configuration should also flow as “event”** CSV headers for example.
* **Should allow for null output queues.** It’ll just discard events, but makes easy configuring stuff. Think /dev/null.
* **Look at:**
  + <https://github.com/heroku/logplex>
  + <https://github.com/fluent/fluentd>
  + <http://www.splunk.com>
  + <http://www.logstash.com>