Event Stream Analyzer

# TODO

* 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.
* **Possible names:** eventful, events.
* <https://github.com/heroku/logplex>
* <https://github.com/fluent/fluentd>
* <http://www.splunk.com>
* <http://www.logstash.com>