Events API (events-api) TODO

**Query improvements:**

* Display the first event that matches the query and then immediately exit the runtime (--first [count])
* Display the last query that matches the query (--last [count]): optimization for files – start from the bottom?
* Time bounds --from --to
* Think of a generic query language.

<https://kb.novaordis.com/index.php/Events-api>

* **JBoss-Related Heuristics**. Move the JBoss-related heuristics (classpath, figuring out version, etc.) from JmxMetricSourceDefinitionUtil into a dedicated project: jboss-cli and rename? novaordis-util? novaordis-jboss?
* **novaordis-utilities**: Move OSType into utilities, alongside OS.
* **novaordis-utilities:** Move sshConnection and all SSH stuff in its own project or utilities.
* Keep <https://kb.novaordis.com/index.php/Events-api_Concepts> up to date.

# Imported from Events Core

* events should be able to modularly parse a Dynatrace CSV output and produce a multi-field timed series that characterizes the system.
* **Tests are commented out** – before doing the next non-SNAPSHOT release, uncomment and fix.
* 12/15/16 the possibility to do a partial parsing of a httpd log line, by specifying only the first (interesting) tokens, and ignoring the rest.
* 10/04/16 Sampling and averaging? Response time and throughput. Use the existing Sampler?
* 10/04/16 Think of an extension that parses the log events based on the log4j pattern and turns them into Events events? Or an Appender?
* 09/03/16 rename the artifact to "novaordis-events.\*", to align it with the rest of the "projects" artifact names.
* 07/25/16 FirstRequestLineParser.identifyEnd() and HttpdLogLine.parseFirstRequestLine() implementations should do the same thing, but they are doing different things:HttpdLogLine.parseFirstRequestLine() is more permissive and allows two or three elements ("GET /path" and "GET /path HTTP-version", while FirstRequestLineParser.identifyEnd() assumes three elements. Unify the implementation.
* 07/25/16 **httpd log parser**: PID:"%P" generates PID:"12121" in the logs. Parse that natively.
* Return to Business Scenarios 07/03/16
  + Start with enabling the commented out test "TODO N7aq32 RETURN HERE"
  + Header values for business scenario CSV: TODO k342t - figure out how to handle the fact that are multiple scenario types, each of them with a different number of requests
* document the methodology to draw this) **Request layer cake for scenarios** Per-request breakdown - layer cake at request level.
* 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.
* 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?
* 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.
* 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.
* 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.splunk.com/)
  + [http://www.logstash.com](http://www.logstash.com/)
* **Filters** Add support for –filter=expression. Example:

--filter="original-request-status-code=400"

or

--filter request.Business-Scenario-Start-Marker=TYPE1

* <https://kb.novaordis.com/index.php/Business_Scenario-Based_Performance_Monitoring_and_Diagnosis_Development_TODO>