Engine Administration Console Guide

Last Update: 12/13/16

1. Overview

The Engine Administration Console (EAC) is a web-based interface for system administrators to monitor the health and status of one or more LUX Engine instances. Additionally, the EAC enables admins to perform functions to modify the operation and configuration of LUX on the fly.

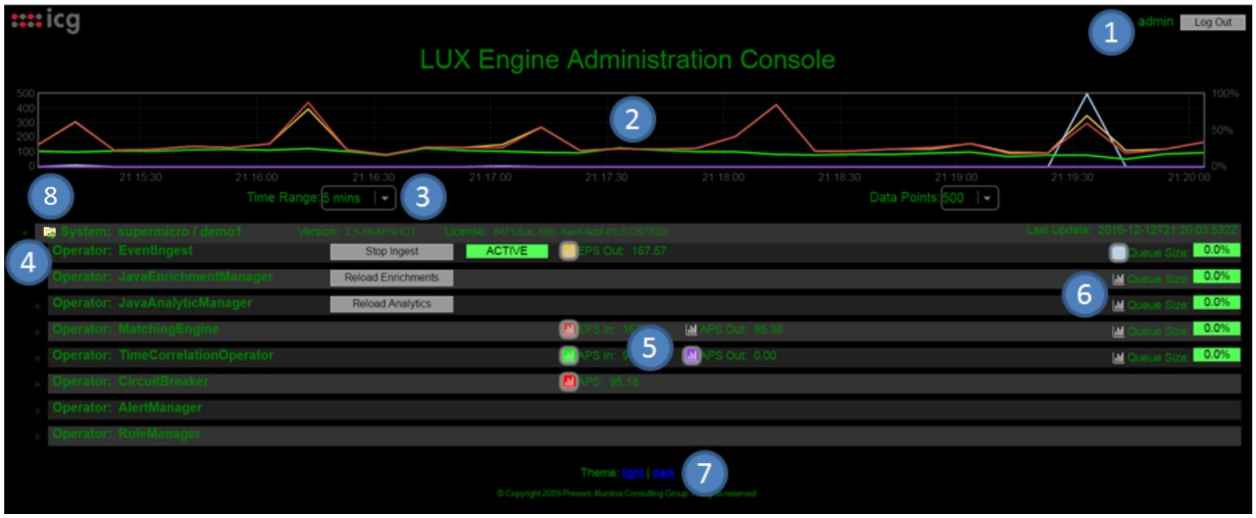

Figure 1 - EAC Overview

Figure 1 shows an overview of the EAC, the numbered components are as follows:

1) Shows the user currently logged in, and provides a button to log out of the EAC.
2) Graphs currently selected metrics in real time.
3) Graph controls allow for modification of time range and data points represented by the graph.
4) Status Tree, where each Engine instance and its components can be expanded to examine in detail. The Status Tree is explained in detail in Section 2.
5) Graph icons are present throughout the EAC, and allow the user to select which metrics appear on the graph.
6) Queue sizes are displayed for many operators, showing any bottlenecks in data flow
7) Allows for EAC theme selection (dark theme shown).
8) File-View (folder icon). This redirects to a new page where users can view, modify, add, and delete files and directories within the LUX Engine. File-View is explained in Section 3.

2. Status Tree

2.1 System

The top level of the Status Tree are System nodes, each representing an instance of the LUX Engine.

Figure 2 - System node

1) System name
2) LUX Engine software version
3) License key
4) Timestamp from the last communication with the Engine. If the Engine fails to communicate with the EAC, the System node will appear red (Figure 3)

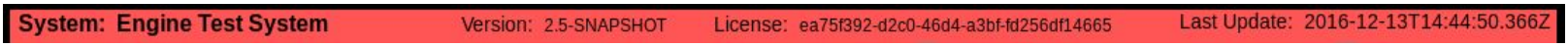

Figure 3 - Loss of Communication with Engine

2.2 Event Ingest

The Event Ingest operator brings data into the LUX Engine via plugins. Each plugin has a status row under EventIngest. Figure 4 shows the Event Ingest section of the Status Tree.

Figure 4 - Event Ingest

Figure 5 - Event Ingest stopped

1) “Stop Ingest” button, disables all Event Ingest plugins and stops the flow of data into the LUX Engine. Will toggle to “Start Ingest” once the operator has stopped (Figure 5).

2) Status Indicator
    a) ACTIVE - ingest is on, data is flowing
    b) STOPPED - ingest is off, no data is flowing
3) Events per second total from all plugins
4) Output queue size. If this queue fills up, EventIngest is running faster than Enrichment (which may itself be waiting for something downstream).
5) Plugin name
6) Seconds since last event was produced by this plugin
7) Events per second produced by this plugin (average over the last update interval)
8) Total events produced by this plugin since startup (or reload of plugin)
9) Button to stop/start this ingest plugin individually
10) Status indicator for whether this plugin is currently running

2.3 Enrichment Manager

The Enrichment Manager loads Enrichment plugins, which add metadata to events. Each plugin has a status row under EnrichmentManager. Figure 6 shows the Enrichment section of the Status Tree.

Figure 6 - Enrichment Manager

1) Reload Enrichments button, causes all plugins to be reloaded from the **`ae.xml`** configuration file. Plugins may be added, removed, and/or reconfigured by this process.
2) Output queue size. If this queue fills up, Enrichment is running faster than Analytics or EventOutput (which may themselves be waiting for something downstream).
3) Plugin name
4) Events per second processed by this plugin (average over the last update interval)
5) Events per second to which this plugin added metadata (average over the last update interval)
6) Total events to which this plugin added metadata since startup (or reload)
7) Button to stop/start this plugin individually
8) Status indicator for whether this plugin is currently running

2.3 Analytic Manager

The Enrichment Manager loads Enrichment plugins, which add metadata to events. Each plugin has a status row under AnalyticManager. Figure 7 shows the Analytic section of the Status Tree.

Figure 7 - Analytic Manager

1) Reload Analytics button, causes all plugins to be reloaded from the **`ae.xml`** configuration file. Plugins may be added, removed, and/or reconfigured by this process.asdf
2) Output queue size. If this queue fills up, Analytics are running faster than the Matching Engine (which may itself be waiting for something downstream).
3) Plugin name
4) Events per second processed by this plugin (average over the last update interval)
5) Analytic plugins create Analytic Events, which contain zero or more normal Events, along with metadata about the analytic result. EPS Out is the number of constituent normal events contained in Analytic Events output per second (average since last update).
6) Analytic Events per second output average since the last update.
7) Total Analytic Events output since startup (or reload)
8) Button to stop/start this plugin individually
9) Status indicator for whether this plugin is currently running

2.4 Matching Engine

The Matching Engine performs CEP matching between Rules from users and Events from Ingest plugins, and produces Alerts when they match. Figure 8 shows the Matching Engine section of the Status Tree.

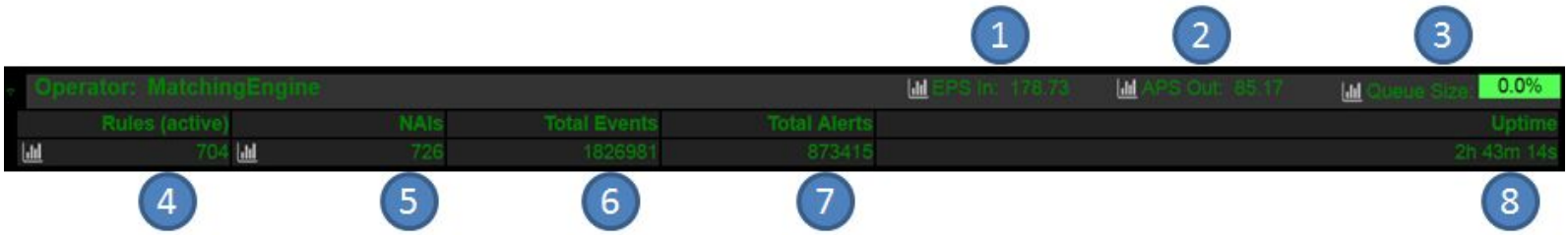

Figure 8 - Matching Engine

1) Events per second the Matching Engine is processing
2) Alers per second the Matching Engine is producing
3) Output queue size. If this queue fills up, the Matching Engine is running faster than the Time Correlation Operator (which may itself be waiting for something downstream).
4) Active Rules currently running in the Matching Engine
5) Named Areas of Interest currently stored in the Matching Engine
6) Total events processed by the Matching Engine since startup
7) Total alerts created by the Matching Engine since startup
8) Time since startup

2.5 Time Correlation

The Time Correlation operator evaluates Time Correlated rules against a stream of Alerts from the Matching Engine. Figure 9 shows the Matching Engine section of the Status Tree.

Figure 9 - Time Correlation

1) Alerts per second being processed
2) Time Correlated alerts per second being produced
3) Output queue size. If this queue fills up, the Time Correlation Operator is running faster than the Alert Manager (which may itself be waiting for something external to LUX).
4) Number of Time Correlated rules being processed
5) Total alerts processed since startup
6) Total Time Correlated alerts produced since startup

2.6 Circuit Breaker

The Circuit Breaker automatically disables rules which produce too many alerts for their alert destinations. Each Alerter plugin has a row under CircuitBreaker. Figure 10 shows the CircuitBreaker section of the Status Tree.

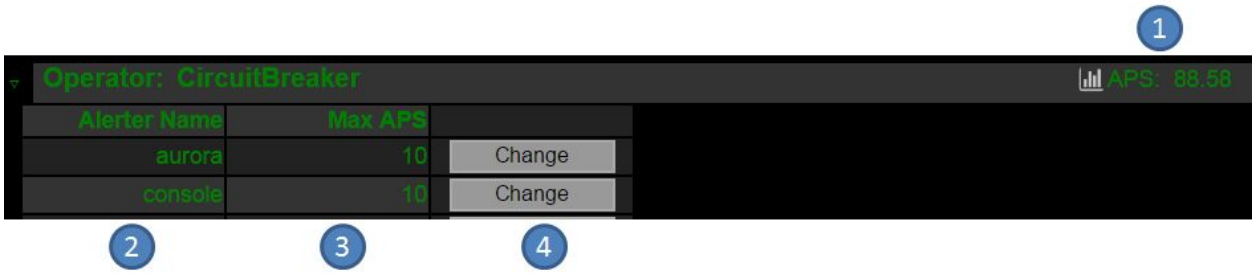

Figure 10 - Time Correlation

1) Alerts per second processed by the Circuit Breaker
2) Name of Alerter plugin destination
3) Max alerts per second allowed to the Alerter plugin before rules are disabled
4) Button to change Max APS for an Alerter plugin

2.7 Alert Manager

The Alert Manager sends Alerts out of the LUX Engine to various destinations determined by Alerter plugins and their configurations. Each Alerter plugin has a status row under AlertManager. Figure 11 shows the AlertManager section of the Status Tree.

Figure 11 - Alert Manager

1) Alerter plugin name
2) Alerts per second processed by the plugin
3) Total alerts processed by the plugin since startup
4) Output queue size. If this queue backs up, the plugin is unable to send alerts to the destination as fast as they're coming in.
5) Button to purge all pending alerts in the queue, they will be deleted
6) Current policy selection for what to do when the queue fills up
    a) WAIT - backs up the pipeline without discarding data

b) CACHE - caches unsent alerts to disk
c) DROP - discards alerts and doesn't back up the pipeline

7) Button to change the discard policy

2.8 Rule Manager

The Rule Manager stores a cache of the latest user Rules. Each Rule has a status line under Rule Manager. Figure 12 shows the Rule Manager section of the Status Tree.

Figure 12 - Rule Manager

1) Each column of the Rules table has a Sort button
2) Rule name
3) Rule ID
4) Total number of alerts produced by rule since startup
5) Alerters the rule is configured to alert to
6) Project that the rule belongs to
7) Status of rule

3. File-View

3.1 File Tree

The Engine Admin Console File Tree provides user access to files and directories utilized for the LUX Engine configuration. The top level of the tree starts at:
`/lux/engine/EngineMain/data/.`

Figure 13 shows the basic layout of each directory within the File Tree.

Figure 13 - Analytic Manager

1) Upload a new file to the listed directory
2) Create a new subdirectory within the current directory
3) Delete the current directory.
4) Download the selected file.
5) Replace the selected file by uploading a new file.
6) Delete the selected file.