Lemon Tutorial

Sensor Exception

Miroslav Siket, Dennis Waldron
http://cern.ch/lemon
CERN-IT/FIO-FD

Outline

- What is it?
- Configuration
- Correlation Examples.
- Actuators
- Dealing with transient alarms.

What is it?

- Sensor-exception
  - An officially supported Lemon sensor coded in C++.
  - Developed in collaboration between CERN and BARC.
  - Implements the Lemon alarm protocol.
  - Has a LEX & YACC correlation engine which allows it to evaluate 1 or more metrics to determine if a problem exists on a machine.
  - Supports reporting alarms on behalf of other monitored entities.
  - Allows corrective actions (actuators) up to n-times or within a given time window.
  - Is the primary interface to inserting alarms into the Lemon framework. The output of the sensor is used by LAS and lemon-host-check.
  - Provides one and only one metric class “alarm.exception”
- Full documentation at:
  - http://lemon.web.cern.ch/lemon/doc/sensors/exception.shtml

Configuration

- The sensor has 6 configuration options:
  - Correlation
    - The power behind the sensor exceptions capabilities
    - This tells the sensor which metrics are involved in the alarm and how they should be evaluated
  - Actuator
    - The path to an actuator to run if the correlation string is true.
  - MaxRuns
    - The maximum number of times an actuator can run consecutively before a final alarm is generating
  - Timeout
    - The maximum number of seconds that an actuator is allowed to run before being terminated by the sensor.
  - MinOccurs
    - The minimum number of consecutive times a problem must be present before raising an alarm.
    - Good for dealing with transient alarms.
  - Silent
    - Defines whether the exception should run in silent mode. A silent exception will continue to be evaluated but the result will not be displayed on LAS or lemon-host-check.
    - Good for testing and deployment of new alarms.

Configuration (II)

- Basic format of a correlation is:

  [entity_name]:<metric_id>:<field_position> <operator> <reference_value> ...

- Where,
  - entity_name
    - An optional parameter, used for reporting on behalf of other entities
    - The name of the entity (wildcards '*' are supported)
  - metric_id
    - The id of the metric to check
  - field_position
    - The field to use within the metric.
    - Allows the correlation to extract a single value from a multi-valued metric
  - Operater
    - E.g. ==, !=, >, <, eq, ne, regex, !regex …
  - reference_value
    - A string or number used to compare the metric_id:field_position against

Correlation Example (I)

- Objective:
  - To run a actuator when the occupancy of the /tmp partition is greater then 80%.
- Involved Metrics
  - 9104 (system.partitionInfo)
  - Field 1 = mountname, field 5 = percentage occupancy
- Correlation

Correlation	**((9104:1 eq '/tmp') && (9104:5 > 80))**
Actuator	/usr/local/sbin/clean-tmp-partition -o 75
MaxRuns	3 900
Timeout	300

Correlation Example (II)

- Objective:
  - To raise an alarm “lemon_agent_wrong” if the memory utilisation, cpu utilisation or number of errors in the agents log file is not within acceptable limits.
- Correlation

10004:1 > 600 && (10004:7 > 10 || (10004:8 > 150000 && 4109:3 eq 'i386') ||
(10004:8 > 600000 && 4109:3 regex '64') || 10007:2 > 50 || 10007:3 > 10 || 10007:4 > 0)

If the:
(uptime of the agent (10004:1) is greater then 600 seconds) **AND**
(the cpu utilisation of the sensors (10004:7) over the last sampling frequency is greater then 10%) **OR**
(the memory consumed by the sensors (10004:8) is greater then 150 megabytes for machines of architecture type (4109:3) i386 or 600 megabytes for machines of architecture type x86_64) **OR**
(the number of warning messages (10007:2) recorded over the last sampling frequency is greater the 50) **OR**
(the number of error messages (10007:3) recorded over the last sampling frequency is greater the 10) **OR**
(the number of fatal messages (10007:3) recorded over the last sampling frequency is greater the 0) **raise an alarm**

Actuators

- Information:
  - Run as forked processes.
  - Are connected to the sensor via a pipe.
  - All information written to stdout or stderr by the actuator is caught and recorded in the agents log file.
  - All actuator attempts are logged centrally and recorded locally in the agents log file.
- Running shell style actuators:
  - The system call used to run actuator doesn't provide shell style conveniences.
  - To use shell style syntax like *, &&, | etc you must define you actuator like this:

Actuator /bin/sh –c \\" /bin/echo 'This is a demo message from $HOSTNAME' \\"

Dealing with transient Alarms

- Why do we get transient alarms?
  - By default monitoring isn't very tolerant of outside interventions
  - Maybe network issues.
  - A resource maybe temporarily unavailable.
- What can be done?
  - Use the configuration option MinOccurs
  - MinOccurs gives an exception a level of tolerance, a delay factor between detecting a problem and raising an alarm