opEvents EventParserRules - Adding Rules For SNMP Traps

Overview

opEvents provides the ability for the adminstrator to customise an event's properties from a variety of inputs. For example, if a user wanted to set a specific priority for an event it can be done during the input parsing stages. This article will provide a methodology for creating events from SNMP traps, via a generic extensible parser with `EventParserRules`.

The generic parser rules are defined in `EventParserRules.json` which is found in the configuration directory `/usr/local/omk/conf`. Please read the notes at the top of this file first as they are very informative as to what is possible in regard to the parser rules.

Evaluate The Traps To Be Processed

Create a list of SNMP traps that are required be processed by opEvents.

Correlate Events Into Stateful Pairs

For this discussion we will assume that the concept of 'state' is desirable. i.e. If there is a "down" event, there should be a corresponding "up" event, and opEvents should keep track of the state and ignore duplicate inputs. (It is possible that several "down" events could share a single "up" or clearing event.)

State

opEvents tracks state based on a tuple of three event properties.

- node
- element
- stateful

This is a critical concept. The node property will always be the same for any given node. The element property will be somewhat dynamic, usually a regular expression will parse and 'capture' it. The most comment element example would be an interface; gig0/0 versus gig0/1. The stateful property is necessary because the same element may have different events; consider an interface down event versus an OSPF event on the same element (gig0/0).

If any of these three event properties are not set state will not function well.

Consider a case where the element property is not set; thus being null. In this case if a 'port down' for gig0/0 was received a 'port up' for gig0/1 would clear the g0/0 'port down' event. Without the element being set opEvents cannot differentiate between interfaces.

Example parser rule for the element property.

```
            "53" : {
               "IF" : "IF-MIB::ifIndex\\.\\d+=(\\d+)",
               "THEN" : [
                  "capture(element)"
               ]
            },
```

Example parser rule for the stateful property

```
            "51" : {
               "IF" : "IF-MIB::linkDown",
               "THEN" : [
                  "set.event(Interface Down)",
                  "set.stateful(Interface)",
                  "set.state(down)",
                  "set.priority(3)"
               ]
            },
```

Create Parser Rules

opEvents will process the trap log file as specified on opCommon.json. When parsing the traps, at least the following properties should be extracted:

- date
- host
- trap
- details
- event
- element
- stateful
- state
- priority

The shipped version of `EventParserRules.json` has a traplog section that will extract the date, host, trap and details fields for most situations.

This article focuses on situations where customers want customization for the remaining fields.

Set the Element

Review all the SNMP traps to determine which OID best describes what will become the element property. Write a regular expression that matches this.

```
### This is observed the trap opEvents is receiving, and
### is the best candidate to become the element property:

STARENT-MIB::starSlotNum=6

### Refering to the vendors mib file starSlotNum is found:

starSlotNum OBJECT-TYPE
    SYNTAX  Integer32(1..48)
    MAX-ACCESS  accessible-for-notify
    STATUS  current
    DESCRIPTION
        "The slot number"
    ::= { starSlotEntry 1 }
```

Based on this we can write the regular expression to set the element.

```
"2": {
        "DESCRIPTION": "Set element for card number.",
    "IF": "(STARENT-MIB::starSlotNum=\d+)",
    "THEN": ["capture(element)"]
},
```

Notice the regular expression will catch an number of digits following the '=' character. This rule 'captures' the element. In this way we can dynamically assign event properties based on a regular expression.

Set Other Properties

Generally the other properties that we wish to set can be done with one rule. Consider the following trap received by opEvents.

```
2017-07-12T12:23:37   10.113.176.4    UDP: [10.113.176.4]:36570-&gt;[10.255.26.7]
SNMPv2-MIB::sysUpTime.0=6:1:04:53.72
SNMPv2-MIB::snmpTrapOID.0=STARENT-MIB::starCardTempOK
STARENT-MIB::starSlotNum=1
STARENT-MIB::starCardTemperature=40 degrees Celcius
SNMPv2-MIB::snmpTrapEnterprise.0=STARENT-MIB::starentTraps
```

Evaluating this trap it's determined that a single rule can set the properties below.

Notice that if the author is creative and accurate with regular expressions the number of rules my be decreased.

```
"103": {
    "IF": "STARENT-MIB::starCardTempOK",
    "THEN": [
      "set.event(Card Temperature OK)",
      "set.stateful(temperature)",
      "set.state(up)",
      "set.priority(2)"
    ]
},
```

Based on a match of "STARENT-MIB::starCardTempOK", the rule will take action.

- event - "Card Temperature OK"
- stateful - "temperature"
- state - "up"
- priority - "2"