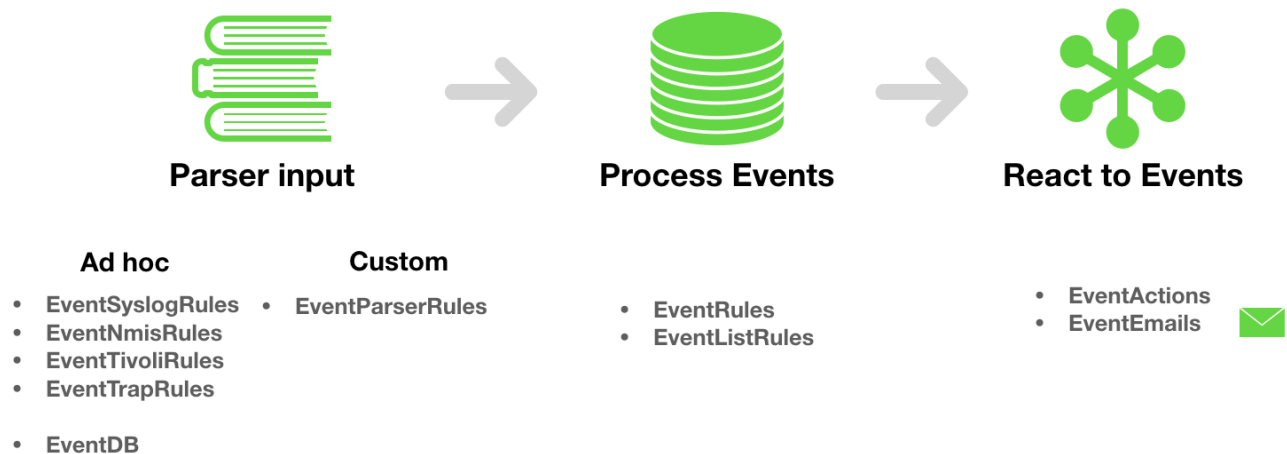


Configuring opEvents



opEvents is flexible and configurable. In the above schema you can see all the involved files to configure opEvents.

Parser Input

opEvents can parse a variety of inputs in a wide variety of formats, and the way opEvents has to understand them is with parsers. opEvents has different parsers for [natively-supported log formats](#), to filter and normalise the data. It uses some files for each format file supported:

- **EventSyslogRules**
- **EventNmisRules**
- **EventTivoliRules**
- **EventTrapRules**

It also can process [events from a database](#). All this configuration is in the file **EventDB.nmis**.

If some other syslog formats are required, we can write our [own parser](#) in **EventParserRules.nmis** file. It defines regular expressions to extract data and the variables to assign them to. If a parser is not enough, we can also use a [parser plugin](#) and generate the event using perl code.

Process Events

It is possible to specify rules to reduce voluminous inputs down to the relevant details, suppress [duplicates](#) or [correlate](#) events. This is the purpose of **EventRules.nmis** file. **EventListRules.nmis** also helps in controller the event in the system by [blacklisting](#) of [whitelisting](#) the events.

React to Events

Once the event is created, **EventActions.nmis** file specifies how to react to events. The file consist in a set of rules with flexibility to being adapted to a wide variety of use cases.

This file can be edited used the opEvents editor, <http://host/en/omk/opEvents/config/actions>:

ppcvs2

VIEWS

Views

Modules

System

Help

User.rms

Home

Event Actions Editor

Validate
 Load From Server
 Save

Filter Period ↺

Draft autosaved on Tuesday, May 19th 2020, 12:19:28 pm ↻ Restore 🗑️ Trash Autosave

Event Actions

```

<>
27   if == {
28       THEN => "event_event == gr(Node Up)",
29       THEN => "priority(<1) AND escalate EscalatePolicy()",
30       BREAK == "false"
31   }
32   IF "AB" == {
33       IF == "event_event == 'Proactive Interface (InputOutput) Utilisation'",
34       THEN == "script.opconfigIC()",
35       BREAK == "false",
36       IF
37     }
38     BREAK == "false"
39   },
40   IF == {
41       THEN == eq "PACK*",
42       IF == {
43         "1B" == {
44           IF == "node.roleType eq 'distribution' and event.stateful eq 'ISG Neighbor'",
45           THEN == "priority(<2) AND script.ping_node() AND script.ping_neighbor()",
46           BREAK == "true"
47         },
48         "2B" == {
49           IF == "node.roleType == gr{(distribution|core|access)} and event.event eq 'Node Down'",
50           THEN == "script.ping_node()",
51           BREAK == "true"
52         },
53         "3B" == {
54           IF == "node.roleType == gr{(distribution|core)} and event.event eq 'SWMP Down'",
55           THEN == "script.ping_node()",
56           BREAK == "false"
57         },
58         "4B" == {
59           IF == "node.roleType == q{f} and event.event eq 'Service Down'",
60           THEN == "script.ping_node()",
61           BREAK == "false"
62         },
63       },
64       BREAK == "true"
        
```

Help

Event Actions policy consists of any number of nested *If-then*-that clauses, which specify the conditions an event must conform to and what actions to take in case of a match.

Further configuration sections specific to particular actions can be present in the same file.

More in depth documentation can be found on the OpmanTek Community Wiki [↗](#)

- `log.logtype()`
- `script.scriptname()`
- `escalate.policyname()`
- `email.contactname()`
- `systop.targetserver(priority)`
- `rminsyslog.targetserver(prio)`
- `priority.adjustment`
- `tag.tagname(value)`
- `acknowledget[]`
- `watchdog.set(waittime)`
- `watchdog.disable()`
- `element_watchdog.set(waittime)`
- `element_watchdog.disable()`

Console Output

One of the available actions is to notify using [email](#), and different templates can be defined in **EventEmails.nmis** file.