opEvents Normalised Event Properties

Introduction

opEvents processes syslog, SNMP Traps, NMIS Events into a common format for further processing. This process is called normalisation.

The following tables represents the *standard* properties of normalised events - but as event properties are pretty much infinitely adaptable and extensible (e.g. from custom parser rules or event action policies), the tables cannot be exhaustive.

Standard Properties

Event Property	Description	Example
eventid (`_id`)	A globally unique Event ID	`64d2192a0454aa024079fef4`
`time`	Unix time of the event (seconds since 1970). With decimal seconds.	`1691490602.72516`
`date`	~~The event time in human readable format~~ **Note: This Property has been deprecated**	`2023-08-08T20:30:02`
`node`	The name of the node in question. Normally the same as the NMIS node name.	
`host`	The DNS hostname or IP address of the node in question, as extracted from the input data. See section on Node vs. Host below for details.	
`event`	Name of the event	Node Down, Node Up
`element`	What element of the node the event refers to. Optional, but always present if `state` and `stateful` are present.	FastEthernet1, Neighbor 1.2.4.5
`state`	Is the state good or bad, up or down. Optional, but always present if `state` and `stateful` are present.	up/down, open/closed, etc
`stateful`	Name of the stateful object. Optional, but always present if `state` and `element` are present.	Node, Interface, OSPF Neighbor
`details`	Other event details	
`type`	Where did the event originate?	cisco_syslog, trap, NMIS, (remote) API
`escalate`	Has the event been marked for escalation? This is set if an action policy sets the event up for escalation, and cleared once the event is acknowledged.	0 or 1
`priority`	opEvents priority level, see opEvents priority levels vs. NMIS and Syslog levels	0 to 10, inclusive
`level`	nearest NMIS severity level, computed from `priority` (only in opEvents 2.2 and newer)	Normal to Fatal
`acknowledged`	Has the event been acknowledged?	0 or 1
`flap`	Is this event a flap?	0 or 1
action_required	Should the GUI show the event as open? Only present in opEvents versions up to (and including) 2.0.3.	0 or 1

planned_outage	opEvents looks up the node in the NMIS planned outage system and sets the value of planned_outage (default value) to be true or false if a planned outage is scheduled or not. Scheduled Outages or Maintenance Windows Only available in opEvents versions 4.1.1 and newer, Release Notes: opEvents 4.1.1	true or false

Optional but Common Properties

In addition to those a number of properties are optional and created only under certain conditions:

Event Property	Description	Example
`interface_description`	The ifAlias (or Description) of the interface in question • only available with opEvents versions 2.0 and newer, • only for interface-related stateful events (i.e. element is an interface), • and only if the node was refreshed or imported from NMIS with opEvents 2.0 or newer	
`authority`	The server name of the system that originated the event; Optional, only relevant for remotely/API-generated events, but plays an important role for reorder protection of stateful events.	
`location`	The URI for this event at the originating server. Optional, only relevant for remotely/API-generated events.	
`duplicateof`	List of Event IDs that this one is a duplicate of. Only present when programmable suppression rules affected this event.	
`nodes`	lists nodes that caused this synthetic event. Only present if this is a synthetic event.	
`eventids`	List of Event IDs that were involved in causing this synthetic event. (Only in opEvents versions 2.0.3 to 2.2.1 this is also set for relationships between events, e.g. for auto-acknowledged events the up event lists the down event's id here and vice versa.)	
`stateful_eventids`	List of Event IDs that are related to this stateful event, e.g. the preceding down event if this one is an up event and vice versa. Only present in opEvents versions 2.4 and newer, and only if this event is stateful.	
`delayedaction`	Unix time, until then the event is held back from processing for actions and policies	1385079231
`action_checked`	Has the event been processed wrt. actions and policies?	0 or 1

`script.<scriptname>`
 	
`script.<scriptname>_time`
	If an event triggered a script action that is set to save output, then the script output (and the script's execution time) is stored in these properties.	

`synthetic`	whether this event was created by a correlation policy action, or because a watchdog expired	0 or 1

triggerof	**Deprecated** as of opEvents 2.4, see trigger_eventids for replacement. Event ID of the synthetic event that this event was a trigger for. Only present in opEvents versions 2.0.4 to 2.2.1, and only if a correlation policy action identified this event as a trigger. If multiple policies apply, only the last trigger is stored.	
`trigger_eventids`	List of synthetic event IDs that this one is a trigger of. Only in opEvents 2.4 and newer, and only if one or more correlation policy actions have identified this event as a trigger.	
`watchdog`	Whether this is a watchdog expiration event	0 or 1
`escalation_age`	If the event is or was subject to escalation, then this property indicates the event's most recent escalation threshold. Note that this property is *not* cleared when the event is acknowledged and escalation terminates.	60, 900 etc.
`escalation_policy`	If the event is or was subject to escalation, then this property lists the event's most recently active escalation policy name. Like the previous property, this one persists after escalation terminates.	
`notes`	A list of originator- and time-tagged comments for this event (optional, supported in opEvents 2.0 and newer)	
`notes_context`	A title and text field which can be used as event notes, It is a modifiable field for both title and text.	`"notes_context": {"title": "Example note title","text": "The note text can hold up to 150 characters."}`
`tag_<anything>`	These enrichment tags are controlled by your action policy, and have no special meaning - with the exception of `tag_kb_topic`, which controls linking to online sources (in opEvents 2.0.2 and up), - and `tag_servicePriority`, which is shown with the event priority if present (only in opEvents 2.0.4 and up)	
status_history	A structured record of changes and activities related to the event.	
nodeinfo	A deep structure for copied node properties on event creation	

Additional event properties to capture additional event data

opEvents works on an event, the event can be thought of as a document and all the contents of that document move through opEvents together, additional properties are added and updated during event processing. It is also not only possible but strongly encouraged to add additional properties as the richer the event, the more useful it will be during processing, obviously the data captured should be relevant and useful. This is easily done during event parsing and a variable is created by including a new variable name in the capture statement, see more details in opEvents EventParserRules - Adding Rules For SNMP Traps.

During event processing by EventActions, you can tag events with the tag function, and use it for event processing and conditions. If an event is tagged like this:

```
'1' => {
        IF => 'event.details =~ "outage_current=true"',
        THEN => 'tag.outageCurrent(TRUE)',
        BREAK => 'false'
},
```

The result would then be available to be used in the variable event.tag_outageCurrent, e.g.

```
IF => 'event.tag_outageCurrent eq "TRUE"',
```

Using event tagging is a powerful way to implement event policies and have an easy to follow flow in EventActions.

Node vs. Host, and how opEvents handles inconsistent input data

opEvents works hard to normalize inputs from disparate and often inconsistent input data; one of the most common issues is event input data that lacks the correct node name, only has an IP adress or a DNS shortname and not an FQDN and similar.

Here is an overview of the heuristic that opEvents applies to make sure that events are associated with the correct node:

1. First, opEvents extracts the prospective nodes' host name from the input data.
   The parser normally extracts a `host` property from the input, which may be an FQDN, DNS short name or IP address.
   (The parser *may* also set the final `node` property, but that's **not recommended** and our example parsing rules don't contain such operations.)
   In case of the NMIS event log being the source, *only* the NMIS node name is available and is used as host name for all subsequent steps.
2. opEvents now looks for the one node that is identified by this hostname/address etc.
   This is done by looking in a cache of associations between FQDNs, shortnames and IP addresses that opEvents maintains.
   If this succeeds, then the node's name is set as the event's `node` property and the procedure is finished.
3. If no matching association is found, then the DNS is used to find potential intermediate associations.
   The `host` property is resolved via the DNS, forward to IP address if it was a hostname, or reverse to FQDN if it was an IP address.
4. If this intermediate step resulted in something that is associated with a known node, then that node is used and the intermediate info is added to the cache.
   (There are also some internal safeguards to reduce the number of DNS requests opEvents might perform.)
5. If none of this worked, then the original raw, untranslatable, `host` property is used as a new node's name.
   In opEvents 2.0.4 and newer, the configuration option `opevents_auto_create_nodes` controls the behaviour in this situation: if set to false, the event is skipped.
   If set to true (or not present), then a new node record is created. (Older versions of opEvents always created new but somewhat incomplete node records in this case.)

The consequences of this normalisation setup are as follows:

- An event's `event.node` property will always identify the node that is associated with the event.
- The event's `event.host` property is set from the raw inputs, and is *very often not the same* as a known node's `node.host` property, nor is it guaranteed to exist.
  The only thing guaranteed about `event.host` is that at the time of the event creation it was somehow, possibly even just temporarily, associated with the node in question.
  For NMIS eventlog as source the `event.host` is always the NMIS node name.
- The opEvents dashboard and event-centric pages all show the `event.node` property as primary id, and the `event.host` property as supplementary information.
- Editing a node's `host` or `addresses` properties in the Edit Nodes GUI does **not** affect any existing events; it just sets up association info for finding the right node for future events.