External Enrichment in opEvents

Overview

By default opEvents combines information from the actual input sources, NMIS and policy actions to populate an event with relevant details. However, for situations where the authoritative source of knowledge of nodes and environments is external to opEvents and NMIS, is also possible to collect such information from an SQL database. In addition to this enrichment mechanism for incoming events, opEvents (in version 2.0.2 and newer) also offers enrichment by linking events to external Knowledge Bases, WIKIs or other online sources.

Linking Events to Online Sources

opEvents 2.0.2 and newer provide a mechanism for displaying links to external sites for selected (or all) events. This feature is controlled by the following configuration settings in `conf/opCommon.nmis`:

```
'opevents' => {
  # ... lots of other settings
  # the search/wiki feature is active if search_link or kb_link or both are correctly configured
  'opevents_search_link_title' => undef, # both title and url are required to activate
  'opevents_search_link_url' => 'https://duckduckgo.com/html/?q=%s',
  'opevents_kb_link_title' => 'KB Lookup',
  'opevents_kb_link_url' => 'https://community.opmantek.com/x/%s',
},
```

- The `opevents_search_link_title` and `opevents_search_link_url` settings define whether a ***generic*** search link should be included on every Event Context page.
  If both settings are present, and the `...link_url` is a valid URL with one "%s", then the full event name replaces the `%s` in the URL and a link button with the given title will be shown.
  In the example above this feature is disabled by not giving a title.
- The `opevent_kb_link_title` and `opevents_kb_link_url` settings define whether a ***targetted*** link should be included on *matching* Event Context pages.
  Both settings must be present to activate the feature, **and** the event in question must have been tagged (with event action `tag.kb_topic(somevalue)`). If all of these criteria are met, then the `...link_url` gets the `%s` replaced by the event's kb_topic tag (the "`somevalue`" in the tagging action) and a link button with the given title is shown.
- If both variants are enabled, then the targetted link takes precedence.
  i.e. a targetted link is shown if possible, otherwise a generic search link is attempted, and if that isn't fully enabled no link button is shown.

Generic Search

The generic search is useful for delegating oft-repeated and static events to a general or internal search engine; e.g. would work well for looking up information for events like "CISCO-MAC-NOTIFICATION-MIB::cmnMacChangedNotification".

Targetted Lookup

The targetted lookup is meant for specific events or event classes, and to support operators by providing them with access to internal knowledge bases, action checklists and the like. In the example above the base link points to the Opmantek WIKI (actually to the base URL for URL-shortened links). To provide a link specifically for threshold-type events from NMIS, we would create an event action policy that includes the following clause:

```
12345 =>  { IF => 'event.event =~ /Proactive/', THEN => 'tag.kb_topic(rYKG)' },
```

which would construct and display a link to https://community.opmantek.com/x/rYKG - the wiki page for NMIS threshold configuration.

Enrichment of Incoming Events

At present opEvents only support MySQL databases for external enrichment. To use external enrichment it is necessary to define the relevant source databases and event actions that pull informaition from these databases.

The configuration file `conf/EventDB.nmis` contains the settings for the external databases, and is only consulted if the option `opevents_db_enrichment` (in `conf/opCommon.nmis`) is activated. Here is an example enrichment db setup:

```
"cmsdbA" => # db name cannot be be 'event' or 'node'
 {
 db_host=>"thor",
 db_port=>"3306",
 db_driver=>"mysql", # currently the only supported driver
 db_name=>"opexport",
 db_user=>"enrichment",
 db_password=>"poor",
 db_cache_age=>120, # how long until a query must be re-executed.

 # all node.X and event.Y occurrences on the right of the WHERE are replaced by
 # opEvent's node/event data.
 # queries therefore CANNOT use column names or table.column expressions like node.X
 # or event.Y in the WHERE expression.
 queries=> {
        "test1" => "SELECT * FROM nodeProperties WHERE node.name = name and property = 'util_out' ",
        "test2" => "SELECT * FROM nodeStatus WHERE from_unixtime(event.time) > lastUpdate and node.name = name
",
 },
},
```

The configuration above defines two access queries for use in an action policy, which will be rewritten at evaluation time: all tokens of the form `node.X` and `event.Y` will be replaced by the respective node and event properties. As external queries can be expensive and time-consuming, opEvents supports the caching of query results - please note, however, that caching depends on *all* the input values to the query and as a consequence some queries like example `test2` above are not cacheable in practice.

Making use of an external enrichment database takes place exclusively in an event action rule's `IF` clause, like in the following example:

```
# syntax: dbname.queryname.column_to_return
'30' => {
         IF => 'cmsdbA.test1.id > 10 AND cmsdbA.test2.name eq node.name',
         THEN => "log.tmp()"
}
```

In the example action rule, the expression `cmsdbA.test1.id` is interpreted as "execute query `test1` on database `cmsdbA` with the details from this event and node, and return the column `id` from the result", which is then used in subsequent comparisons.