System Events signalling

System Event Signal messages need setting-up for the environment and mostly requires the definition of the remote targets and priority.

The System Events configuration file is **`system_events.conf`** and its customizable copy resides in `/opt/yce/etc`. If it is missing the distribution version located in `opt/yce/system` is copied to the `etc` directory.

This configuration file consists of two parts, the top section defines the events and what effect they have. The second part defines the event signal types. This article explains the second section.

The file is formatted in the Perl syntax and contains the definition of the two hashes making up the aforementioned sections. The **`$system_signals`** hash has the attributes associated with the four signal types: 'syslog', 'trap', 'email', and 'rest-api'.

The configuration of the System Events signalling is very similar to the configuration of the Compliance signalling which uses the file `signal_cmpl.conf`. Both configuration files need to set up separately.

The system-event messages are determined by the system monitors and generally use formats similar to these examples:

```
CLEAR: Replication status - Primary db 'devel6b' in-sync with Secondary
'devel6a'
SWITCH: Database connection - Secondary db 'devel6b' failed, activating
Primary 'devel6a'
CLEAR: System debug mode is inactive
```

Apart from the descriptive message the Event has some additional attributes which will be included in the signal if the format supports it.

Traps

An SNMP (v2) trap has the following parameters that can be configured:

- **enterprise:** the traps's enterprise, a single number. We append this to “1.3.6.1.4.1.”.
- **specific:** the trap's specific number
- **varoid:** the trap's varoid
- **community** the trap community string
- **tmsg:** the system-event will create its own trap-message. Include your own message to override it. The system-event message will still be included in the trap using higher oid.
- **nmsaddr:** the addresses(es) to send the trap to. Add all trap-destination ip-addresses to this array.

The relevant entries in the `system_events.conf` file under `$system_signals` are:

```
# Available values:
# - trap
```

```
#    - enterprise        # defaults to '696'
#    - specific          # defaults to '0'
#    - varoid            # defaults to '1.0'
#    - community         # defaults to 'public'
#    - tmsg              # optional trap-message override. use undef to use
default system event messages
#    - messages (array)  # optional text lines to be included as individual
oids
#    - nmsaddr (array)   # a list of ip-addresses to send trap to. Defaults
to Common::traphosts

our $system_signals = {
    'trap' => {
        'enterprise' => '696',
        'specific'   => '0',
        'varoid'     => '1.0',
        'community'  => 'public',
        'tmsg'       => undef,
        'messages'        => [
            "additional messages line one",
            "additional messages line two",
        ],
        'nmsaddr'    => [
            '127.0.0.1',
            '2.3.4.5',
            '3.4.5.6',
        ],
    },
    #
    # ... other signal-type definitions
    #
};
```

The trap will also include additional attributes associated with the event:

- class
- state
- name
- msg_type
- severity
- user_level
- signal_type
- signal_int

The values for these attributes include the name in the “<attribute>: <value>” format.

Syslog

The system-event call sends a syslog message to one or more remote syslog receivers using a configurable syslog facility and priority.

All syslog messages are assigned a 'facility' (category) and a 'priority' (severity):

```
Valid 'facilities' are:
  kern user mail daemon auth syslog lpr news uucp cron authpriv ftp
  local0 local1 local2 local3 local4 local5 local6 local7

Valid 'priority' values are (ordered high to low):
  emerg alert crit err warning notice info debug
```

The format of the syslog message:

```
<timestamp> host-ip [hostname] [event|state] [priority]: <message>

Example:
Mar 26 10:33:03 172.17.10.29 [devel6b] [replication_status|warning] [err]:
WARNING: Replication status - Primary db 'devel6b' out-of-sync with
'devel6a'
```

The relevant entries in the `etc/system_events.conf` file are:

```
# - syslog
#    - facility          # defaults to 'daemon'
#                        # one of:
kern|user|mail|daemon|auth|syslog|lpr|news|uucp|cron|authpriv|ftp|
#                        #
local0|local1|local2|local3|local4|local5|local6|local7
#    - priority          # defaults to 'warning' - needs mapping from
severity
#                        # one of:
emerg|alert|crit|err|warning|notice|info|debug
#    - priority_map      # translation hash of numeric 'Severity' to
'priority' string, overrides 'priority'
#    - nmsaddr (array)   # a list of ip-addresses to send syslog to.
Defaults to Common::traphosts

our $system_signals = {
    'syslog' => {
        'facility'   => 'daemon',
        # translate the event severity to the desired priority
        'severity_map' => {
            # '6' => 'emerg',
            # '5' => 'alert',
            '4' => 'crit',
            '3' => 'err',
            '2' => 'warning',
```

```
            '1' => 'notice',
            # '0' => 'info',
            # '1' => 'debug',
        },
        # the default priority
        'priority'   => 'warning',
        'nmsaddr'    => [
            '127.0.0.1',
            '172.17.10.20',
            '172.17.10.28',
        ],
    },
    #
    # ... the other signal-types
    #
};
```

The 'facility' is directly configured using its name. For the 'priority' a mapping can be made from the numeric event **severity** to the desired syslog 'priority'. When the mapping does not result in a match, the default 'priority' setting is used.

Email

The System-event can send an email message to one or more email-addresses.

The relevant entries in the `etc/system_events.conf` file are:

```
# - email
#    - subject           # defaults to 'NetYCE system event'
#    - mail_to           # array of mail addr. defaults to 'yce@<server-
fqdn>'
#                        #  may also be space or comma separated string of
mail addr
#    - mail_from         # defaults to 'yce@<server-fqdn>'
#    - messages (array)  # optional text lines to be included as email
trailer
#

our $system_signals = {
    'email' => {
        'subject'   => 'NetYCE system event',
        'mail_to'   => [
            'yce@nms.netyce.org',
        ],
        'mail_from' => 'yce@netyce.org',
        # optinal lines included in the email trailer
        'messages'  => [
```

```
            "additional messages line one",
            "additional messages line two",
        ],
    },
    #
    # ... other signal-types
    #
};
```

The `mail_to` configuration accepts either an array (set between '[..]' with individual quoted email-addresses) or as a string (a single quoted list of email-addresses). The use of the array format is recommended.

The `mail_from` configuration must contain a single quoted email-address.

The `subject` value is prepended to the system-event message. The resulting email subject has the format:

```
subject format:
  <subject> [<Hostname>] [<Name>|<Severity>] <message>";

example subject:
  NetYCE system event [devel6] [replication_status|warning] WARNING:
Replication status - 'devel6' down, no synchronization
```

The mail body will also include additional attributes associated with the event:

- class
- state
- name
- msg
- msg_type
- severity
- user_level
- signal_type
- signal_int

The values for these attributes include the name in the “<attribute>: <value>” format.

The mail body uses the format:

```
This is an automatically generated notification
message, please do not reply
----------------------------------------

<subject> <event-message>
  ::
  :: ... list of event attributes
  ::

----------------------------------------
```

```
::
:: ... optional trailing messages
::
----------------------------------------

Issued by <server> at <current-timestamp>
```

Example:

```
This is an automatically generated notification
message, please do not reply.
----------------------------------------

NetYCE system event [devel6] [debug_status|active] NOTICE: System debug mode
is enabled by 'Eric Yspeert'
  class: debug_state
  state: active
  name: debug_status
  msg: NOTICE: System debug mode is enabled by 'Eric Yspeert'
  msg_type: 2
  severity: 1
  user_level: 3
  signal_type: syslog,email
  signal_int: 1800
----------------------------------------

additional messages line one
additional messages line two
----------------------------------------

Issued by devel6.netyce.org at 2020-03-26 11:43:35
```

The resulting mail is sent as plain text to each of the addresses in the mail_to list.

Rest-Api

The Rest-api call is a POST request in JSON format to a single remote host at a specific url. The REST call has the following parameters that can be set:

- host: The host ip or fqdn to post to. Has to include the transfer protocol (“http:” *or “https:*”) and the port number
- url: The URL path of the Rest-api to post to. Must have a leading `/`
- webtoken: Optional bearer AUTH's webtoken
- username and password: optional; basic authentication username and password

The relevant entries in the `system_events.conf` file are:

```
# - rest-api
#    - host              # host web address eg:
'https://devel6.netyce.org:8080', 'http://192.17.10.28:8080'
#    - url               # Rest-api url part of address:
'/api/v1/nms_events'
#    - username          # basic authentication username part or undef for
none
#    - password          # basic authentication password part or undef
#    - webtoken          # webtoken authorization
#    - parameters        # optional struct of data to include
#

our $system_signals = {
    'rest-api' => {
        'host' => 'http://nms.netyce.org:8080',
        'url' => '/api/v1/nms_events',
        'username' => undef,
        'password' => undef,
        'webtoken' => undef,
        # optional struct of data
        'parameters' => '',
    },
    #
    # ... other signal-types
    #
};
```

The optional `parameters` value can be any perl-struct that is to be included in the POST. It is converted to JSON for this purpose.

The submitted JSON structure will have various attributes. The attribute `message` has the system-event message.

The format of this message:

```
  [<server-name>] [<event-name>|<event-state>] <message>
```

The JSON will also include additional attributes associated with the event:

- class
- state
- name
- msg
- msg_type
- severity
- user_level
- signal_type
- signal_int

The `parameters` attribute will have the structure and values from the config file.