Table of Contents

System Events configuration

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 top section.

The file is formatted in the Perl syntax and contains the definition of the two hashes making up the aforementioned sections. The **`system_events`** hash has the attributes associated with the events coming from the various system monitors.

This hash has three levels, the first level is referred to as the *CLASS* of the event, the second level its *STATE* and the third level has the various *ATTRIBUTES* and values for the event state. There is a structure like this for every System event that is generated by one of the monitors.

```
#-------------------------------------------------------------
# CLASS  STATE  ATTR          VALUE
# db_conn_state => {
#        name =>              connection_status  # the 'database_name' this
class is mapped to
#        error => {                             # defines the 'high' state
setting for this class
#               msg_type =>   3                 # message will show at pre-
login (1), post (2) or both (3)
#               severity =>   4                 # event impact (and color)
as set in lookup
#                                               # 0=clear, 1=low .. 4=high
#               user_level => 2                 # minimum user-level to
display message
#                                               # 0=everyone, 1=browser ..
5=manager
#               signal_type => 'syslog, trap'    # none | syslog | trap |
email | rest-api
#               signal_int =>  120              # interval in seconds to
repeat event, 0=once
#-------------------------------------------------------------
```

Attributes explained

As Events are generated by the monitors, three vales are passed into the System Events.
The first is the CLASS of the event. This is the internal name of the monitor. In the example above it is the `db_conn_state`.
The second is the STATE of the event. Its value depends on the monitor and what states it can be in. In the example is shown the `error` state. As this specific monitor reports on the connection status to the database the state can have three values: 'error', 'switch', and 'clear'. These correspond to the 'no database connection', 'switched to the secondary database' and '(re-)established database connection to primary'.

The third and last value of the event is the message. This message will be used in the notifications and signalling to inform what the event is all about.

The customization of the events in this configuration will usually only affect the attributes of the event. They specify for who, where and how the notifications are displayed and which signalling takes place. As these attributes are defined per monitor and per state, the event notifications can be tailored to a high degree.

Attribute	Example	Description
msg_type	3	The message-type determines if the notification-bar will show the event on the login screen (1) only, after login (2) only, or both (3). When events are to be displayed before login it will be displayed for all users, the user_level setting has no effect before logging in
severity	4	The numeric value of the severity-level of this event is used to perform a lookup to find the severity name and colour of the severity. See the paragraph below on “Severity customization”.
user_level	2	Not all messages are useful for all users. Some are relevant only for 'manager'-level users where others are also relevant for 'engineer'-level users. The user-level attribute defines the lowest user-level that will get the notification. These levels correspond to the NetYCE operator levels
signal_type	'syslog,email'	The value is either 'none' for no external signalling or a string with one or more of the values: syslog, trap, email, rest-api. In the example the signalling will send out syslog(s) and an email if this event occurs
signal_int	1800	When this value is set to '0', the signals are sent only once for this event. To get periodic signals when the state is still unchanged, set it to the number of seconds to receive a 'reminder' of the event. The signalling interval is the same for all four signalling types.

Class mapping

One additional setting in the Events configuration has been ignored - up to now. The **`name`** attribute at the 2nd level plays an important role in the way event monitors can be combined to form a single status. This `name` value is what the CLASS name will be translated to to form the notifications and signalling.

Where `db_conn_state` is the name of the monitor, the events will set the `connection_status` events. And similarly will the class `db_sync_state` generate `replication_status` events. Thus we get two distinct statuses for the DB connection and the DB replication. For these two monitors that was the way it was designed but for other monitors for e.g. cpu and disk usage these could very well be combined in a single event status for 'system resources'.

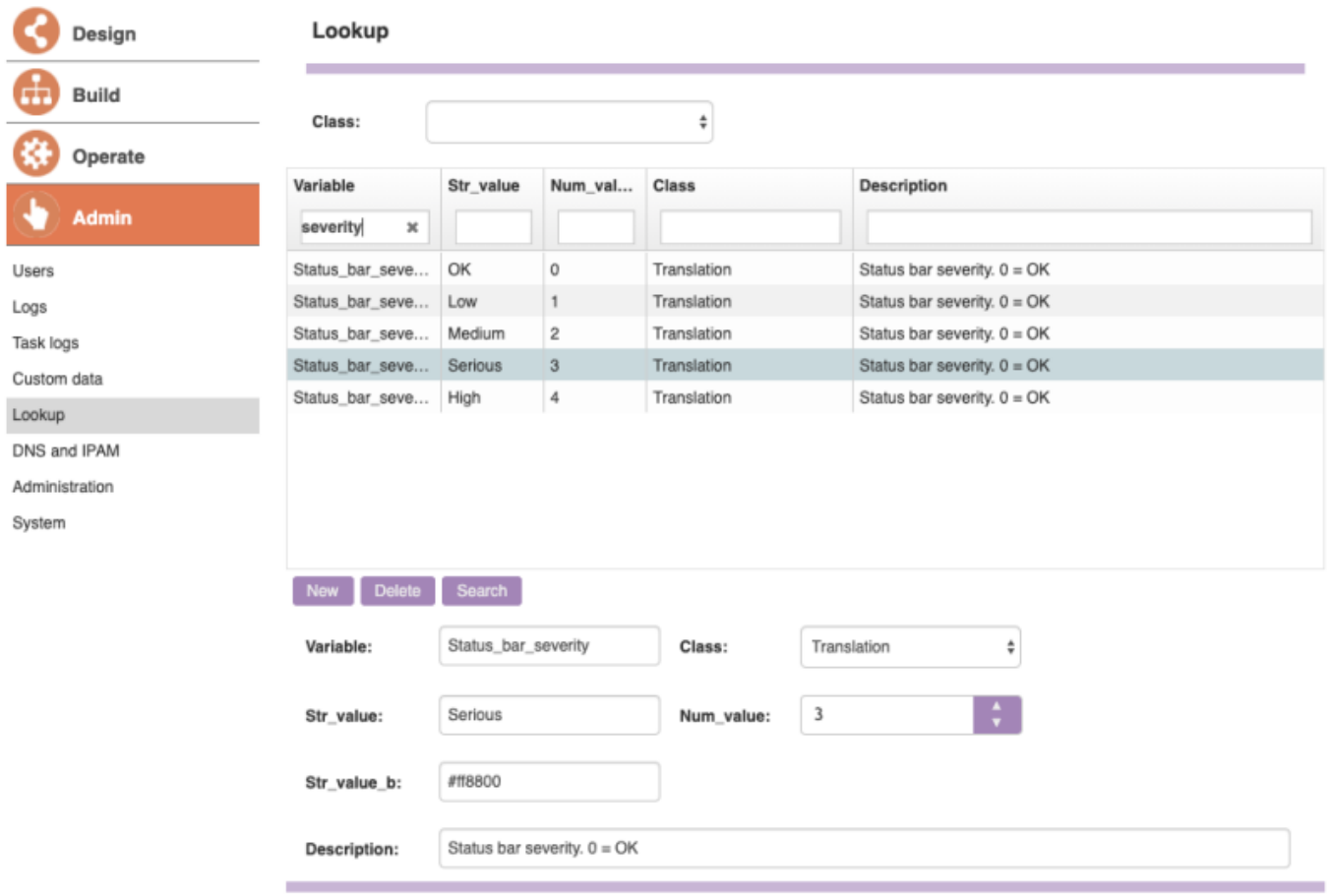
It is suggested not to modify these monitor class to event status mappings unless there is good understanding of the various states each monitor produces.

Severity customization

By default four severity levels exist to assign to System events: 'Low', 'Medium', 'Serious', and 'High'. A fifth, 'OK', should be considered a reserved value indicating all is 'ok'.

Using the 'Admin - Lookup' tool, these levels can be customized and extended as required. The entries using variable `Status_bar_severity` and class `Translation` are used to assign the severity level number to its name (3 → Str_value='Serious') and its notification color (3→ Str_balue_b='#ff8800'). Both values can be modified as can the number of severities.

The severity numeric value is used to assign to the `severity` attribute value in the configuration file section for the system_events.



Syntax check

CLI changes

After making modifications it is recommended to check the syntax of `system_events.conf`:

```
$ cd /opt/yce/etc
$ go chk system_events.conf
-- perl -cw current dir
system_events.conf syntax OK
```

System configuration tool

If changes to the configuration file were made using the 'Admin - System - Edit configs' tool, the syntax is automatically checked for errors. Only when the file syntax is valid will the file history be updated and the configuration be activated.

Since most system-monitoring is performed by the **`yce_skulker`** daemon, this process needs to be restarted after changes to the configuration were made. This can be done on the cli (`go restart`

`skulker`) or by clicking the `Stop yce_skulker` button in the 'Admin - System status' tool. It will be restarted automatically by the process monitor daemon (yce_psmon) within 20 seconds.