snmd

Apr 01, 2021

Contents

Contents:

CHAPTER 1

About

snmd uses HTML5 and JavaScript code to build dashboards visualizing monitoring and process states. Any data to be visualized needs to be available by a MQTT subscription.

1.1 3rd party libraries

The following JavaScript libraries are used by *snmd* via *bower*:

- alameda 1.1.0
- favico.js 0.3.10
- Font Awesome 4.7.0
- howler.js 2.0.2
- jQuery 2.2.4
- js-cookie 2.1.3
- js-logger 1.3.0
- Open Sans @font-face kit 1.4.2
- Push 0.0.12
- qTip^2 3.0.3
- require-css 0.1.8
- sprintf.js 1.0.3

The following JavaScript libraries are minified and embedded since missing *bower* support:

- jQuery SVG 1.5.0
- JSON.minify
- Paho JavaScript Client 1.0.2

- SVGPathData 1.0.3

1.2 Authors

- Thomas Liske <liske@ibh.de>

1.3 Copyright

- 2012 - 2013 (C) Thomas Liske [https://fiasko-nw.net/~thomas/]
- 2014 - 2017 (C) IBH IT-Service GmbH [https://www.ibh.de/]

1.4 License

CHAPTER 2

Architecture

The following figure shows the software stack required to deploy a SNMD environment:

Fig. 1: Software stack required by SNMD.

2.1 Data Aquision

The *Data Aquision* layer collects data and publishes them to a MQTT message broker. The collected data can be in any (binary) format as long there is a approperiate widget library for SNMD to do the visualization.

2.1.1 nag2mqtt

The nagios plugin nag2mqtt can be used to publish nagios state and performance data to a MQTT message broker using JSON. SNMD ships with a corresponding widget library (*snmd-widgets-nagios*).

2.2 MQTT Message Broker

A message broker is required to make the data of the DAQ layer available to the SNMD frontends. The broker needs to provide a web socket port since the frontend is a native HTML5 app. The usage of Mosquitto as MQTT broker is recommended.

2.3 Web Server

A simple web server is required to serve static files:

- SNMD package

- config files (JSON)
- views (SVG)

2.4 Frontends

Any HTML5 enabled browser should work (read: IE is not a (HTML5) browser). For 3D animation it's recommended to use a WebKit based one, the Gecko engine seems to have poor performance on CSS transformations.

Example of frontend devices:

- *Chromium 49+* on Desktop
- *Firefox 31+* on Desktop (3D not recommended)
- *Raspberry Pi* using *barkery* (*Raspbian Stretch* required)

2.4.1 barkery

Barkery is a small perl script using GTK and WebKit to implement a kiosk browser window in full screen. The package can be easily installed on any recent Debian based distribution using systemd. On system boot *barkery* will be auto started and it works with read-only filesystems.

This is the prefered frontend if using a *Rasperry Pi*. To be able to manage a large number of frontends *barkery* is remote controlable via MQTT.

CHAPTER 3

Installation

3.1 Prerequisites

In order to rollout SNMD you require different components at the target environment.

3.1.1 MQTT message broker

A MQTT v3.1 message broker with **web socket support** is required. The usage of Mosquitto is recommended. More details on deploying *Mosquitto* can be found in *Appendix: Mosquitto Setup*.

3.1.2 MQTT publishing

Data needs to be published on MQTT so it can be visualized by SNMD. Look at nag2mqtt to publish Nagios performance data to MQTT.

More details on deploying *nag2mqtt* can be found in the nag2mqtt docs.

3.1.3 Web server for SNMD

A web server publishing the static SNMD files is required. For more advanced setups using *https* and *HTTP Basic Authentication* the use of nginx is recommended.

More details on deploying *nginx* can be found in *Appendix: Nginx Setup*.

3.1.4 Bower

To download the external JavaScript dependencies of SNMD the installation of bower is required.

3.2 Installation

3.2.1 Preparation

Install *git*, *nodejs*, *npm* and *bower* on the web server. On *Debian GNU/Linux* use:

```
# apt-get install git nodejs nodejs-legacy npm
# npm install --global bower
```

Hint: The package *nodejs-legacy* is required as the *bower* command uses the legacy shebang `/usr/bin/env node` to run *nodejs*.

3.2.2 Download or clone SNMD

Download the latest release archive or clone the development repository using *git*:

```
$~/ git clone https://github.com/liske/snmd.git
```

You need to pull the *bower* dependencies in your local SNMD directory:

```
$~/snmd/ bower update
```

This will download the *snmd-core* and *snmd-widgets-nagios* components including any 3rd party libraries required by them.

CHAPTER 4

Configuration

Hint: It is strongly recommended to *not* touch or add any files in the source directory of SNMD. Instead you should configure your web server to redirect files access to another directory (see also *Appendix: Nginx Setup*) where you keep your local stuff.

Three different JSON files are used to configure SNMD's behavior.

Hint: C-like comments within the JSON config files are stripped using JSON.minify before the files are parsed. No other deviations from the JSON specification is allowed and will result in a fatal parsing error.

4.1 Global config

The SNMD configuration file `config.json` configures the included snmd widget library packages and some global configurations.

```
{
    /* Enable development mode.
       FALSE
         - use minified js files (snmd-core and widgets)
         - set loglevel to info
         - allow js caching by using a version depending
           urlArgs parameter (requirejs)
       TRUE
         - use non-minified js files (snmd-core and widgets)
         - set loglevel to debug
         - prevent any caching by using a full dynamic urlArgs
           parameter (requirejs)
    */
    "snmd_devel": false,
```

(continues on next page)

(continued from previous page)

```
    /* Load widget packages
       Each entry requires a prefix and a library value:
       - prefix: string used in SVG to reference a widget of this library
       - package: the bower package name
    */
    "snmd_widgets": [
        {
            "package": "snmd-widgets-nagios",
            "prefix": "Nagios"
        }
    ]
}
```

4.2 Terminal config

While loading SNMD a terminal name can be passed using the *config URL Parameters*. SNMD will try to load the terminal's configuration from `configs/${TERMINAL}.json`. SNMD will fallback to `configs/default.json` if it is unable to load the terminal file or no parameter has been passed. Terminal names can be used to load a predefined views configuration without user interaction.

```
{
    "view": {
        "title" : "Monitoring",
        "json" : "views/view1.json"
    },
    "vlinks": [
        {
            "name": "terminal1",
            "title": "NOC"
        },
        {
            "name": "terminal2",
            "title": "Bathroom"
        }
    ]
}
```

The terminal configuration is a object with the following keys:

- **view** (required) - A object configuring the path to the views config *json* and the *title* shown in the UI.
- **vlinks** (optional) - An array of objects which are added as hyperlinks so the user can switch to alternative terminal configs.

4.3 Views configuration

The views configuration referenced by terminal configurations are a JSON structures defining the views visible at the SNMD UI. These views configs are located in the `views/` directory.

```
[
 { "title": "Check_MK", "url": "/cmk-rproxy/dashboard.py?name=snmd", "render": "html",
↪ "reload": 300 },
```

(continues on next page)

(continued from previous page)

```
 { "title": "Core",     "url": "svg/LAN-Core.svg" },
 { "title": "Distri",   "url": "svg/LAN-Distri.svg" },
 { "title": "Access",   "url": "svg/LAN-Access.svg" }
]
```

The JSON structure is an array of objects with the following keys:

- **title** (required) - The label of the view used for the navigation bar.
- **render** (optional) - The rendering type of the view. Defaults to 'svg'. The 'html' renderer will load a html site in an *iframe* to embed arbitrary websites into the SNMD UI. Could be used to embed a state or event dashboard of your network monitoring system.
- **url** (required) - URL to load the content of the renderer (i.e. path to the SVG file).
- **reload** (html; optional) - Reload the *iframe* content periodically (value in seconds).

Hint: If you want to embed the nagios or Check_MK dashboard you might want to use a reverse proxy to inject an HTTP authentication header for read-only access to your monitoring dashboard with-out any user query (i.e. digital signage).

CHAPTER 5

Views & Widgets

The visualization of SNMD is based on SVG files. It is highly recommended to use Inkscape to create those SVG files.

Hint: SNMD's color scheme is based on the SOLARIZED palette of Ethan Schoonover. SNMD provides a GIMP palette file which can be used with Inkscape. Put the snmd.gpl into *~/.config/inkscape/palettes/* and after (re)starting *Inkscape* you are able to select the new palette *Solarized & SNMD Dark*.

A view is a common SVG file. When the SVG file is loaded by SNMD it will replace any SVG element matching the special ID pattern by a corresponding SNMD *widget*. SNMD widgets may replace the original SVG elements completly, change their style, do some CSS transformations or replace their text content. Widgets are provided by widget libraries and based on one of the basic widget implementations:

5.1 Widget Usage

Any SVG element where the ID begins with the string `snmd_` will be treated as a SNMD widget. The behavior of a widget is configured using a `desc` child element.

This is an example widget which will plot the interface bandwidth monitored by Nagios. The escaping of quotes has been removed to improve readability:

Listing 1: *example widget configuration*

```
<rect
 id="snmd_mywidget1"
 y="..."
 x="..."
 height="..."
 width="..."
 style="...">
<desc>
```

(continues on next page)

(continued from previous page)

```
    {
        "type":"Nagios:Chart-IfBw",
        "topics":[
            "nagios/checks/switch-1.demo.lan/Interface TenGigabitEthernet1/1"
        ]
    }
</desc>
</rect>
```

Hint: Remember that the text content of the `dect` element is interpreted as JSON. The JSON syntax does **not** allow a comma after the last element within arrays or objects. SNMD will not be able to use the widget if there is an JSON syntax error. Use your browser's debugging console to check for parsing errors if widgets do not work as expected.

The *Object Properties* editor of *Inkscape* is the prefered way to set the SVG element ID and edit the `desc` element.

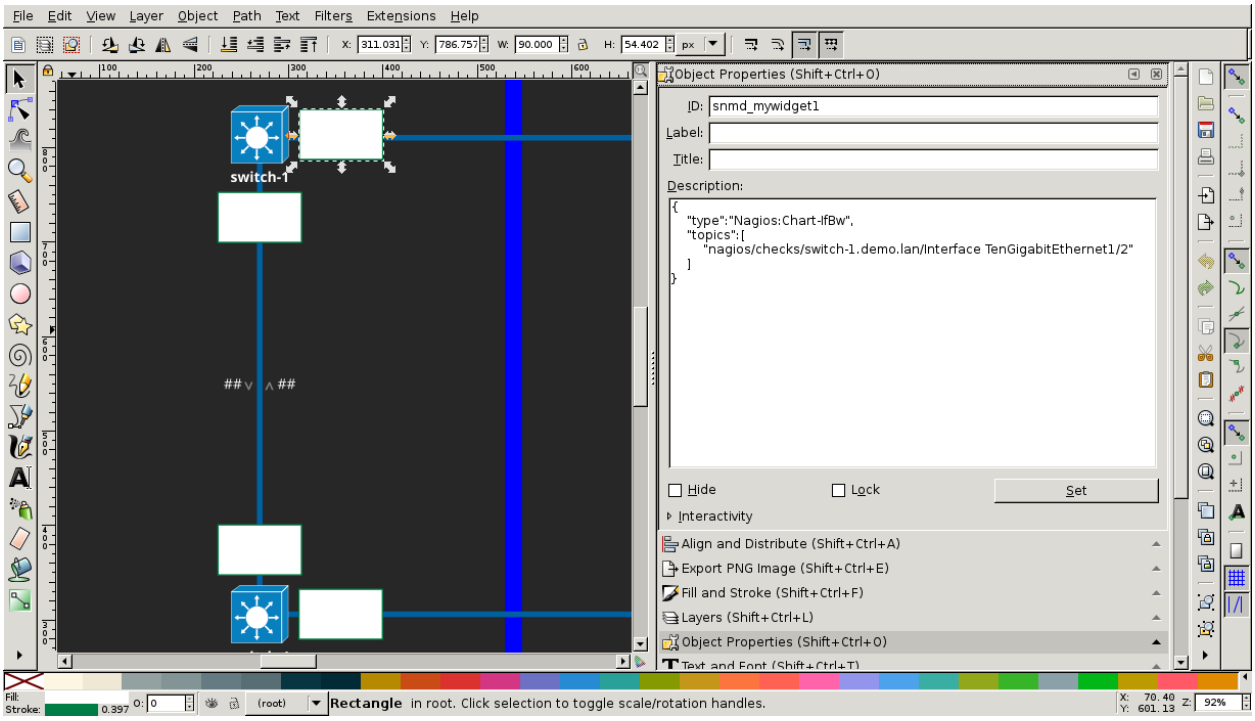

Fig. 1: *Object Properties* editor

5.2 Widgets Implementations

This is the list of available widget implementations. They can't be used directly but all widgets provided by *widget libraries* do use them. Some of the implementations have common configuration options to change their layout or behavior.

All widget do support the following configuration options:

type

(required) This defines the widget type to be used. The value is the concatination of the widget library names space, a literal colon and the name of the widget class.

Example: `Nagios:Chart-IfBw`

bcls

(optional) An array of additional CSS base classes of this widget for *Customizing Style*.

scls

(optional) An array of additional CSS state classes of this widget for *Customizing Style*.

5.2.1 Chart

- **SVG Element:** *replaced*

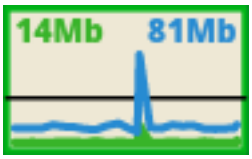

Fig. 2: *Chart* widget plotting interface bandwidth usage over time

Chart widgets are used to plot time series monitoring data.

5.2.2 Class

- **SVG Element:** *kept*

Class widgets get CSS classes applied depending on monitoring states.

5.2.3 Gauge

- **SVG Element:** *replaced*

Fig. 3: *Gauge* widget (arc) showing storage allocation

Gauge widgets will show a radial gauge for a monitored value.

5.2.4 Gradient

- **SVG Element:** *replaced*

Gradient widgets will show a *linear* gradient where the gradient stops are replaced by monitoring values. The monitoring values are mapped to a color using HSL a like coloring scheme.

Fig. 4: *Gradient* widget showing cabinet temperatur distribution

5.2.5 RadialGradient

- **SVG Element:** *replaced*

RadialGradient widgets are simular to *Gradient* widgets but use a radial gradient.

5.2.6 StrokeWidth

- **SVG Element:** *kept*

StrokeWidth widgets will clear the SVG element's stroke property and replace it by the current monitoring value.

5.2.7 Text

- **SVG Element:** *kept*

Fig. 5: *Text* widget showing current temperature

Text widgets will replace the SVG element's text content by the monitoring value.

5.2.8 Transform

- **SVG Element:** *kept*

Fig. 6: *Transform* widget (light blue rectangle) used as linear gauge

Transform widgets will apply CSS transformations depending on monitoring values.

Hint: SNMD is view in dark mode (default) and can be switched to light mode. Although SNMD uses CSS variables which can be easely redefined to switch between *Solarized Dark* and *Solarized Light* colors it is not possible to use them for the SVG files since *Inkscape* does not support to use CSS variables(, yet).

CHAPTER 6

Widget Libraries

6.1 Usage

Widget libraries can be added by installing them (i.e. using *bower*) and configure *SNMD* to load them using a namespace of your choice. The libraries need to be added to the *snmd_widgets* option within the global *config.json* file:

The example will load the bootstrap code of the package *snmd-widgets-nagios* and make it's widgets available under the namespace `Nagios`. They can be referenced in views using the following notation for the *type* parameter:

- `Nagios:Chart-IfBw`
- `Nagios:Chart-DiskTp`
- `Nagios:Text-PerfData`
- ...

6.2 Available Libraries

The following libraries are available:

- snmd-widgets-nagios - visualize state and performance data from *nagios* and *Check_MK*

CHAPTER 7

Customizing Style

The SNMD UI loads additional CSS definitions from `css/local.css`. This allows customizing the layout of widgets using CSS selectors.

The following example uses the `Class-State` widget type from the snmd-widgets-nagios library and changes the stroke color of the state `OK` to *DarkBlue*:

Listing 1: *widget configuration*

```
{
    "type": "Nagios:Class-State",
    "bcls": [
        "snmd-bcl-opacity-stroke", "my-bcl-test"
    ],
    "clrsty": [
        "stroke"
    ],
    "topics": [
        "nagios/checks/foo.demo.lan/Service Test"
    ]
}
```

Listing 2: `css/local.css`

```
.snmd-bcl-opacity-stroke.my-bcl-test.snmd-scl-0 {
    stroke: DarkBlue;
}
```

Hint: Remember that SVG elements have some custom CSS attributes. You need to use the `stroke` and `fill` properties to change their color, they do not have a `color` or `background` property. An overview of styling properties can be found in the Scalable Vector Graphs W3C Recommendation.

CHAPTER 8

URL Parameters

By default the SNMD UI loads the default *Terminal config* and uses default UI settings. You change the terminal config name and the UI settings using the following URL query parameters.

8.1 List of parameters

config

The name of terminal configuration (default: `default`).

3d

UI setting: *toggle 3D view*

- `0` Disable 3D view (increases performance).
- `1` Enable 3D view (decreased performance).

solarized

UI setting: Switch solarized theme.

- `0` Switch theme to solarized dark.
- `1` Switch theme to solarized light.

volume

UI setting: Toggle alert sounds.

- `0` Enable notification sounds.
- `1` Disable notification sounds.

notify

UI setting: Toggle browser notifications.

- `0` Enable desktop notifications.

- `1` Disable desktop notifications.

rotate

UI setting: Toggle interval or continuous view rotation.

- `0` Rotate view after activity timeout.
- `1` Keep current view.
- `2` Continuous view rotation.

follow

UI setting: Toggle current view follows state changes.

- `0` Switch to views if there state changes at least warning.
- `1` Do not switch to views due to state changes.

8.1.1 Example

To load the terminal config `wall` and change the *rotate* setting to mode `2`:

```
http://snmd.foo.lan/?config=wall&rotate=2
```

8.2 Fragment identifier

The view which should be shown initially when the SNMD UI is loading can be specified. The URL needs to be sufffixed by a hash mark `#` and the view's ID. The first view has the ID `1` and the IDs are counted from left-to-right.

8.2.1 Example

To initially load the second view:

```
http://snmd.foo.lan/?config=rpi#2
```

CHAPTER 9

Appendix: Mosquitto Setup

9.1 Installation

Take a look at the Mosquitto Downloads page to find out howto install *Mosquitto* on your system. On *Debian GNU/Linux* it is sufficient to install *Mosquitto* via *apt*:

```
# apt-get install mosquitto mosquitto-clients
```

The *mosquitto-clients* package is optional but helps debugging by subscribing to MQTT topics.

9.2 Configuration

You need to explicitly enable web socket access in *Mosqitto* and allowing anonymous MQTT (read) access is recommended:

Listing 1: /etc/mosquitto/conf.d/mosquitto.conf

```
allow_anonymous true
password_file /etc/mosquitto/users.pw
acl_file /etc/mosquitto/users.acl
listener 1883
listener 9001
protocol websockets
```

Hint: If SNMD is exposed to the internet you should setup a reverse proxy which enforces authentication (see also *Appendix: Nginx Setup*) and prevent direct MQTT access.

The ACL file `/etc/mosquitto/users.acl` defines subscribe and publishing access on MQTT topics. The following example allows anonymous subscribe access to all topics and publishing access for the `nag2mqtt` user to all topics below `nagios/`.

Listing 2: /etc/mosquitto/users.acl

```
topic read #
user nag2mqtt
topic readwrite nagios/#
```

The users password file can be created using `mosquitto_passwd` command:

```
# touch /etc/mosquitto/users.pw
# mosquitto_passwd /etc/mosquitto/users.pw nag2mqtt
```

CHAPTER 10

Appendix: Nginx Setup

It is recommended to use a reverse proxy to publish SNMD and the MQTT websockets. This enables you to protect the site by *https* and use HTTP basic authentication.

10.1 Installation

You need to install *nginx* - on *Debian GNU/Linux* use:

```
# apt-get install nginx
```

10.2 Configuration

The following example configuration for *Nginx* publishes SNMD at `http://HOSTNAME/snmd/`:

Listing 1: `/etc/nginx/site-enabled/default.conf`

```
# Mosquitto websocket upstream
upstream be-mqtt-ws {
        ip_hash;
        server 127.0.0.1:9001;
}

# configuration of the server
server {
    # the port your site will be served on
    listen      80 default_server;

    charset     utf-8;

    # optional: enable HTTP basic auth (Https is recommended!)
```

(continues on next page)

(continued from previous page)

```
    #auth_basic "SNMD";
    #auth_basic_user_file /etc/nginx/snmd.htpasswd;

    # forward MQTT-via-Websocket to Mosquitto
    location /snmd/mqtt {
        proxy_http_version 1.1;
        proxy_cache_bypass 1;
        proxy_no_cache 1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_pass http://be-mqtt-ws/;
    }

    # local SNMD styles, configs, svg files and views
    location /snmd/css/local.css {
        alias /opt/snmd-local/css/local.css;
    }
    location /snmd/configs {
        alias /opt/snmd-local/configs/;
    }
    location /snmd/svg {
        alias /opt/snmd-local/svg/;
    }
    location /snmd/views {
        alias /opt/snmd-local/views/;
    }

    # vanilla SNMD sources
    location /snmd {
        alias /opt/snmd/;
    }
}
```

You need to configure SNMD to use the correct URL to acess the published MQTT websocket. The terminal configuration requires the `mqttws_uri` option:

Listing 2: `configs/default.json`

```
{
    "mqttws_uri": "ws://HOSTNAME/snmd/mqtt",
    "view": {
        "title" : "Default",
        "json" : "views/default.json"
    }
}
```

Hint: If *https* is used the protocol of the MQTT uri (`mqttws_uri`) needs to be changed to `wss://`!