NetGraphz2

Monitoring system interface

Contents

Introduction

NetGraphz2 — web-based interface for Icinga 2 monitoring system to provide visual graph-like overview of network, replacing static status map. It allow users interact with graph, see how nodes connected between. Such representation allows to find possible outages quickly, analyze weak points of network. Placing nodes in comfortable position using layout algorithm allows user to see clusters of nodes, which may represent connection points in real life.

Application goal is not to represent nodes on real geographical map. We believe that people remember connections much quicker than real world map. Evolution made a big gift for us humans giving us abstract associative memory abilities.

Program uses information from existing monitoring system (Icinga 2) using it's opened interface (MK LiveStatus). It stores graph separately from monitoring system configuration. It's not hardly connected to specific monitoring system.

Nodes and links are stored in graph database (neo4j) which allows application to do such actions as graph traversing really quick. It allows to easily automatically generate monitoring system configuration snippets starting from specific nodes using DFS.

Main web-interface is built on JavaScript. Rendering of graph performed by Cytoscape.js library which is leading library for graph draw and interaction. Additionally we use many other libraries under open license. Full list of used JavaScript libraries available at the document. At the same time structure of application own JS in pretty complex, but we don't use any sort of Bower, Require.js, AMD. We don't think that it's too much clever decision.

Structure of application is partial. There is no central daemon which handles all application life cycle. NetGraphz2 based on several components connected between using internal API. Messaging are passing between configured systems.

The main goal of our project is to provide people good, reliable, fast tool to find out problems in theirs difficult network. Visual interface is the primary priority of project. Without ability to see network from bird-eye view you can't make clever decision about future upgrade and development plan. When you have lack of maps, you don't know anything that might happen. We are here to solve this problem using well-tested, reliable technologies and solutions.

We want to TRY make your network management easy as possible!

Structure

Application solution consists of several applications liked by API. As it was said above, there is no central daemon which controls application life cycle. So, we don't use such way.

At the opposite we have several parts and utilities with own role:

- Web-site (PHP 5.6, Phalcon) — Used to deliver pages, summary information, graph with correct nodes location to end users. We use it as back-end for end users. Site is written following MVC pattern, repository pattern and many others. We believe that good and documented code is what we need there. (folder: `web-phalcon`)
    - Client JS — core of visualization. Used to show graph, navigate through, search through and many more. It receives nodes from web-site, builds graph for use. It uses Cytoscape.js to perform node rendering and interaction. (folder: `web-phalcon/public/js`)
- Notifications server (Node.js) – Receives notification from monitoring system (Icinga) using it's RESTful API and monitoring notifications scripts. Gets node identifier, information and forwards notifications to users though WebSocket or AJAX LongPoll, also forwards chat messages from users. (folder: `notifications`)
- Graph API (Node.js) – Small RESTful API to manage graph nodes and links (folder: `api`)
- Icinga2 configuration generator (python) – Utility to generate configuration, traversing graph from root node using DFS. (folder: `icinga_config_generator`)
- Icinga2 notifications scripts (Node.js) – scripts to forward notifications from Icinga 2 to the notifications server
- Icinga2 include configurations example – sample configurations files to set up interactions between notifications server and monitoring system.

Application

Web front-end

Web interface front-end located in **web-phalcon** folder. It consists of several parts and uses MVC pattern. We use Phalcon 2.0.3 framework to build page, so it should be installed in order to install NetGraphz2.

In the root folder of web-front end you may find such folders:

- app — contains Phalcon applications source files (Controllers, View, Models) and code to access MKLiveStatus service.
- public — contains files to be sent to the user browser including JavaScript user interface

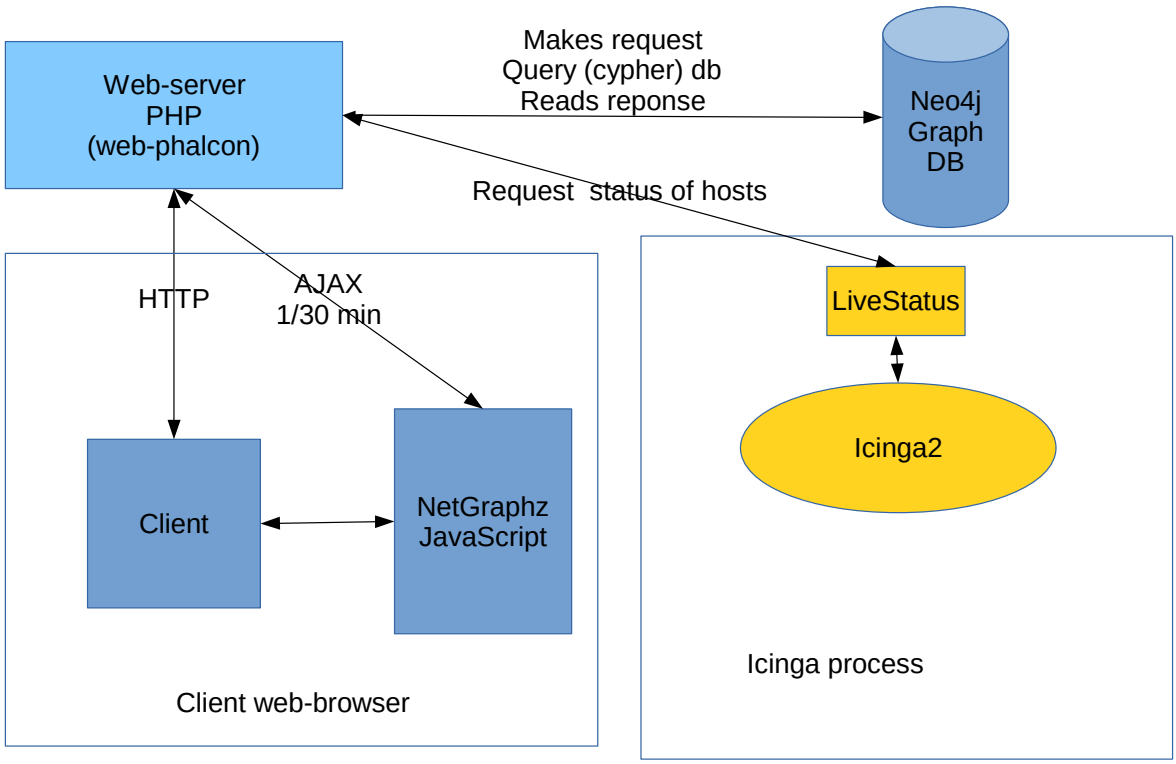

Fig 1: Front-end site role

Let's take a look on front-end role. It is just simple, it gives user information about host from graph database and connects existing nodes to monitoring system data. Front-end application works as visualization: client JS part role — visual interaction, server part — data combining.

Front-end user configuration like node positions, user interface preferences, etc is stored in the MongoDB database. It gives the way to get and store such data in the fastest way.

Client JavaScript

One of the most important part is client JavaScript user interface. Uses Cytoscape.js, jQuery, jQuery UI, socket.io-client and few other 3-rd party libraries. All netgraphz JavaScript code located in **web-phalcon/public/js/netgraphz** folder It split into modules:

- communication.js – Server notifications handler
- core.js – Core definitions and constants
- eventbus.js – Event bus interface to implement publisher-subscriber pattern
- fetcher.js – Graph fetching functions
- graph_utils.js – Useful functions to work with graph states
- main.js – Provides application start-up sequence
- renderer.js – Interface for cytoscape.js renderer
- settings.js – **Contains settings for client scripts**
- store.js – In-memory storage for nodes, links and states
- tools.js – Basic tools like JS-object extend, deep extend, etc
- ui.js – User-interface functions and event-listeners
- ui.notifications.js – Notifications UI
- ui.panel.js – Node/link information panel UI functionality
- ui.search.js – Node search functionality (search field)
- ui.tabstop.js – TAB key problem navigation functionality
- updater.js – Automatic update of graph

Client JavaScript provides verbose error, information console output for browser. You can find out what's going wrong into browser developer console.

Fig 2: JavaScript console output

Notifications server

Forwards notifications between connected users, located in **notifications** folder. Works on Node.js IO framework which is designed for such use cases. Uses netgraphz2 node.js database library located in **node_netgraphz2** folder

Handles web-socket or long-poll connections on port 3433 by default by sending JSON objects to connected clients about current events. Also listens 3434 port by default for Icinga script calls.

Configuration of the server located in **config.json** file.

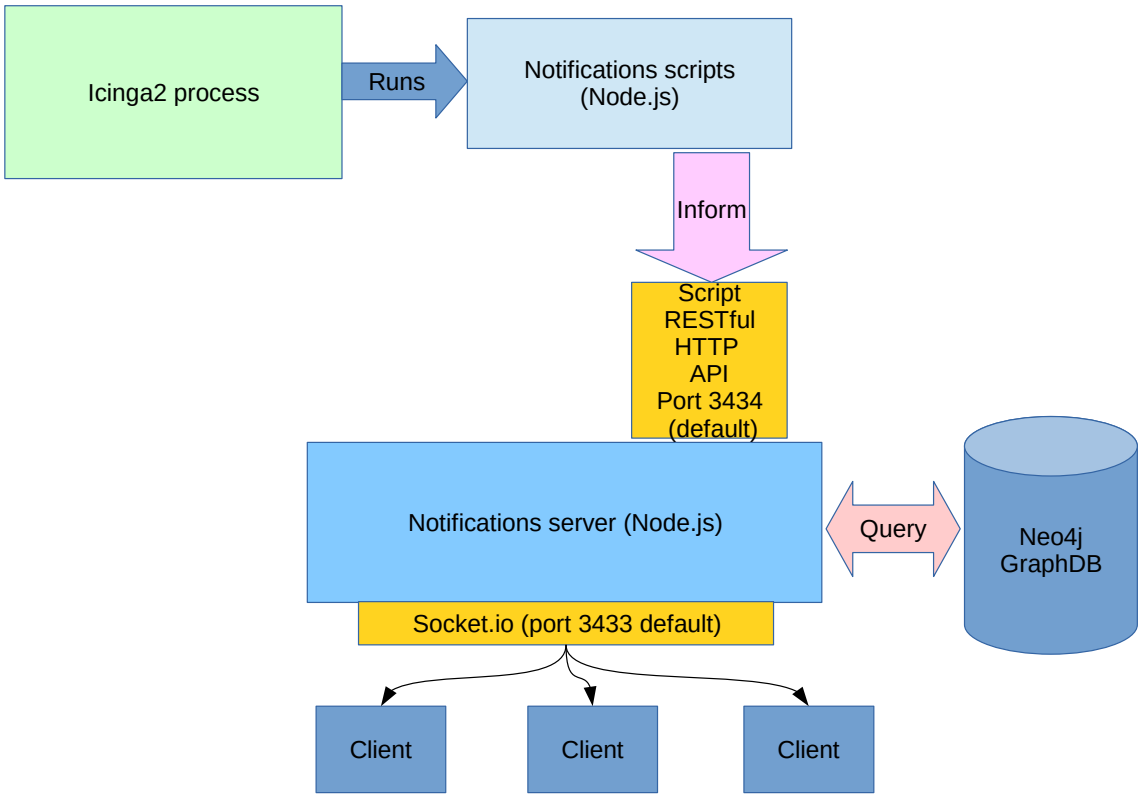

Fig 3: Notifications server flowchart

Graph API

Still in development, will be published in **api** folder. Will use same JavaScript IO framework as run-time and same library as notifications server. Needed to update graph using RESTful API requests. Documentation about this application part will be published later…

Tools

Application also contains few additional tools for deployment purposes.

Icinga2 host configuration stubs generator

Can be used to create Icinga2 host configurations automatically starting from some root host using DFS. Located in **icinga/config_generator** directory, requires few additional Python modules to get started (look into installation section).

Configuration generation is long-time process: it might take 5 minutes to fully finish utility execution (depends from graph size).

Tool uses two files: host_template.cfg, dependency_template.tpl – host and host dependencies templates in order to generate configurations.

Tool configuration sample file can be found config.json file. Prints following help message when started with --help argument:

```
usage: generator [-h] [-v] config_file.json output_folder start_host


Icinga config generator from nodes graph database


positional arguments:
  config_file.json  Configuration JSON file path (see README and manual)
  output_folder     Output folder for config files
  start_host        Starting host for DFS graph scanning


optional arguments:
  -h, --help        show this help message and exit
  -v, --version     show program's version number and exit
```

It generates only host stubs using ip_address and icinga_name graph nodes properties! So, be careful and always backup old Icinga2 host configuration!

Icinga2 scripts and configurations

Located in **icinga** folder. Needed to connect Icinga2 hosts and services notifications mechanism to NetGraphz2 system. Configuration of scripts located in icinga/scripts/config.json file.

Additional required Icinga2 configurations provided in **icinga/conf.d** folder.

Script and configuration installation shown in installation section.

Installation

System installation is easy to do, but requires few manual steps

Unpacking

1. Unpack distribution package into /opt/netgraphz2 directory on your server.
2. Consider changing PATH variable value to make easier to manage NetGraphz2Graph configuration

Preparation of prerequisites

1. Install Icinga 2 (https://www.icinga.org/download/), neo4j (http://neo4j.com/download/), node.js >= 0.10.29 (https://nodejs.org/download/), MongoDB (https://www.mongodb.org/) Python 2.7.5 with Python development files (https://www.python.org/downloads/), Icinga2 LiveStatus plugin service (included in disribution)

   For Debian Jessie or later I recommend to install it from packages:

   ```
   # wget -O - https://debian.neo4j.org/neotechnology.gpg.key |  apt-key add -
   # echo 'deb http://debian.neo4j.org/repo stable/' >/etc/apt/sources.list.d/neo4j.list
   # echo 'deb http://http.debian.net/debian jessie-backports main'  > /etc/apt/sources.list.d/backports.list
   # wget -O - http://packages.icinga.org/icinga.key | apt-key add -
   # echo 'deb http://packages.icinga.org/debian icinga-jessie main' > /etc/apt/sources.list.d/icinga.list
   # echo 'deb-src http://packages.icinga.org/debian icinga-jessie main' >> /etc/apt/sources.list.d/icinga.list
   # apt-get update
   # apt-get install neo4j icinga2 mongodb python python-dev python-pip nodejs
   ```

2. Install Python additional modules (needed to start toolkit). Run following command:

   ```
   # pip install MySQL-python py2neo
   ```

3. Install Apache2 HTTPd >= 2.6 web-server (http://httpd.apache.org/download.cgi).

   Debian Jessie +:

   ```
   # apt-get install apache2
   ```

4. Install PHP 5.6 + FPM enabling JSON, Imagick, GD, memcached, rrd, Mcrypt extensions:

   Debian Jessie +:

   ```
   # apt-get install php5-dev php5 php5-fpm php5-mongo php5-json php5-mysql php5-imagick php5-intl php5-gd php5-memcache php5-memcached  php5-rrd php-pear php5-mcrypt
   ```

5. Enable Apache2 FastCGI proxy extension, enable path rewriting in web-server configuration (e.g mod_rewrite, mod_proxy_fcgi in Apache2).

   ```
   # a2enmod rewrite
   # a2enmod proxy_fcgi
   ```

6. Install PHP Phalcon module (https://phalconphp.com/en/download), follow instruction on Phalcon developers site.

   After buidling and installation of phalcon enable it as PHP module. For Debian Jessie+:

   # echo 'extension=phalcon.so' > /etc/php5/mods-available/phalcon.ini

   # php5enmod phalcon

7. Install some PHP packages from PECL:

   # pecl install SPL_Types

   Enable SPL_Types module in PHP configuration (add extension=spl_types.so into php configuration)

   Debian Jessie +:

   # echo 'extension=spl_types.so' > /etc/php5/mods-available/spl_types.ini

   # php5enmod spl_types

8. Configure FPM:

   Set listening TCP port to 9000 for pool (/etc/php5/fpm/pool.d/www.conf in Debian).

   listen = 127.0.0.1:9000

   and comment or delete line

   listen = /var/run/php5-fpm.sock

   Change automatic restart parameters (PHP sometimes falls down with SEGV) for daemon (/etc/php5/fpm/php-fpm.conf in Debian):

   emergency_restart_threshold = 2

   emergency_restart_interval = 5s

9. Configure Apache2 site for NetGraphz2. Copy basic site definition (**001-netgraphz2.conf**) from **apache2** subfolder of the installation directory to your apache2 **site-available** directory (/etc/apache2/sites-available).

   Edit copied site definition file in the apache2 directory

   Change ServerName, DocumentRoot, ServerAdmin, Directory section path and ProxyPassMatch to your actual values.

   Enable NetGraphz2 site:

   # a2ensite 001-netgraphz2

10. Restart PHP FPM and web-server itself

    Debian Jessie +:

    #service apache2 restart

    #service php5-fpm restart

Neo4j web shell

Once you have installed neo4j, you can enter neo4j web interface by following http://[your server address]:7474/ URL. Here you may change default access credentials and execute some Cypher queries.

We recommend to use neo4j shell which is more powerful and less fancy. You may find instructions how to enable it below.

Neo4j shell configuration

Before starting manipulating graph database we recommend to enable neo4j shell on localhost. In order to make shell works in Debian Jessie edit **/etc/neo4j/neo4j.properties** neo4j configuration file and uncomment following sections:

```
remote_shell_enabled=true

remote_shell_host=127.0.0.1

remote_shell_port=1337
```

Then, you should save modifications to file and restart neo4j service. In Debian you can use following command:

```
# service neo4j restart
```

Now you should be able to use neo4j-shell.

```
# neo4j-shell
```

Shell should start and print output like:

```
Welcome to the Neo4j Shell! Enter 'help' for a list of commands

NOTE: Remote Neo4j graph database service 'shell' at port 1337


neo4j-sh (?)$
```

Now you can quit from the shell: type **quit** and press Enter.

Neo4j adding nodes and links

To add nodes and links, you should use Cypher queries (API still in development, sorry). Information about Cypher query language is available from neo4j manual:

http://neo4j.com/docs/stable/cypher-introduction.html

http://neo4j.com/docs/stable/cypher-getting-started.html

http://neo4j.com/docs/stable/cypher-query-lang.html

Information about all visible attributes is available in “Neo4j Node and Relationship properties”.

To execute query, run neo4j-shell utility. Example queries to make node and links.

```
$ cypher
CREATE (n:NetAssistNode {name: 'naic.29632.as', ip_address: '205.162.49.41', icinga_name: 'netgraphz2 server'}) RETURN n;
$ cypher
MATCH (n:NetAssistNode), (m:NetAssistNode) WHERE n.name='sw-core-1g' AND m.name='netgraphz2 server' CREATE (n)-[r:LINKS_TO]->(m) RETURN r;
$ cypher
MATCH (n:NetAssistNode), (m:NetAssistNode) WHERE n.name='sw-core-1g' AND m.name='netgraphz2 server' CREATE (m)-[r:LINKS_TO]->(n) RETURN r;
```

This set of queries would create node with single attribute name (value 'netgraphz2 server') and link it to existing node with name 'sw-core-1g' (if such node exists).

MongoDB

To get NetGraphz2 user accounts working, you should install MongoDB database dump into your MongoDB server. To install working database dump run **mongorestore** command from **mongo** directory relative to the installation location. It will import installation database to database name 'netgraphz2'.

```
# cd mongo
# mongorestore
```

Access to the MongoDB is available though the **mongo** CLI application.

Linking Icinga2

1. After installation of Icinga2 monitoring system, enable Icinga2 MKLiveStatus extension (https://github.com/Icinga/icinga2/blob/master/doc/15-livestatus.md). Place LiveStatus configuration file into /etc/icinga2/conf.d/livestatus.conf with following contents:

```
library "livestatus"
object LivestatusListener "livestatus-tcp" {
  socket_type = "tcp"
  bind_host = "127.0.0.1"
  bind_port = "6558"
}
object LivestatusListener "livestatus-unix" {
  socket_type = "unix"
  socket_path = "/var/run/icinga2/cmd/livestatus"
}
```

The following configuration will assign listening address to 127.0.0.1 and listening port to 6558 (TCP) and unix socket to `/var/run/icinga2/cmd/livestatus`. It's possible to choose other ports if you configure web-application correctly.

2. Configure Icinga2 scripts (/opt/netgraphz2/icinga_scripts/config.json). Default port of

notifier API is set to be 3434 TCP.

3. Link Icinga2 configuration directories:

   # cd <Icinga2 configuration path> - mostly /etc/icinga2

   # ln -s /opt/netgraphz2/icinga/scripts/ scripts.netgraphz2

   # ln -s /opt/netgraphz2/icinga/conf.d/ netgraphz2.d

   Edit icinga2.conf:

   Add line:

   ```
   include_recursive "netgraphz2.d"
   ```

4. OPTIONAL: Generate Icinga2 configuration stubs from existing graph database using utility located in /netassist/netgraphz2/icinga/config_generator directory (look in “Icinga2 host configuration stubs generator” subsection)
5. Add your hosts configuration into Icinga2 and restart Icinga2

Preconfiguration

Read configuration section of this manual to understand configuration variables meaning.

6. Configure web interface (/opt/netgraphz2/web-phalcon/app/config/config.php).
7. Configure notifications server (/opt/netgraphz2/notifications/config.json).
8. Configure client JavaScript (/opt/netgraphz2/web-phalcon/public/js/netgraphz/settings.js). Don't forget to change notification server URL!

Notifications server

1. If you using **systemd** startup system, you can use provided service definition located in **systemd/netgraphz2.notifications.service** file relative to your installation directory. First of all, edit it and change default path of /opt/netgraphz2 if you have another installation directory. Then, copy a service definition file to your **/lib/systemd** directory and execute following command:

   # systemctl enable netgraphz2.notifications

   Service will be installed in your system and enabled in autostart.
2. Start notifier daemon #[nodejs or node] /opt/netgraphz2/app.js. If you use systemd and installed service following to the previous step, just execute:

   # systemctl start netgraphz2.notifications

Final steps

3. Open web-browser and enter NetGraphz2 web-site address, register you personal account.
4. Sign into your account, wait simulation to finish and save preferred nodes positions
5. Enjoy NetGraphz2

Configuration

Web front-end

Web front-end Phalcon-based application is configured by main configuration file located in /web-phalcon/app/config/config.php relative to your installation path. This Phalcon application configuration file, it uses PHP array syntax to specify configuration variable values. Most of server-side parameters like neo4j database connection, MongoDB connection, company information, Icinga2 LiveStatus connection address and port. Configuration variables names are case sensitive!

Configuration syntax look like:

```
<?php
...
return new \Phalcon\Config(array(
    'information' => array(
        'companyName'   => 'My company',
        'siteName'      => 'NetGraphz2',
        'icingaUrl'     => '/icinga2-classicui',
        'openSignUp' => true
     ),
...
?>
```

It consist of several section represented as PHP arrays. Syntax should be valid PHP,

Every option described below should be specified. Don't change some default values of **'application'** configuration section unless you don't imagine how it may affect application stablity and performance

Information section

Section called **'information'**. Here you can change company information and global page title. These parameters are not quit important for application functionality. The main possible usage is distringuishing different NetGraphz2 installations.

Variable name	Description	Type	Example (default)
companyName	Specifies company name shown on all pages of the web-application	string	'NetAssist'
siteName	Name of site appended to the page title	string	'Our NetGraphz'
icingaUrl	URL of Icinga2 web interface (link shown in the top menu)	string	'http://example.com/icingaweb2'
openSignUp	Allow to signup by follow sign up link (true/false).	boolean	true

MongoDB section

Section called **'mongo'**. Enter your MongoDB connection configuration here. This section is very short comparing to the rest of configuration.

Variable name	Description	Type	Example (default)
connectionString	PHP MongoDB driver connection string: host, port and additional network options. For more information and options follow MongoDB reference manual section here: https://docs.mongodb.org/manual/reference/connection-string/ . Don't forget to specify connectTimeoutMS option to avoid page hungs if MongoDB goes down.	string	`'mongodb://localhost:27017/?connectTimeoutMS=1200'`
database	Database to use	string	'netgraphz2'
options	Additional MongoDB driver options. Reserved for future usage. Leave as empty PHP array.	php array	array ()

Neo4j section

Section is called **'graph'**. It's designed to enter your neo4j server paremeters.

Variable name	Description	Type	Example (default)
host	Neo4j database server host name	string	`'localhost'`
port	Neo4j HTTP RESTful API port	integer	7474
auth	Set's flag is HTTP Authentication required for Neo4j API	boolean	true
username	Neo4j HTTP Authentication username	string	'neo4j'
password	Neo4j HTTP Authentication password	string	'changed'
scheme	URL scheme to be used to access Neo4j API. Possible values are 'http' and 'https' for SSL encrypted connection (recommended for remote hosts)	string	'http'

MKLiveStatus section

This section specifies MKLiveStatus parameters to communicate with Icinga2 proccess through the PHP stream socket API. It's called simply '**livestatus'**. If this section is misconfigured, you'll get always grey nodes. Check out web-server log and PHP stderr output to find out connection problems (see «Troubleshooting» section). There is two possible ways to connect: unix sockets and TCP/IP socket.

Variable name	Description	Type	Example (default)
host	MKLiveStatus hostname	string	`'localhost'`
port	MKLiveStatus TCP port (in case of TCP connection)	integer	6558
keepAlive	Try to keep socket connection alive for future reusage and faster request processing like connection pool	boolean	false
authEnable	Sets if MKLiveStatus request 'auth' parameter used sent during each request. It's not implemented fine in any of MKLiveStatus implementation yet, so it's not used in practice yet.	boolean	false
authUser	MKLiveStatus authentication user parameter (optional). Leave empty if you not using auth parameters	string	''
unixSocket	Use UNIX domain sockets to connect to the MKLiveStatus plugin. Using TCP/IP connection by default.	boolean	false
unixSocketPath	UNIX domain socket file path. Leave empty if you not using UNIX sockets (connect using TCP/IP connection).	string	'/var/run/icinga2/livestatus.sock'

Application section

Phalcon **'application'** section provides mostly framework speicific configuration options. Some of them are quit important and should be changed to make application work properly.

Variable name	Description	Type	Example (default)

controllersDir
modelsDir
migrationsDir
viewsDir
pluginsDir
libraryDir
cacheDir
graphDir
...
	Phalcon application component directories. Don't changes these settings unless you are not an application developer.	string	`APP_PATH . '/app/controllers/'`
			`APP_PATH . '/app/models/'`
			`APP_PATH . '/app/migrations/'`
			`APP_PATH . '/app/views/'`
			`APP_PATH . '/app/plugins/'`
			`APP_PATH . '/app/library/'`
			`APP_PATH . '/app/cache/'`
			`APP_PATH . '/app/graph/'`

baseUri	Base relative URL of web application. Default value is '/' consider application located in the root directory of web-site.	string	'/'
cryptSalt	Cryptographic salt for encryption algorithm. It has to be string of random characters 24 chars long. **Def. value should be changed!**	string	'Thae4pijiexahfahYief3411'

rememberLifeTime	Cookie life time for 'remember me' login option in seconds	integer	604800
failLoginWindowTime	Window time (t) in seconds for N unsuccessful login tries sequence before user account get blocked by application	integer	300
failLoginBlockPermament	Sets if user gets permanently blocked after N unsuccessful login tries sequence in window time (t)	boolean	false
failLoginBlockTime	Sets temporary block time in seconds if user exceeds N unsuccesful login ties in window time (t). Works if failLoginBlockPermament set to be **false.**	integer	600
failLoginWindowMaxCount	Specifies N - number of unsuccessful login before user get blocked by application	integer	5

JavaScript user interface

JS user interface configuration located in /web-phalcon/public/js/netrgraphz/settings.js file relative to your installation path. It's JavaScript file, so configuration is done by modifying object values in JSON-way.

File should stay same valid JavaScript after modifications, otherwise NetGraphz will not work properly. If you break syntax, restore original file from NetGraphz2 distribution package.

File syntax look like:

```
var netgraphz =  netgraphz || {};
netgraphz.settings = (function(){
  return {
    'communication': {
      'remote_url': 'http://netgraphz.naic.29632.as:3433', //URL of notifications server
    },
    'updater': { //Updater settings
      'updateInterval': 30000, //Interval to start fetching new nodes data (30 sec by default)
      'partInterval': 1200, //Interval between parts
      'partSize': 30 //Ammount of nodes to fetch by request, should be not too big and not too small
    },
        ...
 };
})();
```

Communication section

Settings for notification server connection. It used to have reverse communication WebSockets channel for notifications and status updates. Section name is **'communication'**.

Variable name	Description	Type	Example (default)
remoteUrl	URL of NetGraphz2 Notifications WebSocket service	string	`'http://myhost:3433'`

Updater section

Automatic background nodes status updater. It runs each Tupd, downloads new some part of node status information, waits Tpart time then downloads next part until it finish update.

Variable name	Description	Type	Example (default)
updateInterval	Interval between update batches (Tupd) in milliseconds. Update runs every Tupd time, if previous update iteaction took longer than Tupd, waiting time will be substracted from this interval.	integer	30000
partInterval	Interval between parts (Tpart) in milliseconds	integer	1200
partSize	Number of hosts in each update part	integer	30

Renderer section

Graph renderer engine (Cytoscape.js) and Layout algrorithm parameters

Variable name	Description	Type	Example (default)
container_id	DOM Element ID of container to place graph	string	'mynet'
layout_time	Layout (graph simulation) time in milliseconds	integer	5500
initialRadius	Radius of initial positioning circuit to place nodes in px	integer	400
doubleTapTime	Time to detect of double tap (double click) in milliseconds	integer	400
animationTime	Nodes transition animation time in milliseconds	integer	1000
zoomNodeLevel	Level to zoom in/out when node was double clicked	float	1.75
autoResizeContainer	Automatically resize container dimensions when window resizes	boolean	True

state_palette	Sets default color palette for node states
	'-1' — unknown state (unmonitored/not loaded)
	0 — down state
	1 — up state
	2 — warning state
		JS object	{
			'-1': "#8C8B76",
			0: "#FC766D",
			1: "#86D95D",
			2: "#F0DE78"
			}

Layout

Layout algorithm paramaters

Variable name	Description	Type	Example (default)
name	Layout algorith to be used. Currently we support only ported version ForceAtlas2	string	'forceAtlas2'
animate	Animate layout changes	boolean	true
refresh	Number of page refreshes between rendering	integer	1
ungrabifyWhileSimulating	Disable node grabbing ablities when simulation is running	boolean	true
fit	Try to fit nodes into view port (container)	boolean	true
zoomNodeLevel	Level to zoom in/out when node was double clicked	float	1.75
padding	Padding around simulational field in px	integer	30
boundingBox	Constrain layout bounds { x1, y1, x2, y2 } or { x1, y1, w, h } or null (undefined)	JS object	undefined
useWebWorker	Use WebWorker to run simulation. You should use it, otherwise application will tun into laggy mess.	boolean	true
linLogMode	LinLog mode of layout algorithm (forceAtlas2)	boolean	false
outboundAttractionDistribution	Enable outbound attraction distribution for algorithm. It makes final result a little wider.	boolean	false
adjustSizes	Adjust simulation sizes to extend space between nodes	boolean	true
spreadAfterStop	Spread out nodes after layout finishes (to prevent jams)	boolean	true
edgeWeightInfluence	Coefficient of edge influence	float	0
scalingRatio	Scaling ratio of simulation. Important parameter: it set's up physical dimensions scaling and affects on how far node clusters would be after simulation	float	3.0
strongGravityMode	Turns on strong gravity (R^2) mode	boolean	false
gravity	Gravity coefficient	float	0.95
slowDown	Slow down coefficient for equations of motion	float	0.2
infinite	Infinite simulation, don't count on potential energies values	boolean	true

UI section

User interface configuration

Node panel

Node information panel UI configuration. Section 'node_panel'

Variable name	Description	Type	Example (default)
node_panel_id	Node panel DOM element identifier	string	'node_panel'
node_panel_close_button_id	Node panel close button DOM element identifier	string	'node_panel_close'
fadeTime	Time to fade out panel in milliseconds	integer	400
holdTime	Time to hold panel in milliseconds	integer	2400
links	Provides configuration for links in node panel (below) for each node. Look below for link configuration	JS array	[{ 'title': 'Icinga', 'url': '/cgi-bin/icinga2-classicui/extinfo.cgi?type=1&host={icinga_name}', 'type': 'link', 'newTab': true }, …]

Links in node panel

Format of link object

Variable name	Description	Type	Example (default)
title	Label of link	string	'icinga'
url	URL of link (contains tags to replace). Each tag is covered by brackets { }. Possible tags to replace: {icinga_name} {ip} {name} {model}	string	'/cgi-bin/icinga2-classicui/extinfo.cgi?type=1&host={icinga_name}'
type	Type of link ('link' or 'popup'). Link – acts as generic link Popup – creates popup window when clicking (may be blocked by browser)	string	'link'
newTab	Open link in new tab	boolean	true

popupSize	Size of popup window. JavaScript object. Contains 'width' and 'height' variables - integer size of popup in pixels.	JS object	{
			'width': 300,
			'height': 400
			}

popupName	Name of popup. Contains tags to replace.	string	'icinga - {icinga_name}'

Tab stop

Settings of tab navigation module. Responsible for TAB, SHIFT+TAB navigation. Section called 'tabStop'.

Variable name	Description	Type	Example (default)
enabled	Enable UI.TabStop module	boolean	true

Notifications

Settings of UI notifications. Section called 'notifications'.

Variable name	Description	Type	Example (default)
tryUseDesktop	Try to use HTML5 desktop notification API	boolean	true
showTime	Show time for Desktop notifications	integer	900

sounds	JS object to specify sounds to be played during different raise of different types of notifications.
	
	Notification types:
	info – System information
	warning – Node enters warning state (packet loss, etc)
	error – Node goes or still down
	ok – Node goes up
	
	Key-value object ('key': 'value')
	
	'key' – string – notification type
	'value' – string – relative path of sound file
		JS object	{
			'info': '/sounds/KDE-Sys-App-Message.ogg',
			'warning': '/sounds/KDE-Sys-Warning.ogg',
			'error': '/sounds/KDE-Sys-App-Error.ogg',
			'ok': '/sounds/KDE-Sys-App-Positive.ogg'
			}

toastr	Specify toast notification extension parameters used when Desktop notifications not available.
	
	http://codeseven.github.io/toastr/demo.html to get behavior you want
		JS object	{
			'closeButton': true,
			'debug': false,
			'newestOnTop': false,
			'progressBar': false,
			'positionClass': 'toast-bottom-left',
			'preventDuplicates': false,
			'onclick': null,
			'hideDuration': 1000,
			'timeOut': 5000,
			'extendedTimeOut': 1000,
			'showEasing': 'swing',
			'hideEasing': 'linear',
			'showMethod': 'fadeIn',
			'hideMethod': 'fadeOut'
			}

Search

Section called 'search'. It defines behavior of search field.

Variable name	Description	Type	Example (default)
enabled	Set's if search field is enabled	boolean	true
searchInputId	Search field DOM element identifier	string	'node-name-search'

Notifications server

Notifications server configuration located in **notifications/config.json** file. Here you may change basic configuration of NetGraphz2 notifications server.

Variable name	Description	Type	Example (default)
host	Hostname that identifies server	string	'localhost'

Icinga2 scripts HTTP API (RPC) settings

Section 'api'. Provides settings for HTTP server to handle Icinga2 event scripts.

Variable name	Description	Type	Example (default)
authEnabled	Enable authentication	boolean	true
authTokensPath	Path to the authentication tokens collection file	string	'auth_tokens.json'
listenAll	Listen on all addresses (0.0.0.0)	boolean	false
port	Listen TCP port	integer	3434
address	Listen address	string	'127.0.0.1'
icinga	Icinga2 event forward settings. 'broadcast_user' – Icinga user to resend it's notifications to all connected users	JS object	{ 'broadcast_user': '_netgraphz2_notify_all' }

Neo4j NetGraphz settings

Section called 'netgraphzdb'. Settings of graph database.

Variable name	Description	Type	Example (default)
port	Port of neo4j API	integer	7474
useAuth	Use authentication for server	boolean	true
auth	Authentication data. JSON object with string properties login and password	JS object	{ 'login': 'neo4j', 'password': 'neo4j' }

Server section

Section called 'notificator'. It provides settings for WebSocket connection listener.

Variable name	Description	Type	Example (default)
port	TCP port to listen	integer	3433

Neo4j Node and Relationship properties

Hosts

Use nodes with label **NetAssistNode.** **Following properties supported:**

- name [string] — node name
- db_sw_id [int] — database identifier (if imported)
- address [string] — physical location
- comment [string]
- model [string] — Device hardware model
- **icinga_name** [string] — links to the Icinga2 monitooring host name
- serial [string] — Serial number
- num_ports [int] — Number of ports (for switches, routers)
- ip_address [string] — IP address of Node (currently only IPv4)
- manage_vlan_id [int] — VLAN of management
- mac_address [string] — MAC address
- loc_id [int] — NetAssist internal
- net_id [int] — NetAssist internal

Links

Create relationship with type **LINKS_TO**. **Following properties supported**:

- db_sw_id [int] - database identifier of source node
- db_ref_sw_id [int] - database identifier of destination node
- speed [int] — link speed in mbps
- quality [int] — link quality for 0 to 1 [OPTIONAL]
- type [int] — link type (0-fiber, 1-copper, 2-radio, …) [OPTIONAL]
- port_id [int] — port number on source node
- ref_port_id [int] — port number on destination node
- comment [string]

Usage

Open main page, you'll see loading graph. After loading layout start, ForceAtlas2 simulation will automatically make clusters. You can manually control layout execution pressing «Layout start» and «Layout stop» buttons. If user zoomed to much, it can restore graph position to the center using «Reset zoom» button.

Graph navigation is performed by mouse. Zooming is done by scroll wheel. Each node can be dragged in to preferred position. Single click selects node, double click focuses node in the center of the view-port.

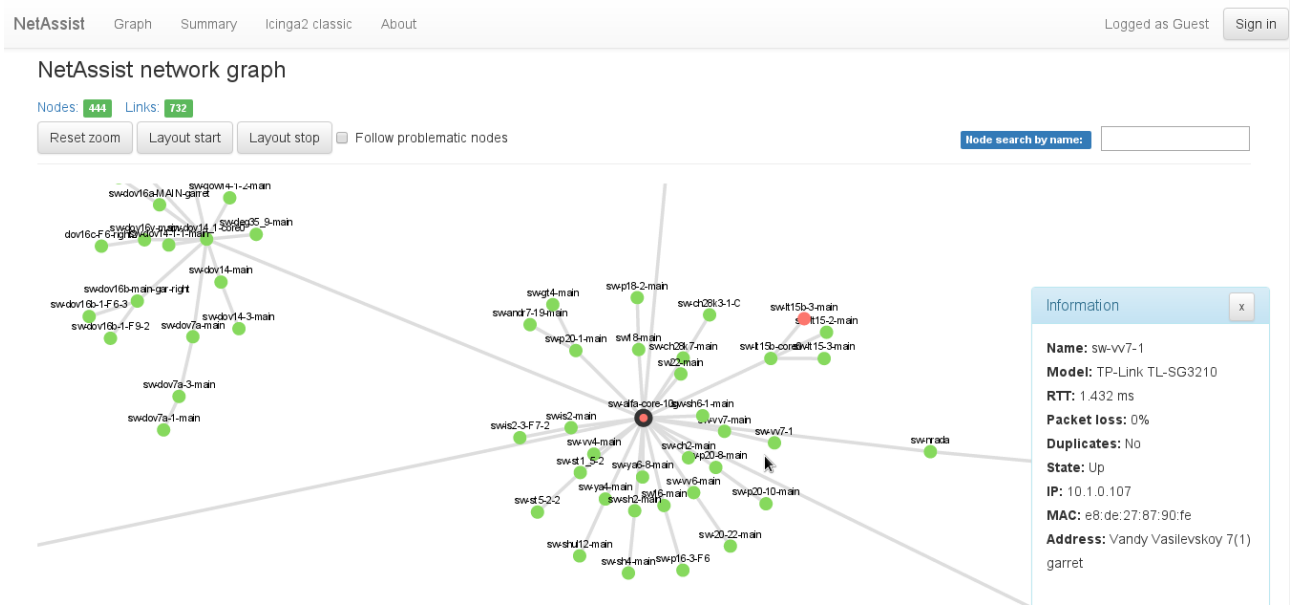

Fig 4: NetGraphz2 user interface

Nodes status are shown in different colors: red — down, green — up, warning — problematic, gray — not monitored by Icinga2.

Node information panel raises automatically when node being hovered by cursor.

Navigation through problematic and down nodes done by [TAB] and [SHIFT]+[TAB] keys. Search through nodes names done by search field (top right, [/] key). Check box on top of the screen enable automatic following to the down nodes in real time.

It's possible to select multiple nodes at once: hold [SHIFT] key and click on nodes you want to choose. Rectangular selection is done by holding [CTRL] key and selecting screen region.

Troubleshooting

There are common errors may happened during application usage. NetGraphz2 relies on several components and theirs communications. If something breaks specific error rise up on the main page.

Neo4j

- No connection to the graph database.

  Symptoms:

  You get error message like specified below, usually, on the blank page. Web server return code HTTP error codes between 500-503. Error description, graph URL, script path depends from your instance configuration.

  Error on Connection "default" with message "Error creating resource: [message] fopen(http://localhost:7474/db/data/transaction/commit): failed to open stream: Connection timed out [file] /opt/netgraphz2/web-phalcon/vendor/guzzlehttp/guzzle/src/Handler/StreamHandler.php [line] 282.

Fig 5: Graph database connection error

Reason:

Web-site (web-phalcon) failed to get graph from graph database due to connection failure (timeout, refuse, etc). If you host graph database on separate host, it's possible caused by network communication error. In most cases the reason is not running or down Neo4j. Neo4j is stable application in most cases, but sometimes it may be killed or crashed because of low RAM available (Java takes a lot of RAM) if no swapping space enabled.

Possible solution:

Check communication to the remote or local neo4j database, check database URL in configuration file, check firewall and network routing topology. Check if neo4j process is running remotely or locally. If neo4j failed to start up or crashed, review error log.

Note:

Error may appear when server CPU load is very high or database is under heavy load. It happens because of HTTP client timeout. Wait for load to come down and try to reload the page.

- Neo4j authentication failure

  Symptom:

  Similar to the graph databases connection failure. Contains different error message like specified below. Usually appears on the blank page the same way as connection error.

  `Error on Connection "default" with message "Client error: 401"`

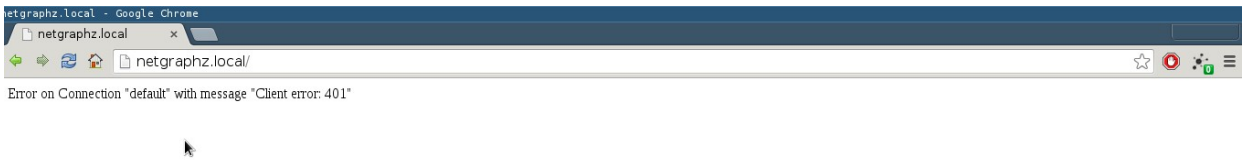

Fig 6: Neo4j authentication failure

Reason:

Web-site (web-phalcon) failed to authenticate on neo4j database instance due to the incorrect login-password pair, specified in application (web-phalcon) configuration file.

Solution:

Change login and password from web-site configuration file (web-phalcon/app/config.php) or change neo4j user name and password. Try to reload page.

MongoDB

Some possible issues with MongoDB

- Cannot connect to MongoDB

Symptoms:

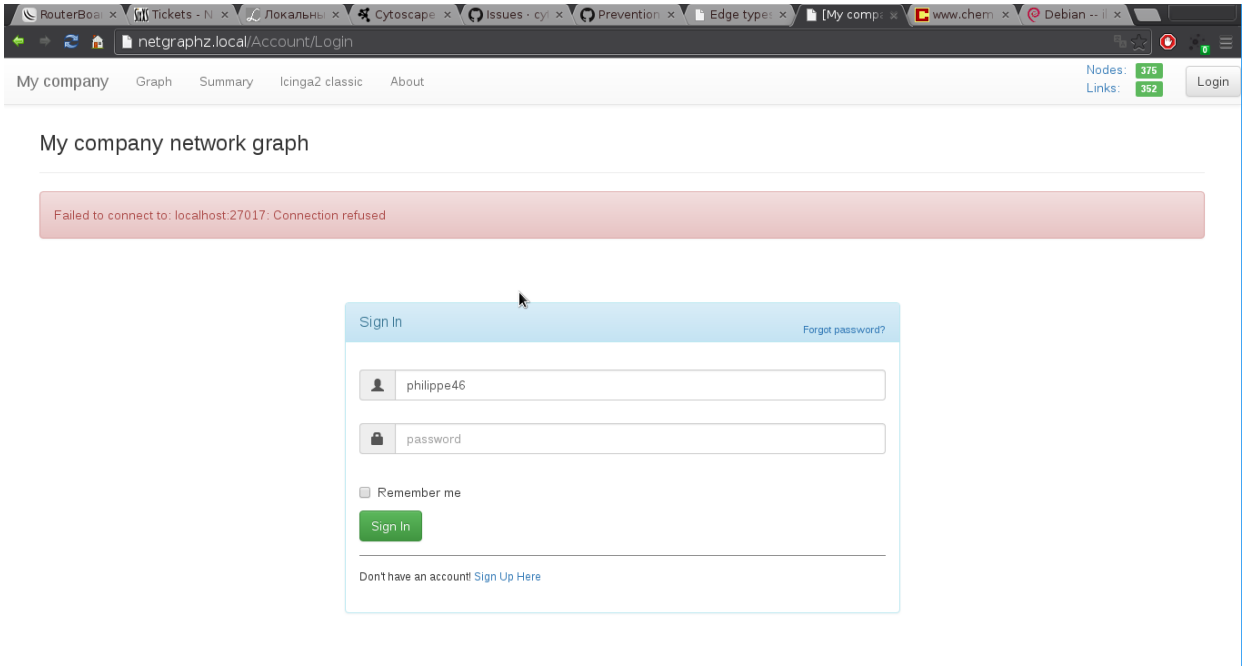

Fig 7: Login failure

Error message during login, impossible to login due to error.

Error message when loading graph page (failed to load user parameters) for already logged user.

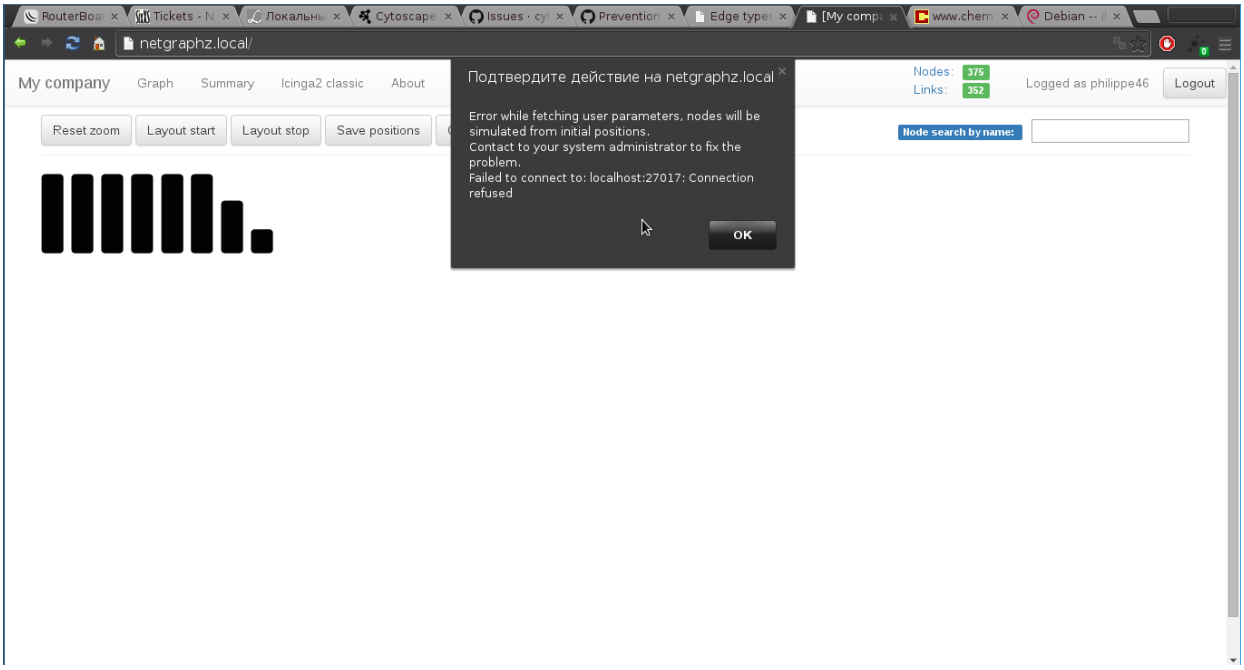

Fig 8: Properties load failure

Error message when clicking on nodes positions save/clear buttons

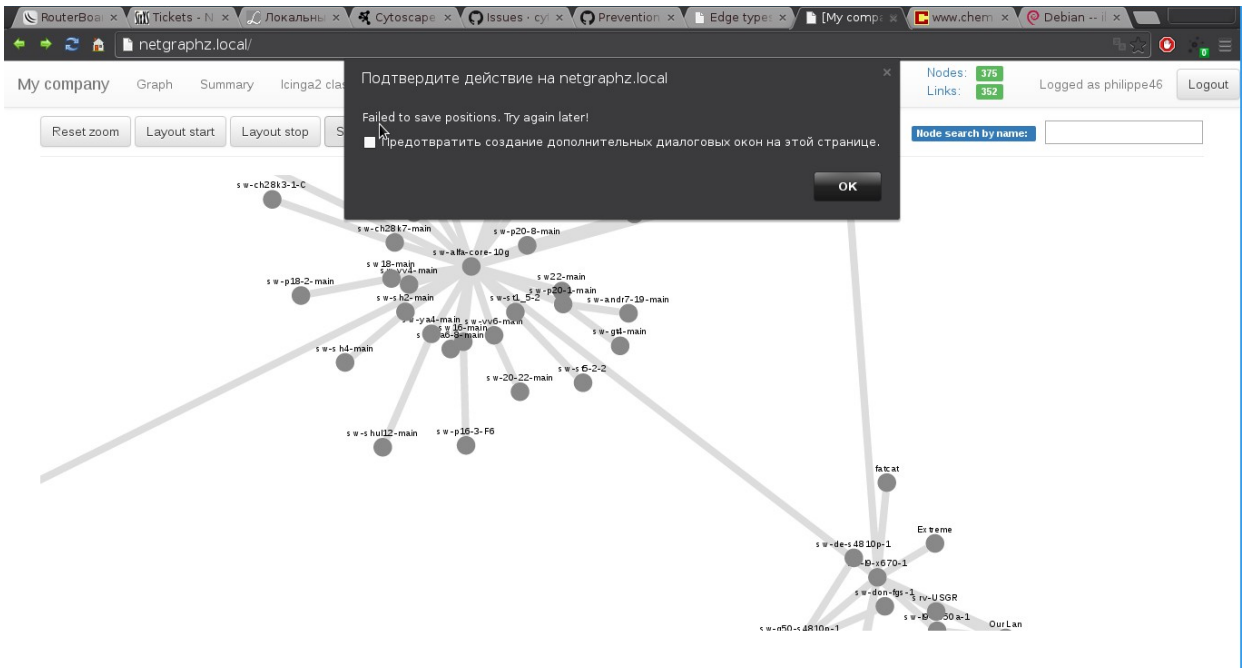

Fig 9: Positions save failure

Reason:

Web-site (web-phalcon) failed to access MongoDB database, error message shows error reason.

Solution:

Check if MongoDB is running, database created correctly. Change connection string in web-application configuration file. Try to reload page. If doesn't help, reinstall MongoDB database from installation dump.

Notifications server

- Failed to reach notifications serve

Symptoms:

Nodes changing it's state but no notifications received for a long time. Notifications server connection error messages from socket.io library present in JS console.

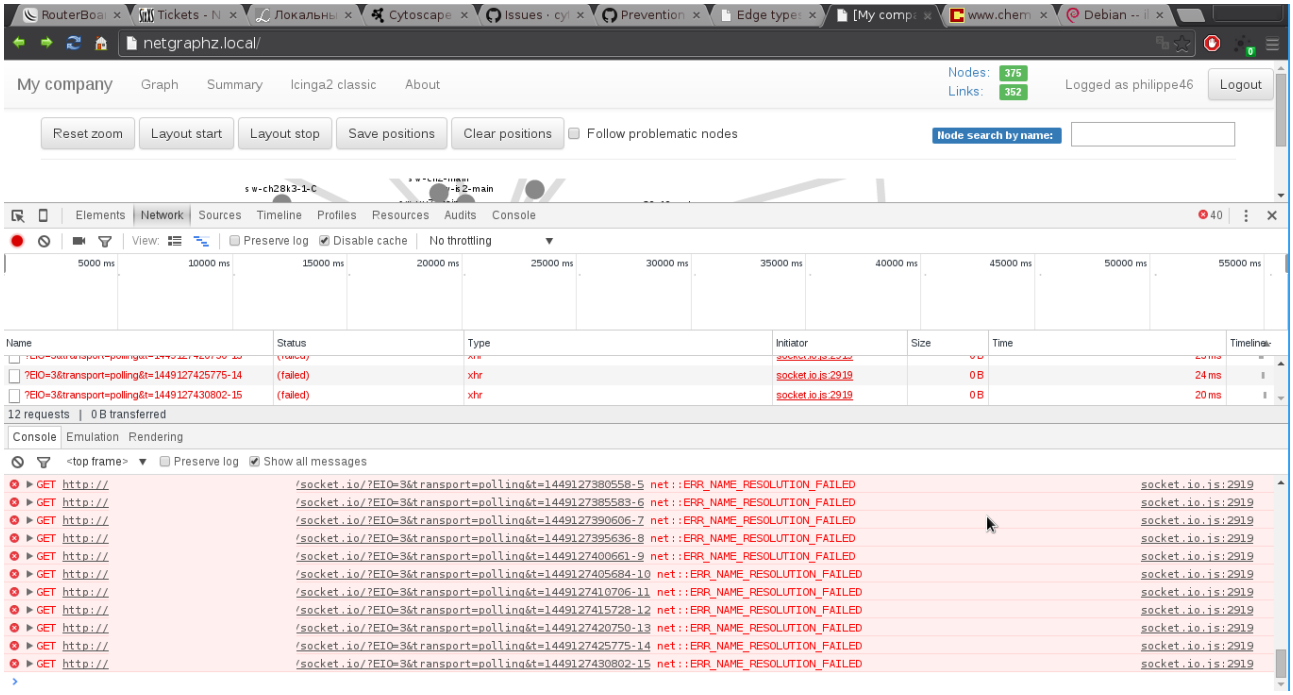

Fig 10: Connection error

Reason:

Browser application failing to connect to notifications server. It might be network error, misconfiguration or problem on notification server side.

Solution:

Check notification server, JavaScript application configuration files. Check network connectivity issues: you must see valid HTTP response on notifications server port. Restart notifications server and try again.

Contacts

Company:

NetAssist LLC

Ukraine, Kyiv

str. Striletskaya 7/9 office 96

http://netassist.ua

Project leader:

Philippe Duke

NetAssist LLC

Email: philippe46@netassist.ua

philippe46@pulsepad.com.ua

philippe46.snproject@gmail.com

philippe46@henkuaile.cn

Site: http://henkuaile.cn

Jabber: philippe46@netassi.st

Facebook: http://facebook.com/philippe46.snproject