GrafanaDashboardCreator Documentation_EN

Inhalt

The Idea of the Program (Functionality)

Since Grafana stores its dashboards, as well as the elements in them, in JSON format and has a REST API to create such JSON objects externally and then enter them into the system via POST, it makes sense to automate this process with the help of a tool.

However, in order not to have to write a complete dashboard creator yourself, the program used a few tricks.

How Grafana stores its dashboards/panels?

As already described, Grafana stores its data in JSON format. For an empty dashboard, this looks like this:

```
Leeres Dashboard als JSON
{
  "annotations": {
    "list": [
      {
        "builtIn": 1,
        "datasource": "-- Grafana --",
        "enable": true,
        "hide": true,
        "iconColor": "rgba(0,0,0,1)",
        "name": "Annotations & Alerts",
        "type": "dashboard"
      }
    ]
  },
  "editable": true,
  "gnetId": null,
  "graphTooltip": 0,
  "id": null,
  "links": [],
  "panels": [],
  "schemaVersion": 27,
  "style": "dark",
  "tags": [],
  "templating": {
    "list": []
  },
  "time": {
    "from": "now-6h",
    "to": "now"
  },
  "timepicker": {},
  "timezone": "",
  "title": "",
  "uid": null,
  "version": 0
}
```

What does the program do now?

The program takes such "empty templates" of Grafana's JSON objects and fills them with the desired information. It is also able to extract templates from existing dashboards in order to be able to use them again. In this way, you do not need a complete "dashboard compiler" and can still easily create dashboards automatically.
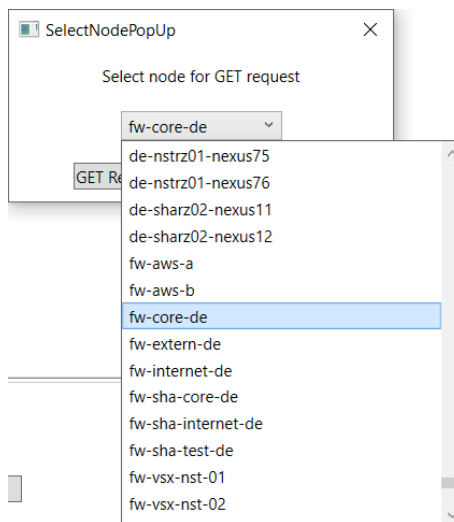
How to use the program?

Reading in data

For the import of the necessary resources and nodes you can use the **GET Resources** button.

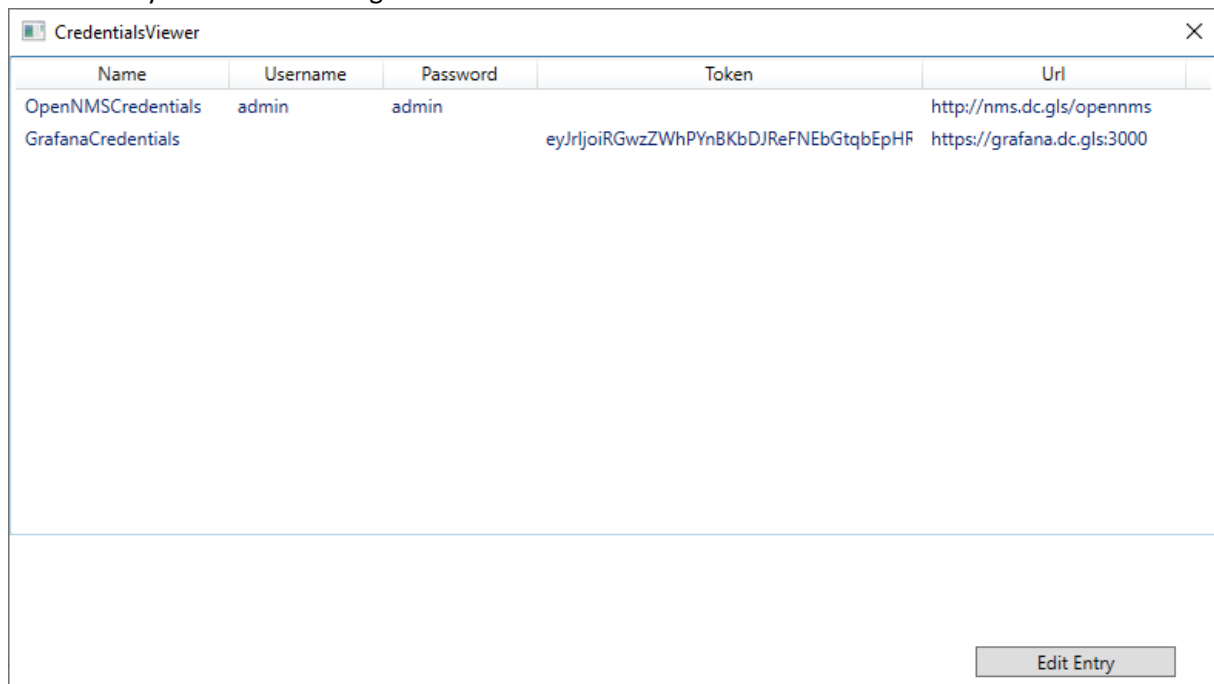First it gets the nodes from the OpenNMS REST-API.

After selecting a node from the drop-down menu, the data sources belonging to this node are then queried and displayed in the program in the "Datasource" tab.
It is possible to load the data sources of several nodes at the same time, these are then first sorted according to their associated nodes and then lexicographically according to their labels.

Managing login data

In order to establish a connection to such APIs, the program needs the corresponding data in the form of a URL, as well as the login data. These are stored in a file in the "Datastore/Credentials" subdirectory and can be managed via the [Open Credentials Viewer] button.
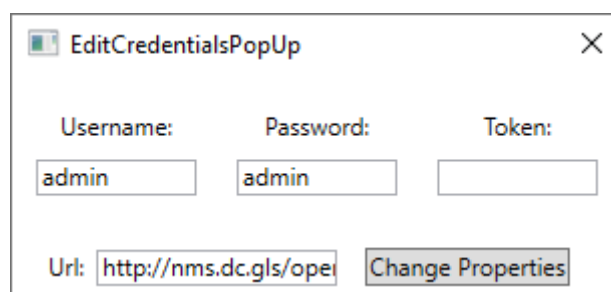
To edit an entry, select the entry and click on the [Edit Entry] button. In the window that opens, the user name is required under "Username", the password under "Password" and the token for authentication on the server under "Token".

Important: OpenNMS expects a username, as well as a password, Grafana needs a token!

Under "Url" the address of the server is expected, important: It is not the API address required! In the case of OpenNMS, the API address would be something of the format "http://opennmsserver/opennms/rest/", only the part before the "/rest/" is needed here, so something of the form "http://opennmsserver:8980/opennms"!
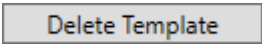
Reading/managing templates

As already described, the program works internally with templates of the objects to be created. This means that it takes a finished object of a type and replaces the necessary information there, such as title, ID, Node/ResourceID. Templates are stored in the program for the objects "Dashboard", "Row" and "Folder". For the panels, however, these must be imported separately. To do this, you create a dashboard in the <u>Grafana interface</u> with the panel as you would like it and export it via the web interface to a JSON file.

In the program you open the [Open Template Viewer] interface for managing the templates via the button.

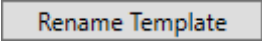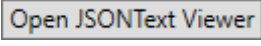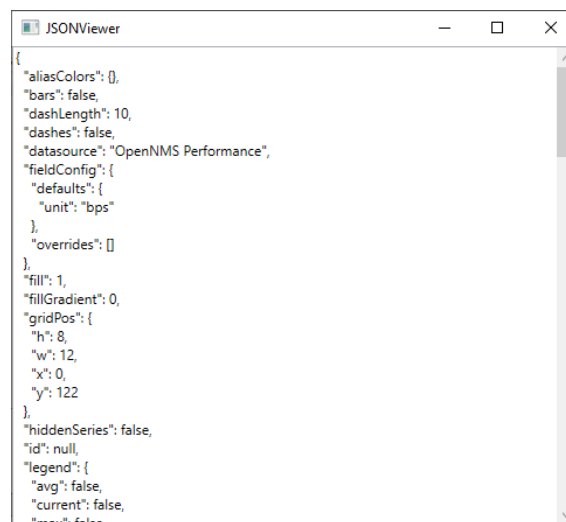| Name	ReplaceNodeID	ReplaceResourceID	JSONText
Bandwidth	True	True	{ "aliasColors": {}, "bars": false, "dashLength": 10, "dashes": false, "datas
CPU	True	False	{ "aliasColors": {}, "bars": false, "dashLength": 10, "dashes": false, "datas
Memory	True	False	{ "aliasColors": {}, "bars": false, "dashLength": 10, "dashes": false, "datas
Response Time	True	True	{ "aliasColors": {}, "bars": false, "dashLength": 10, "dashes": false, "datas

Open JSONText Viewer
Rename Template
Get New Template

Delete Template

There you can delete the selected template with the [Delete Template] button, with the [Rename Template] button the selected template can be renamed and via the [Open JSONText Viewer] button the JSON text of the template can be checked.

JSONViewer

{
 "aliasColors": {},
 "bars": false,
 "dashLength": 10,
 "dashes": false,
 "datasource": "OpenNMS Performance",
 "fieldConfig": {
  "defaults": {
   "unit": "bps"
  },
  "overrides": []
 },
 "fill": 1,
 "fillGradient": 0,
 "gridPos": {
  "h": 8,
  "w": 12,
  "x": 0,
  "y": 122
 },
 "hiddenSeries": false,
 "id": null,
 "legend": {
  "avg": false,
  "current": false,
  "max": false

A new template can be loaded via the [Get New Template] button. Under "Name" a freely selectable name for the internal display is expected, under JSONTitle the title of the template, as it was created in the web interface, is expected and under "PathToJSON" the complete path to the JSON file created above is expected.
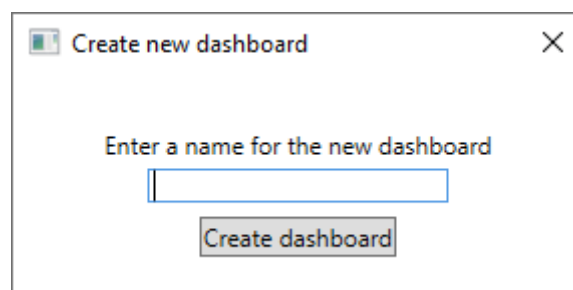
The checkboxes can be used to determine which IDs of the data source must be entered later when creating the export. As a rule, the NodeID should always be selected (here only the option for possible special cases has been left open), the ResourceID does not have to be replaced for templates such as "Memory" and "CPU". So, if the ResourceID would be the same for all panels of all nodes. If you want to load a template of an interface resource instead, it must be adapted later in the template according to the panel to be created for other resources.

Creating a dashboard

The button [Create New Dashboard] can be used to create a new dashboard.
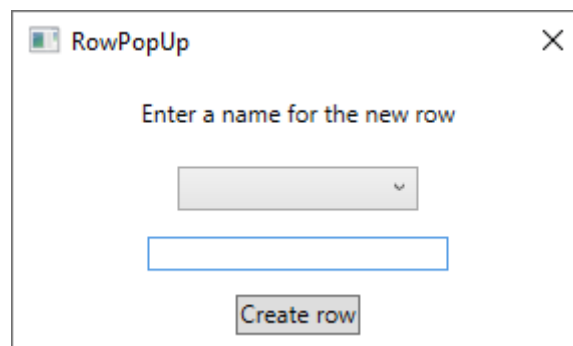
Enter the name of the new dashboard and confirm.

Creating a row

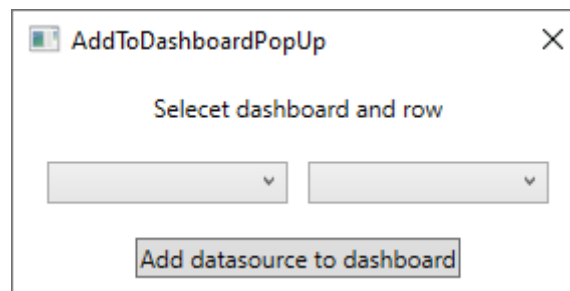A new row can be created with the [Create New Row] button.

In the drop-down menu you select the dashboard to which you want to attach the row and enter a name for the row.

Creating a panel

If you have read in your data sources as described above, you can add selected data sources from the "Datasources" tab to a dashboard via the [Add Datasource To Dashboard] button.
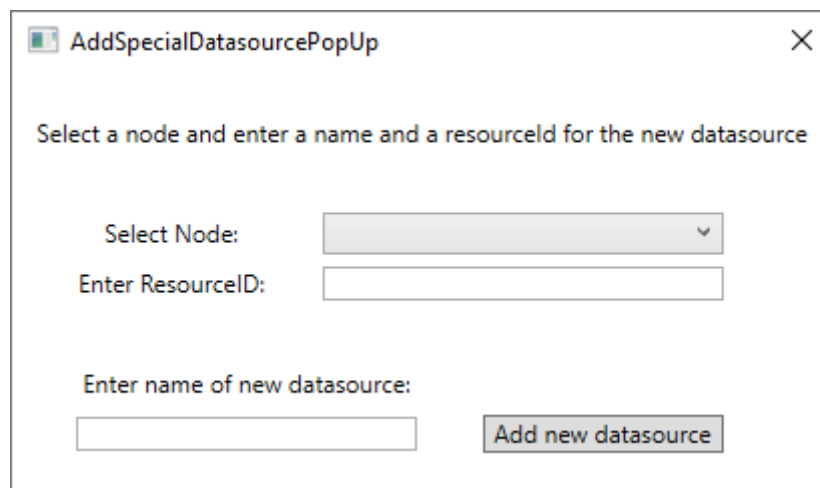

You select the dashboard, as well as the column ("Free Space" stands here for panels, which are above the first column) and confirm. Now you add a template to the data sources in the tab of the Dashboard via the [Set Template For Datasource] button.

Special data sources

Some data sources, such as "CPU", are not listed in the standard data sources. These can be added manually via the [Add Special Datasource] button.

Select the node to which the data source is assigned and specify a name and, if necessary, a ResourceID. For data sources such as "CPU", where the ResourceID does not differ between different nodes, it is not necessary to specify a ResourceID, provided that the appropriate template has been created and imported accordingly. It is also possible to create panels in which the ResourceIDs do not differ between the nodes, and you access different ResourceIDs (e.g., "Memory" with physical/virtual memory).

Organizing the rows/panels

With the button [Show/Hide Move Buttons] some buttons can be shown / hidden, with which the rows/panels can be rearranged.

The buttons [Move Down Selected Datasources] and [Move Up Selected Datasources] can be used to move up/down the data sources (panels), the buttons [Move Left Selected Row] and [Move Right Selected Row] are for moving up (left) and down (right) the rows. The rows/panels will be displayed in the interface later in the same order as in the program.

Editing dashboards

With the button [Edit Dashboards] you can open a window in which dashboards and rows can be renamed and deleted.

Row Name	Dashboard Name	

[Delete Dashboard] [Delete Row] [Rename Row] [Rename Dashboard]
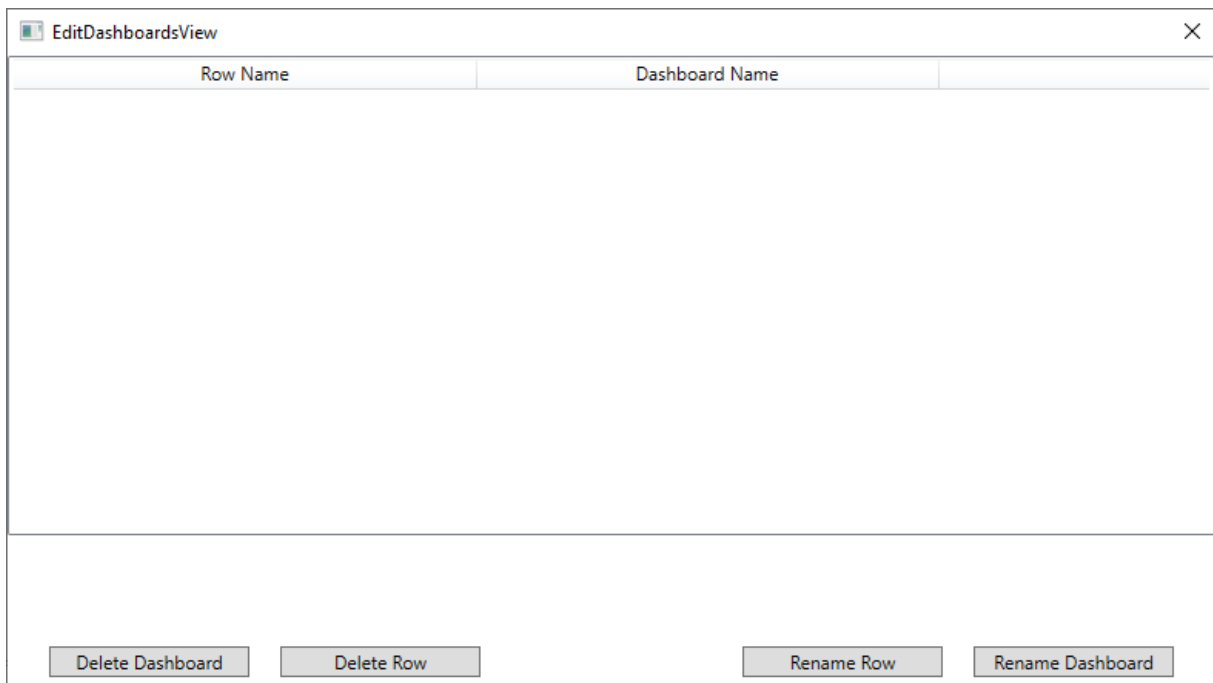
The buttons [Delete Dashboard] and [Delete Row] are corresponding for deleting dashboards/columns, the buttons [Rename Dashboard] and [Rename Row] correspondingly for renaming.
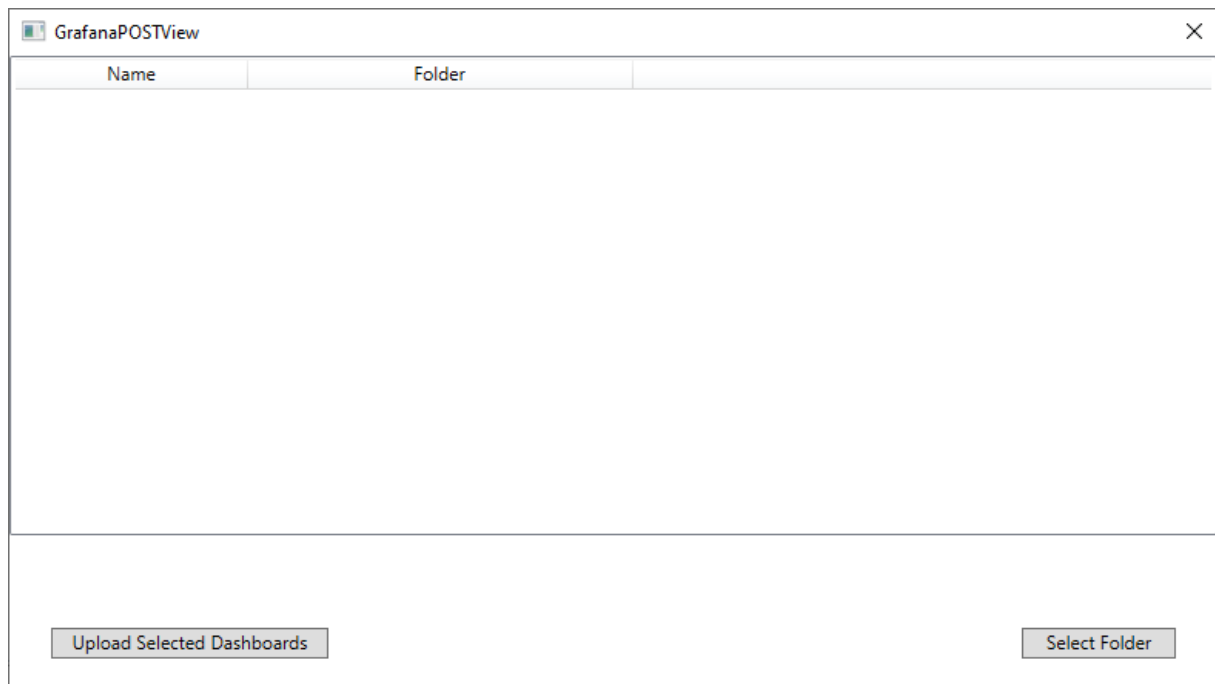
The export

Once you have created your dashboard(s) and set all templates (important!), you can open a text view of the resulting JSON file via the [Create JSON Output] button.
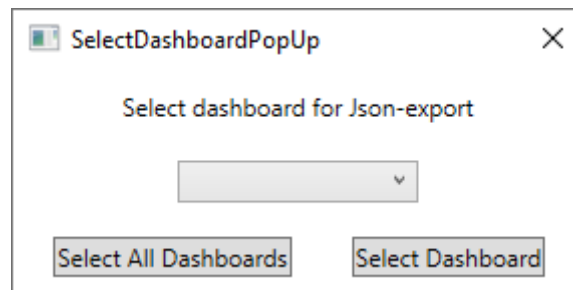
You can select the dashboards individually or display them all at once.

If you want to upload the whole thing to Grafana automatically, you can do this with the [POST Dashboards] button.

With [Select Folder] you can set a folder for your dashboards, if none is set, the dashboards end up in the "General" folder.

The [Upload Selected Dashboards] dashboards are sent via REST POST to the stored Grafana server. Then the answer of the server is displayed, so you can be sure that all has worked properly.

Software Requirements

The program was developed using Visual Studio in the .NET Framework 4.8, which is required to run the application.
Furthermore, the application requires network access to the corresponding servers (OpenNMS, Grafana).