

GrafanaDashboardCreator

Documentation_DE

Inhalt

Grundidee des Programms (Funktionsweise)	2
Wie speichert Grafana seine Dashboards/Panels?	2
Was tut das Programm nun?	2
Wie wird das Programm bedient?	3
Das Einlesen von Daten	3
Das Verwalten der Logindaten	4
Das Einlesen/Verwalten von Templates	5
Das Erstellen eines Dashboards	6
Das Erstellen einer Spalte	6
Das Erstellen eines Panels	7
Das Ordnen der Spalten/Panels	8
Das Bearbeiten von Dashboards	8
Der Export	9
Anforderungen der Software	9

Grundidee des Programms (Funktionsweise)

Da Grafana seine Dashboards, sowie die darin befindlichen Elemente, im JSON-Format speichert und zudem über eine REST-API verfügt, um solche JSON Objekte extern zu erstellen und dann per POST ins System einzupflegen, liegt es nahe, diesen Vorgang mit Hilfe eines Tools zu automatisieren. Um dabei jedoch nicht einen vollständig Dashboard-Creator selbst schreiben zu müssen, wurde bei dem Programm auf ein paar Tricks zurückgegriffen.

Wie speichert Grafana seine Dashboards/Panels?

Wie schon beschrieben, speichert Grafana seine Daten im JSON-Format. Für ein leeres Dashboard sieht das dann wie folgt aus:

Leeres Dashboard als JSON
<pre>{ "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 }</pre>

Was tut das Programm nun?

Das Programm nimmt sich solche „leeren Templates“ der JSON-Objekte von Grafana und füllt sie mit den gewünschten Informationen. Es ist außerdem in der Lage selbst Templates aus vorhandenen Dashboards zu extrahieren, um diese wieder verwenden zu können. Auf diese Weise benötigt man keinen vollständigen „Dashboard-Compiler“ und kann dennoch auf einfache Weise Dashboards automatisiert erstellen.

Wie wird das Programm bedient?

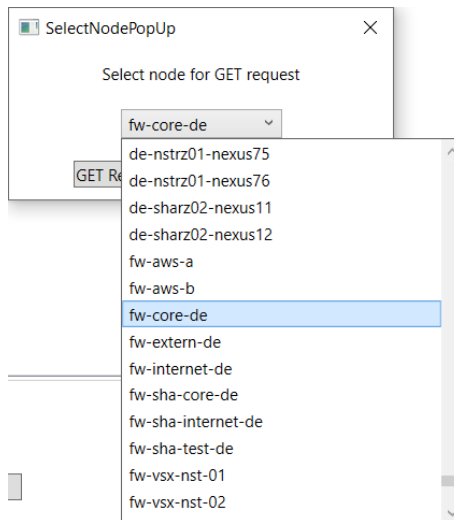
Das Einlesen von Daten

Das Einlesen der notwendigen Datenquellen und Nodes funktioniert mit Hilfe des

GET Resources

Buttons.

Dort werden zunächst die Nodes über die REST-API von OpenNMS abgefragt.



Nach auswählen eines Nodes aus dem Dropdown-Menü werden dann die zu diesem Node gehörenden Datenquellen abgefragt und im Programm im Reiter „Datasource“ angezeigt. Es ist möglich die Datenquellen mehrerer Nodes gleichzeitig zu laden, diese werden dann zunächst nach ihren zugehörigen Nodes und dann lexikographisch nach ihren Labels sortiert.

Label	Node	ResourceID	NodeID
eth13 (192.168.197.81, 100 Mbps)	fw-core-de	interfaceSnmp[eth13-3ca82a5cf996]	Firewall-Checkpoint:1648470074012
eth15 (192.168.198.33, 1 Gbps)	fw-core-de	interfaceSnmp[eth15-3ca82a5cf994]	Firewall-Checkpoint:1648470074012
eth2.1052 (192.168.197.113, 1 Gbps)	fw-core-de	interfaceSnmp[eth2_1052-98f2b33f5b36]	Firewall-Checkpoint:1648470074012
eth3 (192.168.197.49, 1 Gbps)	fw-core-de	interfaceSnmp[eth3-98f2b33f5b37]	Firewall-Checkpoint:1648470074012
eth7 (192.168.197.145, 1 Gbps)	fw-core-de	interfaceSnmp[eth7-3ca82a5cfd4c]	Firewall-Checkpoint:1648470074012
eth8.333 (192.168.197.129, 1 Gbps)	fw-core-de	interfaceSnmp[eth8_333-3ca82a5cfd13]	Firewall-Checkpoint:1648470074012
eth8.383 (192.168.197.161, 1 Gbps)	fw-core-de	interfaceSnmp[eth8_383-3ca82a5cfd13]	Firewall-Checkpoint:1648470074012
eth8.69 (192.168.197.177, 1 Gbps)	fw-core-de	interfaceSnmp[eth8_69-3ca82a5cfd13]	Firewall-Checkpoint:1648470074012
eth8.72 (192.168.197.17, 1 Gbps)	fw-core-de	interfaceSnmp[eth8_72-3ca82a5cfd13]	Firewall-Checkpoint:1648470074012
wrp128 (192.168.196.81)	fw-core-de	interfaceSnmp[wrp128-0012c1688000]	Firewall-Checkpoint:1648470074012
Response Time for 10.3.12.1	fw-core-de	responseTime[10.3.12.1]	Firewall-Checkpoint:1648470074012
cciss/c0d0 (index cciss-c0d0)	fw-sha-core-de	diskIOIndex[cciss-c0d0]	Firewall-Checkpoint:1648470582094
cciss/c0d0p1 (index cciss-c0d0p1)	fw-sha-core-de	diskIOIndex[cciss-c0d0p1]	Firewall-Checkpoint:1648470582094
cciss/c0d0p2 (index cciss-c0d0p2)	fw-sha-core-de	diskIOIndex[cciss-c0d0p2]	Firewall-Checkpoint:1648470582094
cciss/c0d0p3 (index cciss-c0d0p3)	fw-sha-core-de	diskIOIndex[cciss-c0d0p3]	Firewall-Checkpoint:1648470582094
dm-0 (index dm-0)	fw-sha-core-de	diskIOIndex[dm-0]	Firewall-Checkpoint:1648470582094
dm-1 (index dm-1)	fw-sha-core-de	diskIOIndex[dm-1]	Firewall-Checkpoint:1648470582094
dm-2 (index dm-2)	fw-sha-core-de	diskIOIndex[dm-2]	Firewall-Checkpoint:1648470582094
dm-3 (index dm-3)	fw-sha-core-de	diskIOIndex[dm-3]	Firewall-Checkpoint:1648470582094
dm-4 (index dm-4)	fw-sha-core-de	diskIOIndex[dm-4]	Firewall-Checkpoint:1648470582094
dm-5 (index dm-5)	fw-sha-core-de	diskIOIndex[dm-5]	Firewall-Checkpoint:1648470582094
dm-6 (index dm-6)	fw-sha-core-de	diskIOIndex[dm-6]	Firewall-Checkpoint:1648470582094
md0 (index md0)	fw-sha-core-de	diskIOIndex[md0]	Firewall-Checkpoint:1648470582094

POST Dashboards

Create JSON Output

Open Credentials Viewer

Open Template Viewer

Show/Hide Move Buttons

Remove Datasource From Dashboard

Edit Dashboards

Add Special Datasource

Set Template For Datasource

Add Datasource To Dashboard

GET Resources

Create New Row

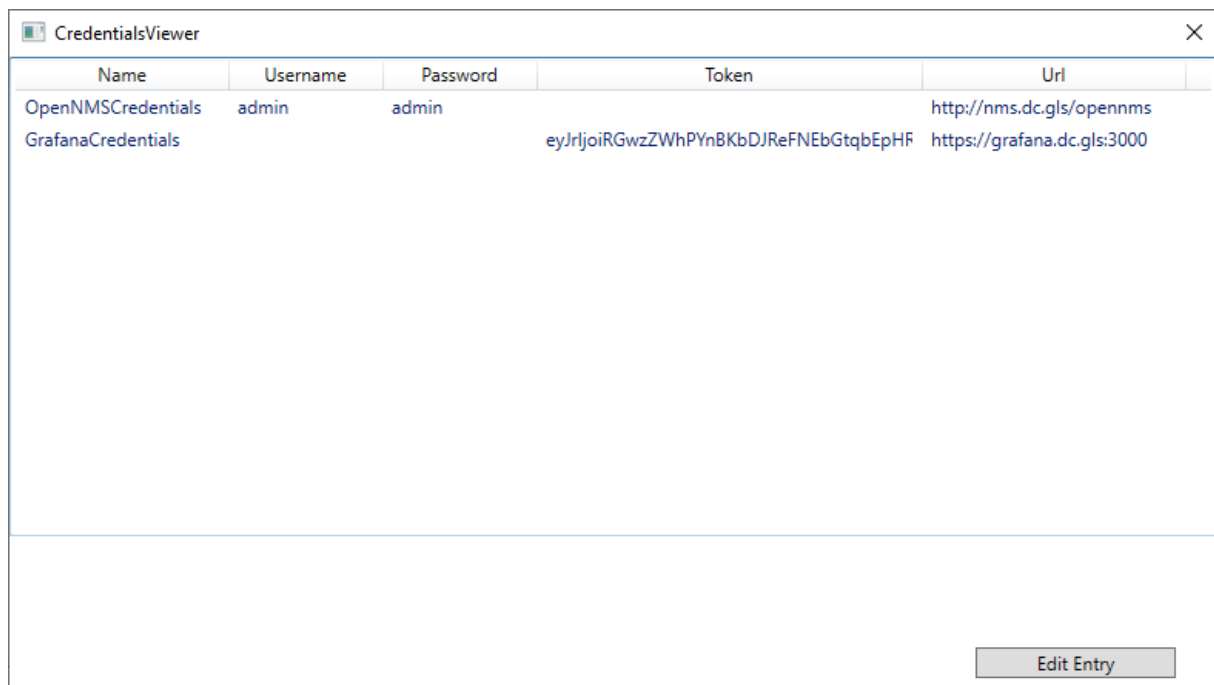
Create New Dashboard

Das Verwalten der Logindaten

Um eine Verbindung zu solchen APIs aufbauen zu können benötigt das Programm die entsprechenden Daten in Form einer URL, sowie der Login-Daten. Diese werden in einer Datei im „Datastore/Credentials“ Unterverzeichnis gespeichert und lassen sich über den

Open Credentials Viewer

Button verwalten.



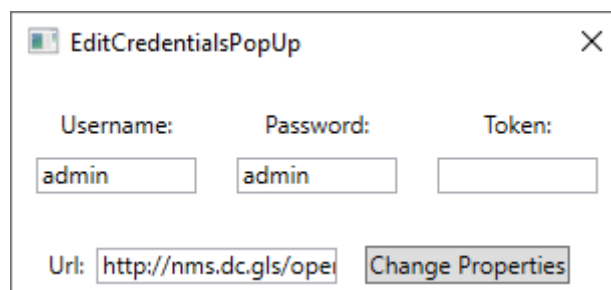
The screenshot shows a window titled "CredentialsViewer" with a close button (X) in the top right corner. It contains a table with the following data:

Name	Username	Password	Token	Url
OpenNMSCredentials	admin	admin		http://nms.dc.gls/opennms
GrafanaCredentials			eyJrljoiRGwzZWwPYnBKbDJReFNEbGtqbEpHF	https://grafana.dc.gls:3000

At the bottom right of the window is an "Edit Entry" button.

Zum Bearbeiten eines Eintrags ein den Eintrag auswählen und auf den **Edit Entry** Button klicken. In sich öffnenden Fenster wird unter „Username“ der Benutzername, unter „Password“ das Passwort und unter „Token“ das Token zur Authentifizierung am Server verlangt.

Wichtig: OpenNMS erwartet einen Benutzernamen, sowie ein Passwort, Grafana benötigt ein Token! Unter „Url“ wird die Adresse des Servers erwartet, wichtig dabei: Es wird die nicht die API-Adresse benötigt! Im Falle von OpenNMS wäre die API-Adresse etwas vom Format „<http://opennmsserver/opennms/rest/>“, es wird hier nur der Teil vor den „/rest/“ benötigt, also etwas von der Form „<http://opennmsserver:8980/opennms/>“!



The screenshot shows a dialog box titled "EditCredentialsPopUp" with a close button (X) in the top right corner. It contains the following fields and buttons:

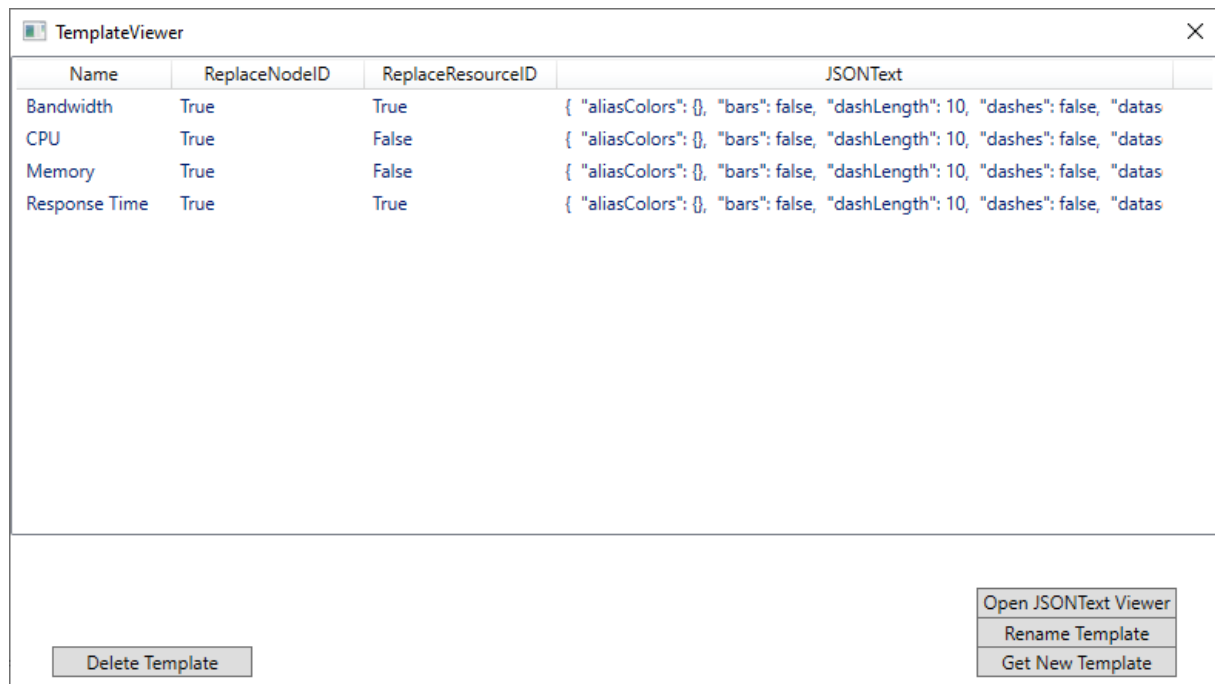
Username: Password: Token:

Url: **Change Properties**

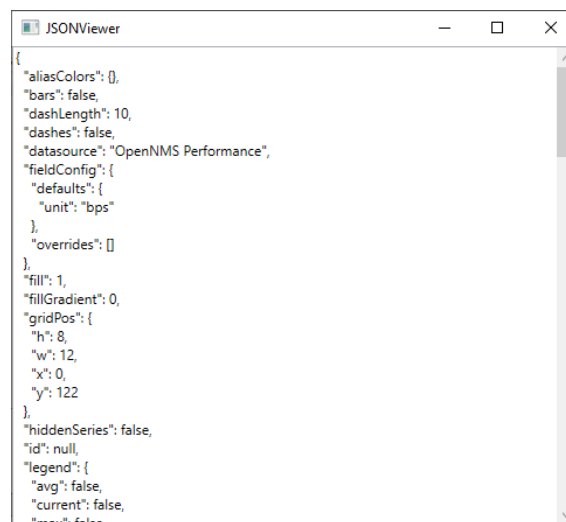
Das Einlesen/Verwalten von Templates

Das Programm arbeitet intern, wie schon beschrieben, mit Templates der zu erstellenden Objekte. Das bedeutet, es nimmt sich ein fertiges Objekt eines Typs und setzt dort die notwendigen Informationen, wie Titel, ID, Node-/ResourceID ein. Für die Objekte „Dashboard“, „Row“ sowie „Folder“ sind Templates im Programm hinterlegt. Für die Panels müssen diese jedoch eigens importiert werden. Dazu erstellt man in der Grafana Oberfläche ein Dashboard mit dem Panel, wie man es gerne haben möchte und exportiert es über die Weboberfläche in eine JSON-Datei.

Im Programm öffnet man über den [Open Template Viewer](#) Button die Oberfläche zum Verwalten der Templates.

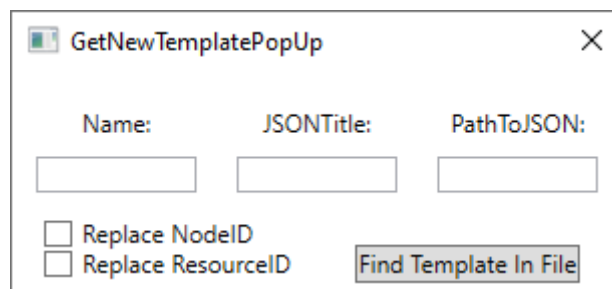


Dort kann man mit Hilfe des [Delete Template](#) Buttons das gewählte Template löschen, mit Hilfe des [Rename Template](#) Buttons kann das gewählte Template umbenannt werden und über den [Open JSONText Viewer](#) Button kann der JSON-Text des Templates überprüft werden.



Über den **Get New Template** Button kann ein neues Template geladen werden. Dabei wird unter „Name“ ein frei wählbarer Name für die interne Anzeige erwartet, unter JSONTitle wird der Titel des Templates, wie man es in der Weboberfläche erstellt hat erwartet und unter „PathToJSON“ wird der vollständige Pfad zur oben erstellten JSON-Datei erwartet.

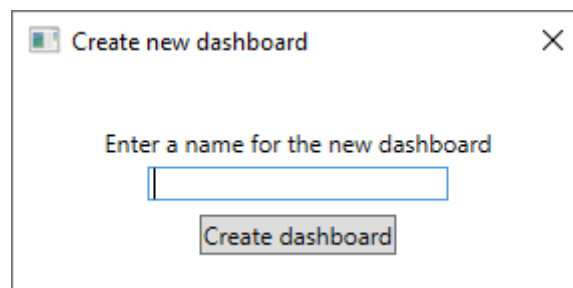
Über die Checkboxen lässt sich bestimmen, welche IDs der Datenquelle später beim Erstellen des Exports eingetragen werden müssen. Die NodeID sollte im Regelfall immer eingetragen werden (hier wurde lediglich die Option für eventuelle Sonderfälle offengelassen), die ResourceID muss nicht für Templates wie „Memory“ und „CPU“ ersetzt werden. Also immer dann, wenn die ResourceID bei allen Panels aller Nodes die gleiche wäre. Möchte man statt dessen ein Template einer Interface Ressource laden, muss diese später im Template entsprechend der zu erstellenden Panel für andere Ressourcen angepasst werden.



The dialog box titled "GetNewTemplatePopUp" contains three input fields labeled "Name:", "JSONTitle:", and "PathToJSON:". Below these fields are two checkboxes: "Replace NodeID" and "Replace ResourceID". A "Find Template In File" button is located at the bottom right of the dialog.

Das Erstellen eines Dashboards

Über den Button **Create New Dashboard** lässt sich ein neues Dashboard erstellen.

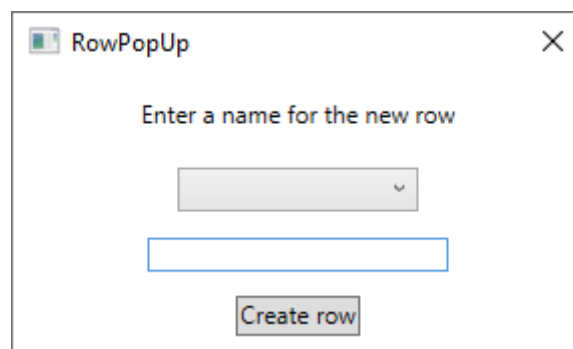


The dialog box titled "Create new dashboard" contains a text input field with the placeholder text "Enter a name for the new dashboard". Below the input field is a "Create dashboard" button.

Man gibt den Namen des neuen Dashboards ein und bestätigt.

Das Erstellen einer Spalte

Eine neue Spalte lässt sich mit Hilfe des **Create New Row** Buttons erstellen.

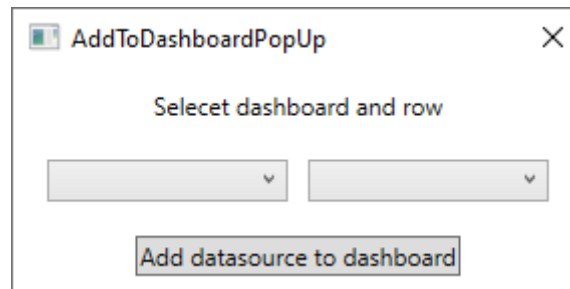


The dialog box titled "RowPopUp" contains a dropdown menu with the placeholder text "Enter a name for the new row". Below the dropdown menu is a text input field. At the bottom of the dialog is a "Create row" button.

Im Dropdown-Menü wählt man das Dashboard, welchem man die Spalte anhängen möchte und gibt einen Namen für die Spalte ein.

Das Erstellen eines Panels

Hat man seine Datenquellen wie oben beschrieben eingelesen, so kann man über den Button **Add Datasource To Dashboard** gewählte Datenquellen aus dem Reiter „Datasources“ einem Dashboard hinzufügen.

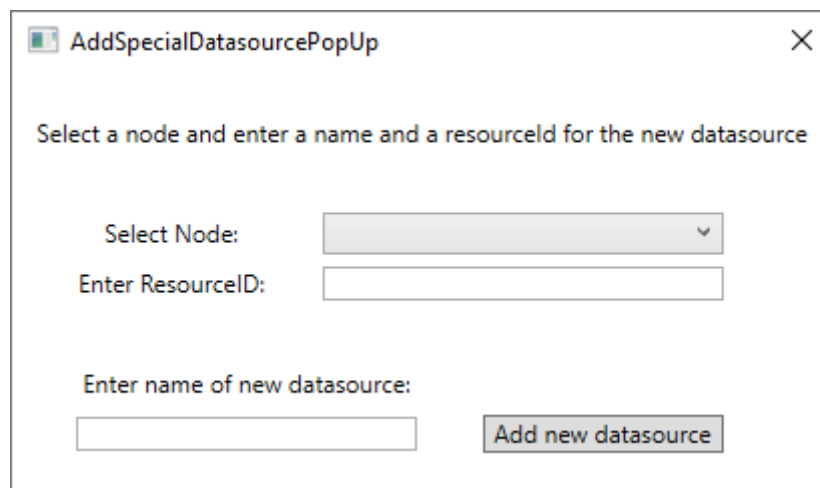


The dialog box titled 'AddToDashboardPopUp' contains the instruction 'Select dashboard and row'. Below this, there are two dropdown menus. At the bottom of the dialog is a button labeled 'Add datasource to dashboard'.

Man wählt das Dashboard, sowie die Spalte („Free Space“ steht hier für Panels, welche über der ersten Spalte stehen) und bestätigt. Jetzt fügt man den Datenquellen im Reiter des Dashboard noch über den Button **Set Template For Datasource** ein Template hinzu.

Spezielle Datenquellen

Manche Datenquellen, wie „CPU“, werden nicht in den Standard-Datenquellen geführt. Solche lassen sich nachträglich über den Button **Add Special Datasource** manuell hinzufügen.



The dialog box titled 'AddSpecialDatasourcePopUp' contains the instruction 'Select a node and enter a name and a resourceid for the new datasource'. It features three input fields: 'Select Node:' with a dropdown menu, 'Enter ResourceID:' with a text box, and 'Enter name of new datasource:' with a text box. An 'Add new datasource' button is located at the bottom right.

Man wählt den Node, welchem die Datenquelle zugewiesen ist und gibt einen Namen sowie, sofern notwendig, eine ResourceID an. Für Datenquellen wie „CPU“, bei welchen sich die ResourceID zwischen unterschiedlichen Nodes nicht unterscheidet ist es nicht notwendig eine ResourceID anzugeben, sofern das dazu passende Template entsprechend angelegt wurde. Auch ist es so möglich Panels zu erstellen, bei welchen sich die ResourceIDs zwischen den Nodes nicht unterscheiden und man auf unterschiedliche ResourceIDs zugreift (Bspw. „Memory“ mit Physical-/Virtualmemory).

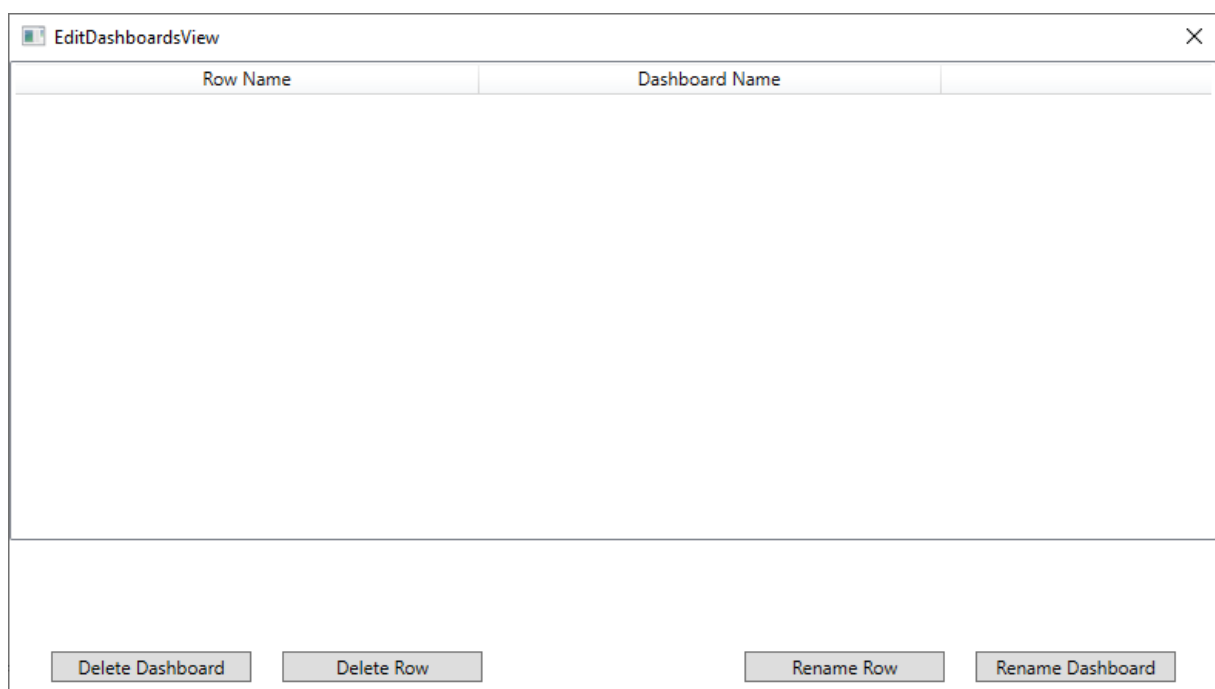
Das Ordnen der Spalten/Panels

Mit dem Button **Show/Hide Move Buttons** können einige Button ein-/ausgeblendet werden, mit welchen sich die Spalten/Panel neu anordnen lassen.

Die Buttons **Move Down Selected Datasources** und **Move Up Selected Datasources** lassen sich zum rauf/runter schieben der Datasources (Panels) nutzen, die Buttons **Move Left Selected Row** und **Move Right Selected Row** sind zum rauf (links) und runter (rechts) schieben der Spalten. Die Spalten/Panels werden in der Oberfläche später in der gleichen Reihenfolge wie im Programm zu sehen sein.

Das Bearbeiten von Dashboards

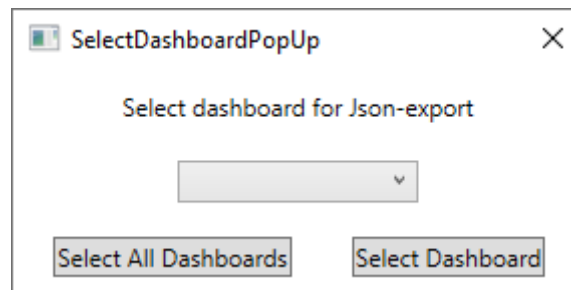
Mit dem Button **Edit Dashboards** lässt sich ein Fenster öffnen, in welchem sich Dashboards und Spalten umbenennen und löschen lassen.



Die Buttons **Delete Dashboard** und **Delete Row** sind entsprechend zum Löschen von Dashboards/Spalten, die Buttons **Rename Dashboard** und **Rename Row** entsprechend zum Umbenennen.

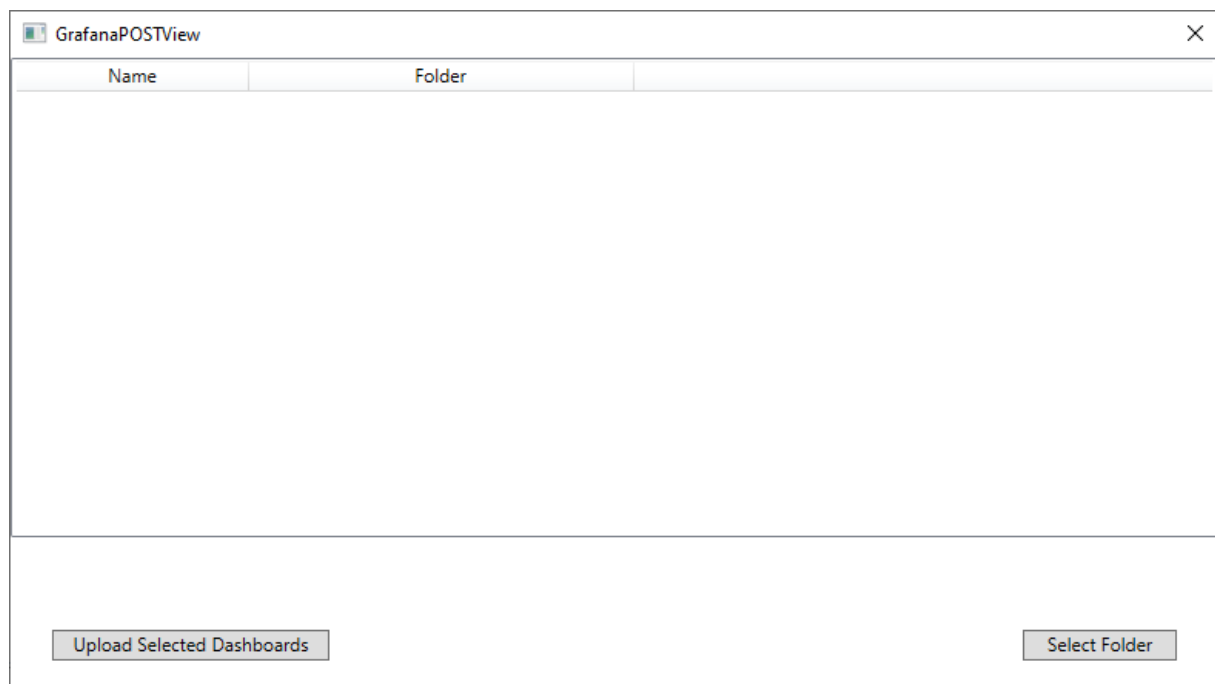
Der Export

Hat man sein(e) Dashboard(s) erstellt und alle Templates gesetzt (wichtig!), kann man über den Button **Create JSON Output** eine Textansicht der resultierenden JSON-Datei einsehen.



Man kann die Dashboards einzeln wählen, oder alle gleichzeitig anzeigen lassen.

Möchte man das ganze automatisch zu Grafana hochladen, so kann man dies mit Hilfe des **POST Dashboards** Buttons.



Mit **Select Folder** kann man ein Folder für seine Dashboards festlegen, wird keines festgelegt, landen die Dashboards im „General“ Folder.

Über **Upload Selected Dashboards** werden die Dashboards per REST-POST an den hinterlegten Grafana-Server geschickt. Danach wird noch die Antwort des Servers angezeigt, so kann man sicher gehen, dass alles funktioniert hat.

Anforderungen der Software

Das Programm wurde mit Visual Studio im .NET Framework 4.8 entwickelt, selbiges ist entsprechend erforderlich, um die Anwendung ausführen zu können.

Des Weiteren benötigt die Anwendung Netzwerkzugriff zu den entsprechenden Servern (OpenNMS, Grafana).