GrafanaDashboardCreator  
Documentation\_DE

Inhalt

[Grundidee des Programms (Funktionsweise) 2](#_Toc103334633)

[Wie speichert Grafana seine Dashboards/Panels? 2](#_Toc103334634)

[Was tut das Programm nun? 2](#_Toc103334635)

[Wie wird das Programm bedient? 3](#_Toc103334636)

[Das Einlesen von Daten 3](#_Toc103334637)

[Das Verwalten der Logindaten 4](#_Toc103334638)

[Das Einlesen/Verwalten von Templates 5](#_Toc103334639)

[Das Erstellen eines Dashboards 6](#_Toc103334640)

[Das Erstellen einer Spalte 6](#_Toc103334641)

[Das Erstellen eines Panels 7](#_Toc103334642)

[Das Ordnen der Spalten/Panels 8](#_Toc103334643)

[Das Bearbeiten von Dashboards 8](#_Toc103334644)

[Der Export 9](#_Toc103334645)

[Anforderungen der Software 9](#_Toc103334646)

# 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 |
| {  "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  } |

## Was tut das Programm nun?

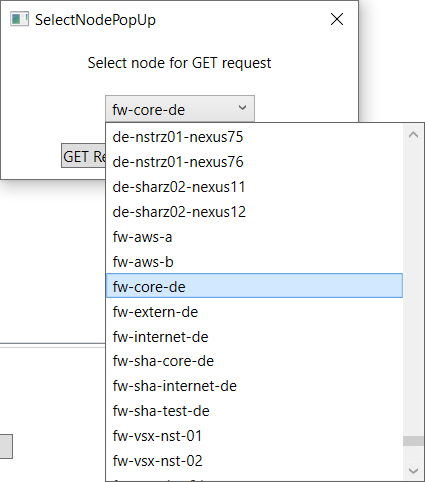
Das Programm nimmt sich solche „leeren Templates“ der JSON-Objekte von Grafana und füllt sie mit den gewünschten Informationen. Es ist außerdem in der Lage selbst Templates aus vorhandenen Dashboards zu extrahieren, um diese wieder verwenden zu können. Auf diese Weise benötigt man keinen vollständigen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

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

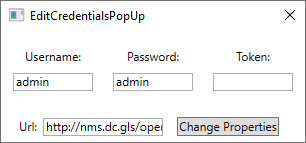
### Das Verwalten der Logindaten

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  Button verwalten.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Zum Bearbeiten eines Eintrags ein den Eintrag auswählen und auf den  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>“!



## 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  Button die Oberfläche zum Verwalten der Tempaltes.

Ein Bild, das Text enthält.

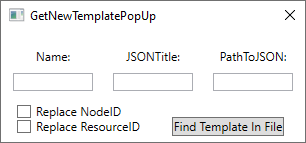
Automatisch generierte Beschreibung

Dort kann man mit Hilfe des  Buttons das gewählte Template löschen, mit Hilfe des  Buttons kann das gewählte Template umbenannt werden und über den  Button kann der JSON-Text des Templates überprüft werden.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Über den  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.



## Das Erstellen eines Dashboards

Über den Button  lässt sich ein neues Dashboard erstellen.

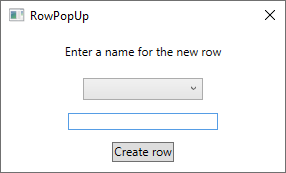
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Man gibt den Namen des neuen Dashboards ein und bestätigt.

### Das Erstellen einer Spalte

Eine neue Spalte lässt sich mit Hilfe des  Buttons erstellen.



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  gewählte Datenquellen aus dem Reiter „Datasources“ einem Dashboard hinzufügen.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

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 Deshboard noch über den Button  ein Template hinzu.

#### Spezielle Datenquellen

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

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

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  können einige Button ein-/ausgeblendet werden, mit welchen sich die Spalten/Panel neu anordnen lassen.  
Die Buttons  und  lassen sich zum rauf/runter schieben der Datasources (Panels) nutzen, die Buttons  und  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  lässt sich ein Fenster öffnen, in welchem sich Dashboards und Spalten umbenennen und löschen lassen.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Die Buttons  und  sind entsprechend zum Löschen von Dashboards/Spalten, die Buttons  und  entsprechend zum Umbenennen.

## Der Export

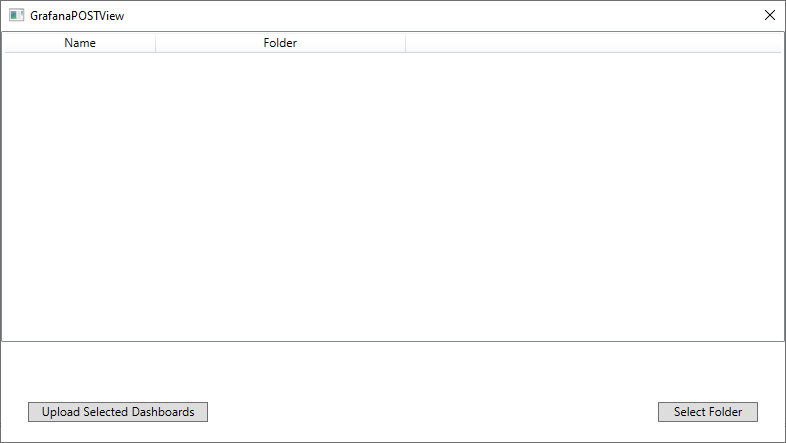
Hat man sein(e) Dashboard(s) erstellt und alle Templates gesetzt (wichtig!), kann man über den Button  eine Textansicht der resultierenden JSON-Datei einsehen.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

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  Buttons.



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