Moduldokumentation

(MOD)

(TINF18C, SWE I Praxisprojekt 2019/2020)

Modul

API

Project: Profinet DCP Client als WEB-Applikation

Customer: *Rentschler & Ewertz*
Rotebühlplatz 41
70178 Stuttgart

Supplier: Team 2 (Jannik Schwarz, Sinan Yurttadur, Noah Broß, Nicolas Breuninger, Marvin Sonntag, Rene Scholz)
Rotebühlplatz 41
70178 Stuttgart

Version	Date	Author	Comment
0.1	28.04.2020	Noah Broß	created
1.0	06.05.2020	Noah Broß	Final draft

1. Content

2. History

Version	Datum	Autor(en)	Kommentare
0.1	28.04.2020	Noah Broß	First draft
1.0	06.05.2020	Noah Broß	Final version

3. Scope

The Module Documentation (MOD) describes the architecture, the interfaces and the main features of the module. It also describes the module/component test including the results. It can also serve as a programming or integration manual for the module. If there are some risks related to the module itself, they shall be noted and commented within this document.

Die Moduldokumentation beschreibt die Architektur, die Schnittstellen und die Hauptmerkmale des Moduls. Außerdem werden die Modul bzw. Komponententests einschließlich der Ergebnisse beschrieben und dokumentiert. Die MOD dient bei Bedarf auch als Programmier- oder Integrationshandbuch für das Modul. Wenn bestimmte Risiken direkt mit der Verwendung des Moduls verknüpft sind, so sind sie in diesem Dokument zu benennen und zu kommentieren.

4. Module Requirements

4.1. User View

This module will be used to communicate to the frontend. It is the communication layer for the backend. The job is to send data to the frontend an receive requests from the user. Requests will be brought into a defined format and handled. It also sends back the results of the results.

4.2. Module Context

This module works with the NetworkScanner, ChangeRequest and NetworkService module. The NetworkScanner module is used to find devices in the network. The ChangeRequest module is used together with the NetworkScanner to configure devices.
The resulting information will be communicated back to the frontend over the NetworkService module.

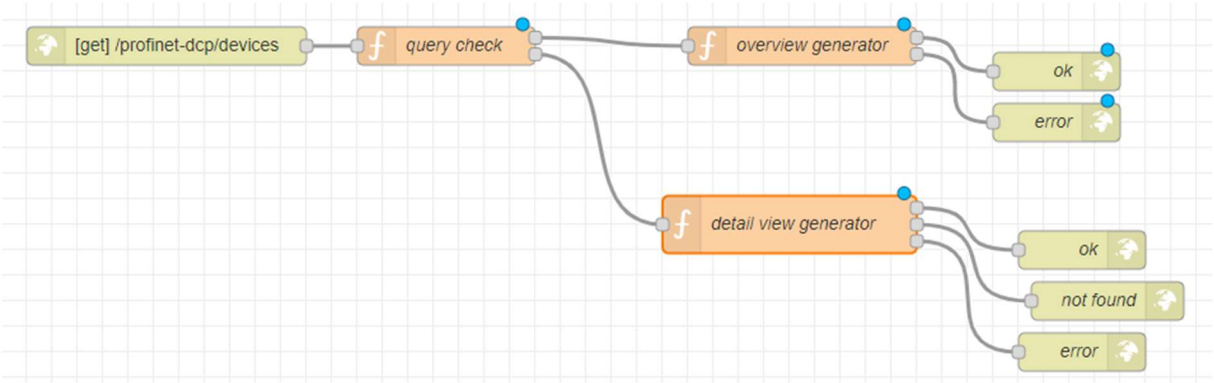
5. Analysis

This module is dependent on the NetworkScanner. If the NetworkScanner module does not return information about the network this module cannot send back information to the frontend. This will not result in an error for the frontend, but no correct information would be displayed.

6. Design

Solution:
The frontend needs data to show to the user. This module was created to enable the communication between the backend device detection service and the frontend display of the network data.

Module architecture:
The module is an API for the frontend. It analyses what data has to be processed and returns standardized data to the frontend.



APIs:
This module has a http-API. So, it is possible to get a result with a corresponding status code. If the requested operation was successful, the status code "200" can be returned together with a generated view.

6.1. Risks

What happens when this component fails?
The frontend will not get any new data to display. Already found devices would still be displayed though.

No third-party software is used.

Minimizing risk:
To minimize the risk of failure this module was built with simplicity in mind. Thus few points of failure can be built in.
If the frontend sends a wrong or broken request this module will not act on it. Each request is tested for validity to stop bad requests.

7. Implementation

This module was implemented with Node.JS and a framework “Node-red”.
Node-red allows this module to be developed visually based on flow-charts. This is used to make quick progress for the most, rough parts of the application. The finer code is implemented using JavaScript.

With Node-red the module was divided into multiple small modules that could be quickly generated using the UI. Details like how functions work were then directly coded by hand.

8. Module Test

8.1. Module Testreport

Test-ID	Pass/Fail	If failed: Test Observation	Date	Tester
12	Pass		14.05.2020	Rene Scholz
13	Pass		14.05.2020	Rene Scholz
14	Pass		14.05.2020	Rene Scholz
15	Pass		14.05.2020	Rene Scholz

9. Summary

Strenghts:
Simplicity and robustness.
Easy to debug because of error-codes.

Weaknesses:
Is dependent on data from the NetworkScanner module.

Other:
Blocks many errors already on its own by blocking false / broken requests.

Improvements:
Because of the simplicity more performance oriented structures like HashMaps were not used. So performance could still be improved. This should only be a issue if a lot of data is requested.

Extensions:
With a "Query Check" node you could add more generators for views.

10. Appendix

10.1. API Documentation (Swagger)