Table of Contents

Changelog

Version	Date	Author	Comment
1.0	04.11.2022	Maris Koch	Chapter 1
1.1	06.11.2022	Maris Koch	Chapter 2-6
1.2	10.05.2022	Maris Koch	Added implementation details
1.3	11.05.2022	Janin Ahlemeyer	Added Modul Tests

1 Scope

This module focuses on requesting the needed data from a chosen AASX server and refining it into a form, which the web interface and the nameplate generation can work with.

2 Glossary

- **AAS** - Asset Administration Shell
- **AASX** - file format to store an asset
- **AASX server** - server, that can store AAS assets and has a standardized API specified in the GitHub repository

3 Module Requirements

3.1 User View

The user does not see this module directly. Parts of the work of this module can be seen in the asset view. The data depicted for the chosen asset is based on the filtering of this module.

3.2 Requirements

DNG.GUI.007 Error handling

The system has an error handling. When the server is down or does not exist, the user shall be notified. Furthermore, the module needs to adapt to new and missing keys in the asset to be prepared for future updates of the aas standards.

DNG.PERF.001 Performance

The software should maintain a high performance in terms of how fast a website loads including the time fetching data from the server and displaying it.

DNG.REL.001 Reliability

The application needs to be reliable in terms of containing the right information.

DNG.MAIN.001 Maintainability

Each team member shall be able to read and understand the code as well as knowing how to make changes. Additionally, developers not belonging to the team shall be able to do so as well after reading the code for five hours.

DNG.LIC.001

The product is an open source software thus a license for publishing it is required.

3.3 Module Context

When a server address is inserted into the search bar on the start page, the server address is forwarded to this module, where the data requesting and refining takes place. This module then requests all assets that this server can provide. The data is then returned to the web interface (MOD01), which then creates the asset list from it. This module provides one array which contains many assets.

```
[
  {
    // asset1
  },
  {
    // asset2
  },
  {
    // asset3
  },
  {
    // ...
  }
]
```

The asset mentioned in the format above are defined as follows. The following data is an example asset. On all levels of the json, keys can be added or removed depending on the level of detail of the asset. The web interface dynamically adapts to new keys being present or keys missing. Some urls have been shortened for presentation purposes.

```
{
  "idShort": "AAS_Type_CD55B20_50",
  "id": "www.example.com/ids/aas/7031_8082_3022_7912",
  "num": 8,
  "productImages": [
    "<url>"
  ],
  "Nameplate": {
    "idShort": "Nameplate",
    "id": "www.example.com/ids/sm/0000_4121_5022_2603",
    "ManufacturerName": "SMC",
    "ManufacturerProductDesignation": "Kompaktzylinder nach ISO 21287",
    "ManufacturerProductFamily": "C55",
    "OrderCode": "CD55B20-50",
    "SerialNumber": "",
```

```
    "YearOfConstruction": "2022",
    "Address": {
      "Department": "Kontakt zu SMC",
      "Street": "Boschring 13-15",
      "ZipCode": "63329",
      "City_Town": "Egelsbach",
      "State_County": "Deutschland",
      "VATNumber": "DE 0123456789",
      "AddressOfAdditionalLink": "info@smc.de",
      "Phone": {
        "TelephoneNumber": "+49 (0) 61 03 / 402 - 0",
        "TypeOfTelephone": ""
      }
    },
    "Markings": {
      "Marking00": {
        "MarkingName": "nach EU-Maschinen-Richtlinie",
        "FilePath": "<url>",
        "MarkingFile": "/aasx/Nameplate/CE_Marking_2016.png"
      },
      "Marking01": {
        "MarkingName": "nach EU-RoHS-Richtlinie",
        "FilePath": "<url>"
      },
      "Marking02": {
        "MarkingName": "nach EU-EMV-Richtlinie",
        "FilePath": "<url>",
        "MarkingFile": "/aasx/Nameplate/CE_Marking_2016.png"
      },
      "Marking03": {
        "MarkingName": "RCM Mark"
      },
      "Marking04": {
        "MarkingName": "c UL us - Listed (OL)"
      },
      "Marking05": {
        "MarkingName": "TÜV"
      }
    }
  },
  "ArticleInformation": {
    // more key-value pairs
  },
  "ContactInformation": {
    // more key-value pairs
  }
  // more submodels
}
```

4 Analysis

The Data Acquisition module needs to be the working part in between the web interface and the nameplate generation. Both other modules rely on the refined data that this module returns. The web interface needs the parsed json object for all available assets on the server to present the asset list and the extracted data from the *nameplate* submodel of a chosen asset to display the asset view. The nameplate generation needs the refined data from the *nameplate* submodel to generate the nameplate including the QR-code. Hence, robustness and reliability is very important.

This module needs to provide an interface, that provides the necessary information in form of json object from a promise resolution for the web interface to display the asset list and asset view.

5 Design

This module was designed to do the data refining. Hence, MOD01 and MOD03 do not have any data retrieval from the aasx server selected by the user implemented. The other two modules focus only on the visual representation of the data in form of the web interface and the nameplate generation.

One of this modules priorities is to work with different api version. To determine the api version of the server the user has selected, an api probing is performed. As described in the module context, there is only one interface of the module supplying MOD01 and MOD03 with the data necessary for the web interface and the nameplate generation.

5.1 Risk

Due to the fact, that the standards are under continues advancement, a breaking change in the api could lead to some parts of this module to be rewritten. Making this module highly dynamic and being able to work with varying levels of detail of an asset, this risk is considered minor. This means that it should be able to handle variations in the data returned e.g. new keys being added or current keys being removed.

6 Implementation

The JavaScript *fetch* API is used to do the requests to the aasx server inputted by the user on the home page.

All the available assets that a server provides are requested from the `<serverAddress>/shells` endpoint. The data acquisition is separated into two major components, the *DataExtractor* and the *DataRefinery*.

DataExtractor

The DataExtractor is a JavaScript class that takes a submodel as an input for the constructor to be initialized. Within this class there are two main functions: `extractAllDataV1()` and `extractAllDataV3()`. These two functions extract all data data out of the complex structure returned by the aasx server for their respective api version. This reduces the complexity of the response from the server to a minimum for this application. Both of these functions are recursive. By that, they automatically adapt to a change in the submodel depth, because the function is called for every nested submodel.

DataRefinery

The DataRefinery is also implemented as a JavaScript class which is constructed from the server address provided by the user. It implements numerous functionalities. One of them is the api probing through the function `analyzeApiVersion()`. It works through looking in the submodels for the `type` key. This key is a specific feature of the V3 api. Hence, if it is present, it can be concluded that the server provides the V3 api, otherwise it is the V1 api.

Core of the entire module is the private function `getDataFromServer()`. This function is the one that makes the *fetch* call the to aasx server given by the user. It has advanced error handling implemented to detect if the given server does not respond as expected.

Also in this class implemented is the function `loadAssetRef()`. It extracts the AssetRef key out of an asset which is demanded by the IEC 63365 standard.

Another important implementation of the DataRefinery is the `getFullAASList()` function. It coordinates the reformatting of the raw data by instantiating instances of the DataExtractor. With its already described functionalities, it breaks down the complex response of the aasx server.

7 Module Tests

In this section nearly all requirements will be tested separately on their functionality. Some requirements are not tested, because they have no function to test on.

7.1 Module Test Plan

Req. - ID	Functionality
DNG.GUI.007	Error handling
DNG.PERF.001	Performance
DNG.REL.001	Reliability
DNG.MAIN.001	Maintainability

7.2 Module Test Report

Req. - ID	Pass / Fail	When failed: Observation	Tester
DNG.GUI.007	PASS		Janin Ahlemeyer
DNG.PERF.001	PASS		Janin Ahlemeyer
DNG.REL.001	PASS		Janin Ahlemeyer
DNG.MAIN.001	PASS		Janin Ahlemeyer

8 Summary

The DataAcquisition module is responsible for fetching the assets from the given aasx server and reducing the response down to the bare content needed by MOD01 and MOD03. It implements api probing to seamlessly work with V1 and V3 apis without any adaption in any other module. With the advanced error handling, this module is an error resilient component of the Nameplate Generator as a whole.