MOD05: Support of Bill of Material

TINF22F, Software Engineering I
Praxisproject 2023/24

Project

AAS-Webclient

Customer

Markus Rentschler, Ivan Bogicevic

Lerchenstraße 1, 70178 Stuttgart

Supplier

Armin Taktar (Project Leader)

Lara Lorke (System Architect)

David Bauer (Product Manager)

Ümmühan Ay (Documentation)

Rafael Sancho Pernas (Developer)

Kyle Zieher (Test Manager)

Author

Lara Lorke 10.05.2024

CONTENTS

1 Scope

The module "Support of Bill of Material" is responsible for processing and displaying the Hierarchical Structures enabling Bills of Material concerning a AAS object.

2 Glossary

- BoM: Bill of Material
-

3 Module Requirements

3.1 User View

The user has selected an AAS server and an AAS object he wants to display. When clicking the “Assetsubmodules” button all available submodules for the AAS object are shown with their names. If the AAS possesses the submodule “Bill of Material” the parts of an AAS are listed

3.2 Requirements

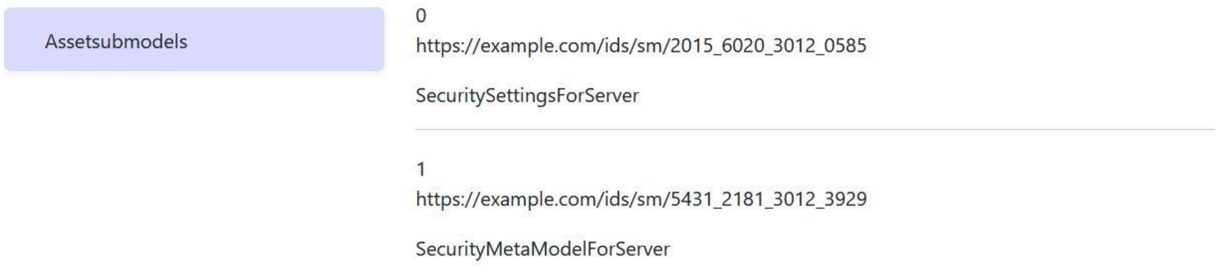
REQ/ Select submodules of AAS and retrieve submodule elements if the submodule’s name is “Bill of Material”

4 Analysis

The backend already processes the submodules for an AAS object and fills in the submodule list. For each submodule a request to the chosen server needs to be made to receive the kind of submodule as well as it’s elements.

5 Design

When the User clicks on the submodule the button “Assetsubmodels” is available:



0
https://example.com/ids/sm/2015_6020_3012_0585
SecuritySettingsForServer

1
https://example.com/ids/sm/5431_2181_3012_3929
SecurityMetaModelForServer

By clicking onto the button further information about the available submodels is displayed. If the submodel is the “Bill of Material” it’s elements are listed below:

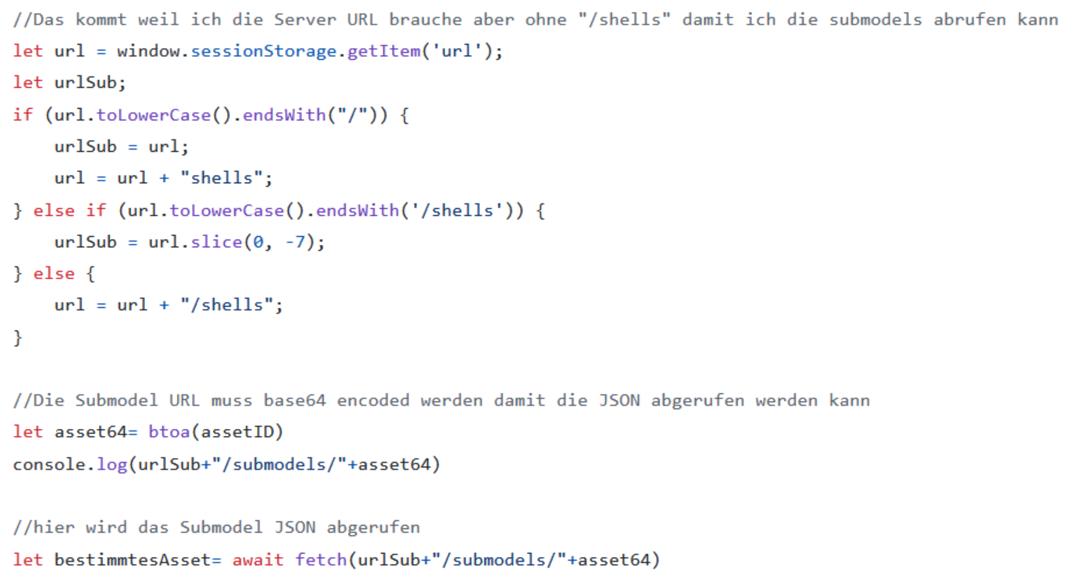
5
https://aas.digitaltwin-vdma.org/sm/1240_3113_3022_4726

BillOfMaterial

Vdma Robotics Demonstrator

6 Implementation

The function "getSubmodel(value)" (getSubmodel(value) in assetBody.js) in the assetBody receives the asset's submodelid and creates an URL witch consist of the chosen server url +"/submodels"+ the submodelid in base64.

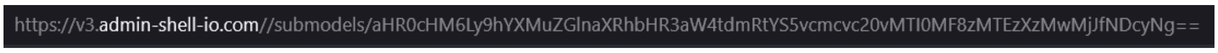


```
//Das kommt weil ich die Server URL brauche aber ohne "/shells" damit ich die submodels abrufen kann
let url = window.sessionStorage.getItem('url');
let urlSub;
if (url.toLowerCase().endsWith("/")) {
    urlSub = url;
    url = url + "shells";
} else if (url.toLowerCase().endsWith('/shells')) {
    urlSub = url.slice(0, -7);
} else {
    url = url + "/shells";
}

//Die Submodel URL muss base64 encoded werden damit die JSON abgerufen werden kann
let asset64= btoa(assetID)
console.log(urlSub+"/submodels/"+asset64)

//hier wird das Submodel JSON abgerufen
let bestimmtesAsset= await fetch(urlSub+"/submodels/"+asset64)
```

The URL looks like this:



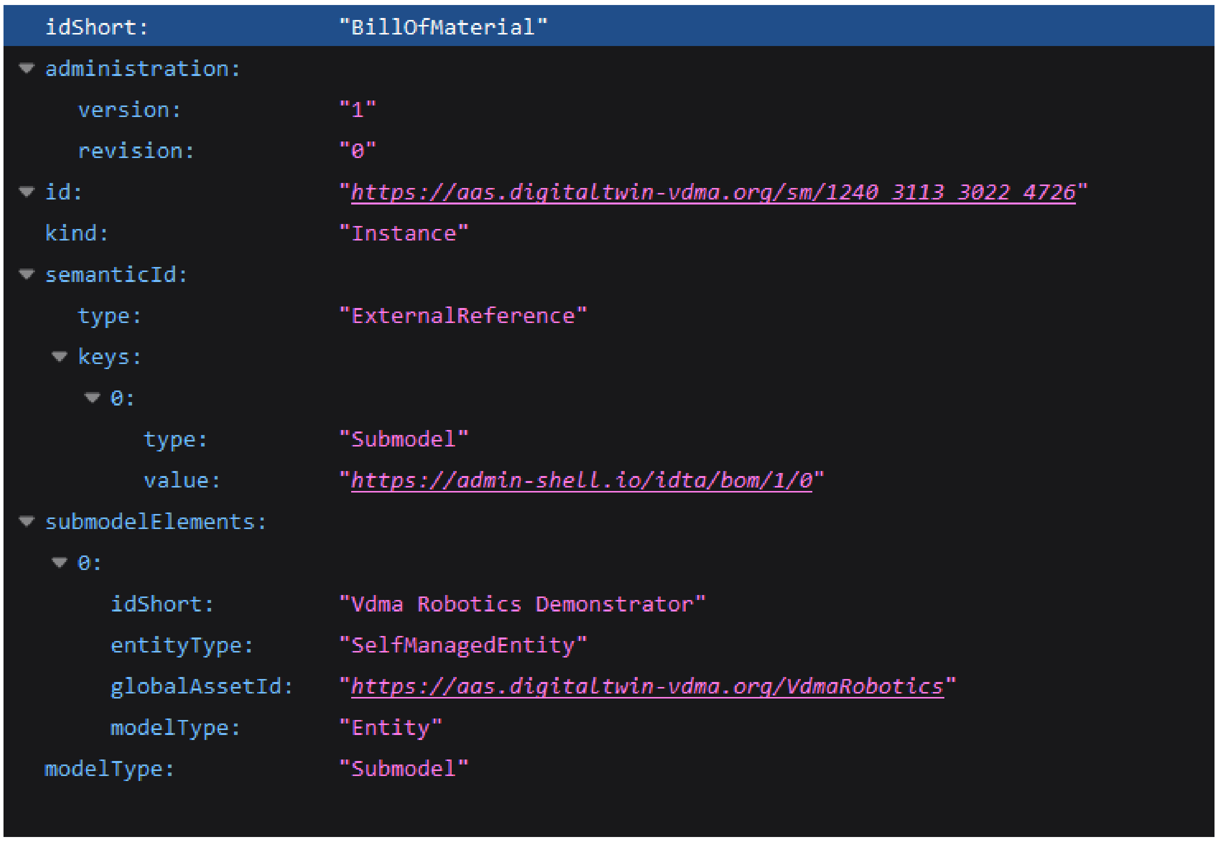
```
https://v3.admin-shell-io.com//submodels/aHR0cHM6Ly9hYXMuZGlnaXRhbHR3aW4tdmRtYS5vcmcvc20vMTI0MF8zMTEzXzMwMjJfNDcyNg==
```

The passed value is the already displayed submodelId value from the HTML.

```
async function getSubmodel(modelID)
```

```
<div id={`submodelElements-${value}`} onLoad={getSubmodel(value)}></div>
```

With a fetch the data is retrieved:



```
idShort:               "BillOfMaterial"
▼ administration:
    version:           "1"
    revision:          "0"
▼ id:                  "https://aas.digitaltwin-vdma.org/sm/1240_3113_3022_4726"
  kind:                "Instance"
▼ semanticId:
    type:              "ExternalReference"
  ▼ keys:
    ▼ 0:
        type:          "Submodel"
        value:         "https://admin-shell.io/idta/bom/1/0"
▼ submodelElements:
  ▼ 0:
      idShort:         "Vdma Robotics Demonstrator"
      entityType:      "SelfManagedEntity"
      globalAssetId:   "https://aas.digitaltwin-vdma.org/VdmaRobotics"
      modelType:       "Entity"
  modelType:           "Submodel"
```

After that the HTML paragraph for the submodel is located and filled with the fitting idShort of the submodel.

```
document.getElementById("idShort-"+assetID).innerHTML = bestimmtesAssetJson.idShort;
```

If the idShort is a Bill of Material, for each submodel element a paragraph is created and filled with the elements idShort.

```
if (bestimmtesAssetJson.idShort === "BillOfMaterial" || bestimmtesAssetJson.idShort === "BoM" || bestimmtesAssetJson.idShort === "BillofMaterial"){
//hier aus dem Submodel das submodel Array Elements gezogen
let submodelElementsList= bestimmtesAssetJson.submodelElements

console.log("Submodel Elements")
console.log(submodelElementsList)

let submodelContainer = document.getElementById("submodelElements-" + assetID);
submodelContainer.innerHTML = "";

for(let i= 0; i<submodelElementsList.length; i++){
    let pElement= document.createElement('p')

    pElement.innerHTML=submodelElementsList[i].idShort

    document.getElementById("submodelElements-"+assetID).appendChild(pElement)
}
```

7 Module Tests

Testing for this module is conducted through usability tests. In this process, a server and an AAS object are selected. Furthermore the “Assetsubmodels” button is pressed. The expected outcome is that the data from the selected server reaches the web client without causing a crash and then checks and displays the submodule information.