Customer: Rentschler & Holder Rotebühlplatz 41 70178 Stuttgart

Supplier: Team 2 (Paul Brenner, Jonas Alexander Graubner, Mohaddeseh Tibashi, Selvana Dwi Ayunda, Luka Dominik Pavic) Rotebühlplatz 41 70178 Stuttgart

Version History

Version	Date	Author	Comment
0.1	02.05.2023	Jonas Graubner	Created 1-3
0.1	08.05.2023	Jonas Graubner	Created 4-6
1.0	12.05.2023	Jonas Graubner	Finalization

Table of content

1. Scope

The AASX Server used in this Project is based on the AASX Server from the IDTA IDTA Github.

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 has no view for this module, since this module provides the API. Nonetheless there is a simple Backend visualization found at the root path of the Webserver (example: http://localhost) it is not ment for

enduser use it'S just ment for developers. If the AASX Server runs into an error he will always respond with an semy user-readable error.

3.2. Requirements

This Module provides the REST-API for the frontend hereby it fulfills the following Requirements.

- AASM-REQ2 Identity & Access Management
- AASM-REQ3 AAS content data management
- AASM-REQ5 Rest-API Support
- AASM-REQ6 Error Display
- AASM-REQ7 MongoDB Integration
- AASM-NF20/ Reliability
- AASM-NF30 Performance

3.3. Module Context

This implemantation of an AASX Server is based on the AASX Server of the IDTA. It provides the Rest-API for the frontend (MOD3 & MOD4.)

4. Analysis

The primary Job of this Module is to convert all different AASX standardised formats in it's internal storage objects and back, which are stored in the MongoDB with the help of MOD01 and the local .aasx files.

5. Design

All Interfaces for this Module were already existing. The Module implements the following Interfaces:

- DotAAS Part 2 | HTTP/REST | Asset Administration Shell Repository (ASP.NET Core 3.1) - used for the json communication on the API
- .aasx file format to store the assets in file for easy exchange.
- Mongo DB Interface from MOD01

6. Implementation

To accomidate all requirements the Server was modified in multiple ways. First was added an extra Class which houses the MongoDB Interface (s. MOD01). In the second step all internal API Methods were rewritten to read from MongoDB instead of the internal Array. Lastly the security Check Method was enhanced so that it can accomidate a more widly type of request from the V3 standard.

Place in Code: Startup: SOURCE/Server/aasx-server/src/AasxServerStandardBib/Program.cs

File API: SOURCE/Server/aasx-server/src/IO.Swagger.V1RC03/Services/AasxFileServerInterfaceService.cs

AssetAdministrationShellEnvironmentAPI: SOURCE/Server/aasx-server/src/IO.Swagger.V1RC03/Services/AssetAdministrationShellEnvironmentService.cs

7. Module Tests