# Moduldocumentation

(MOD)

(TINF20C, SWE I Praxisprojekt 2021/2022)

## Modul

# Controller

*Project:* *Modelling Wizard for Devices*

*Customer:* *Rentschler & Holder*
Rotebühlplatz 41
70178 Stuttgart

*Supplier:* by Lukas Ernst – Team 1
(Linus Eickhoff, Florian Kellermann, Lukas Ernst, Malte Horst, Florian Kaiser)
Rotebühlplatz 41
70178 Stuttgart

| Version | Date | Author | Comment |
|---|---|---|---|
| V0.1 | 07.09.2021 | Lukas Ernst | Created |
| V0.2 | 27.04.2022 | Lukas Ernst | Filled with information |
| V0.3 | 28.04.2022 | Florian Keller-mann | Improved information |
| V0.4 | 03.05.2022 | Malte Horst | Checked |
| V1.0 | 06.05.2022 | Lukas Ernst | Improved design |

# Contents

# 1. Scope

This module documentation explains the Controller in more detail. It shows how the standalone application is working in the backend for example how files are imported/exported and which features are implemented. The individual functions are tested in advance and their results are documented here. If there are existing problems, they are also listed here and possible solutions are explained in more detail.

It can also serve as a programming guide, if further features should be implemented.

# 2. Definitions

**GUI -** Graphical User Interface
**SRS -** System Requirement Specification
**STP -** System Test Plan
**STR -** System Test Report
**AMLX -** AML Package
**CAEX -** Computer Aided Engineering Exchange
**URI -** Uniform Resource Identifier

# 3. Module Requirements

## 3.1. User View

This Module should provide the user the following features:

1. An opportunity to create a File
   - Add Role Classes
   - Add Interfaces
   - Add Attachments
2. Export the created File
3. Import other Files
4. Load new Libraries

## 3.2. Requirements

The following requirements are implemented by this module: /LF10/, /LF20/, /LF30/, /LF60/, /LF70/, /LF80/, /LD10/, LD/20/.

**/LF10/:** The user now has the possibility to import files using an absolute path. In the beginning this did not work anymore.

**/LF20/, /LF30/:** This requirement is about the AML Component Checker. The file is to be checked against the AMLX standards. It is checked whether the structure of the file fits and whether all libraries are present.

**/LF60/:** When the user is shown the attributes of a loaded device, he can edit any attribute he wants to change.

**/LF70/:** When starting the application, the user can create a new, empty device model.

**/LF80/:** This requirement is about exporting the file in the correct CAEX version. In the previous project this was version 3.0. Now you can choose between CAEX 2.15 and 3.0. CAEX is a neutral data format for storing hierarchical object information. The versions differ in their standards. This requirement has been implemented.

## 3.3. Module Context

This module provides the backend of the Standalone application. It is responsible for ensuring that the data in the graphical user interface is all correct and also that all the required data is in place. The data entered is then processed by the controller and put into the correct structure. If any information for creating an AMLX file is missing, the user will be informed about it via a specific error message. Images are attached as an external file in the AMLX container. Interfaces, role class and all associated libraries are stored in the root-aml file. Afterwards the file can be exported. The controller is also responsible for allowing the import of AMLX files. If this is not possible the user will be informed.

# 4. Analysis

The main task of the controller is the import and export of an AMLX file. The data for this is transmitted from the graphical user interface to the controller. The controller must then process the data and then create a root-aml file. When exporting, the following things must be taken into consideration: The libraries must be exported, all data must be exported, so that there is no data loss and the data must also be validated. When importing, it is important to be able to map the majority of all files. It is important that the libraries of a file are recognized and that it is included in the correct tab so that it can be processed correctly. In addition, there must be no loss of data, which means that images and all interfaces must also be imported correctly. If there are problems with the export, the user must be informed. This is usually done via an error message from the user interface.

# 5. Design

There are three important functions of the module: Importing a file, exporting a file and validating the data from the graphical user interface.

Validate data: The Validate Data must verify that all the data entered is correct. Thus, fields that are mandatory must logically be filled in. If they are not, a corresponding error message must be issued. In addition, it must be checked whether fields also have the correct data types. For example, a URI must have a certain syntax to be considered valid. Attributes that are not present, i.e. that were additionally entered by the user, may not be transferred, because the libraries are responsible for providing the attributes to the RoleClasses / Interfaces. If all data is correct and complete, the file can be exported.

Export: When exporting, you can choose between CAEX 2.15 an CAEX 3.0. Attachments, such as ComponentIcons and ManufacturerIcons are then added to the container with reference to the image. Once the export is complete, a confirmation message will appear.

Import: The import loads the file with the interfaces and libraries. The file is scanned for data and how to classify it. For example, the BaseClass is loaded into the GenericData tab, the interfaces into the Interfaces tab, and the attachments into the Attachments tab. Again, the data is checked to see if it conforms to the standards. If there are complications during the import, an error message is displayed.

## 5.1. Risks

The export is based on the libraries of CAEX. This decides the standard of the file and its structure. If the CAEX version is no longer maintained and gets outdated, there could be complications with the export. This danger is currently still quite small, since the newest standard is the version 3.0 and it was also task to implement this. However, if there are problems with importing CAEX 3.0 files in future the standalone application would be unusable.

Libraries are very important for import and export. If these are not available, it can happen that files are not loaded correctly, or they are even no longer usable. In addition, it can happen that devices cannot be created because certain libraries are missing.

# 6. Implementation

The controller module contains a lot of different functions that allow to execute certain actions on triggered events from the graphical user interface. However, these are not that important, so they will not be discussed in detail. The most important functions are those that represent the main functions.

**Data validation:** “saveToolStripMenuItem_Click()” the most important data validation function is the saveToolStripMenuItem_Click() function. This validates the data and checks if all necessary data is present. If they are not present, a corresponding error message is output. If all data is correct, the data is passed to the export function.

**Export:** “CreateDevice()” this function is responsible for converting the data into the correct data structure. In this function also for example URIs are converted into the correct data formats and images are attached into the container. The export file starts first of all to read the libraries from the data and to add them to the file. Then the generic data is inserted into the file. The name of the file is also taken from the generic data (Device Name & Vendor Name). Afterwards all interfaces, which were integrated, are attached to the file. If necessary, further libraries are included in the file. Finally, the other attachments are appended. When the file is finished, it is saved in the specified path as an .amlx file.

**Import:** “openToolStripMenuItem_Click()” this function is responsible for importing files. First of all, it checks if the file has the correct structure or if any data is corrupt. If this is the case, a corresponding error message is returned. If the data is in the correct structure, the information is read from the file. These are arranged directly by the function into the three tabs. This means that libraries are loaded and can be seen on the right in the graphical interface, the base data in the generic data tab, the interfaces in the interfaces tab and the attachments in the attachments tab.

# 7. Module Test

In this section nearly all requirements will be tested separately on their functionality.

## 7.1. Module Testplan

| Req. - ID | Functionality |
|---|---|
| LF10: Import | Imports file by absolute path |
| LF20: File validation | Checks whether input file is in a valid format |
| LF30: Error handling | Application throws errors on expected shutdowns and wrong formatting |
| LF60: Edit device | Every attribute of devices should be editable |
| LF70: Create device | Creates a new and empty device |
| LF80: Export device | Loaded device is saved as to file |

## 7.2. Module Testreport

| Req. - ID | Pass/ Fail | When failed: Observation | |
|---|---|---|---|
| LF10: Import | Pass | | Linus Eickhoff |
| LF20: File validation | Pass | | Linus Eickhoff |
| LF30: Error handling | Pass | | Linus Eickhoff |
| LF60: Edit device | Pass | | Malte Horst |
| LF70: Create device | Pass | | Linus Eickhoff |
| LF80: Export device | Fail | File is saved and exported correctly without errors when creating a new de-vice. While editing an ex-isting device, exporting fails. | Florian Kaiser |

# 8. Summary

Most of the module requirements have been implemented successfully Both CAEX versions were implemented, so it is possible to export the AMLX in this versions. Many bugs were found here as well, which were also fixed. However, not all requirements were implemented, because the bug was not found despite debugging for days. Also the import of foreign AMLX files, which have a different structure than the export structure of this standalone application, works to some extent. Further enhancements were also implemented in this module to improve the usability of the program.

# 9. Appendix

## 9.1. References

[1] System Requirements Specification: https://github.com/H4CK3R-01/TINF20C_ModellingWizard_Devices/wiki/1.-Software-Requirements--Specification
[2] Previous Project: https://github.com/DekaAthlos/TINF19C-ModellingWizard
[3] System Test Plan: https://github.com/H4CK3R-01/TINF20C_ModellingWizard_Devices/blob/47d2ba67fc73ebc080f303f0e29ca2260d8c7d88/PROJECT/STP/TINF20C_STP_Team_1.pdf
[4] System Test Report: https://github.com/H4CK3R-01/TINF20C_ModellingWizard_Devices/blob/47d2ba67fc73ebc080f303f0e29ca2260d8c7d88/PROJECT/STR/TINF20C_STR_Team_1.pdf

## 9.2. Code

The source code for this module can be found at:

- https://github.com/H4CK3R-01/TINF20C_ModellingWizard_Devices/blob/app-source-code/SOURCE/Plugin/MWData.cs
- https://github.com/H4CK3R-01/TINF20C_ModellingWizard_Devices/blob/app-source-code/SOURCE/Plugin/DeviceDescription.cs