System Requirements Specification

(Pflichtenheft)

(TINF18C, SWE I Praxisprojekt 2019/2020)

Project: *DD2AML Converter*

Customer: Rentschler & Ewertz
Rotebühlplatz 41
70178 Stuttgart

Supplier: by Lara Mack - Team 3
(Nora Baitinger, Antonia Wermerskirch, Carl Beese, Lara Mack, Bastiane Storz)
Rotebühlplatz 41
70178 Stuttgart

Version	Date	Author	Comment
0.1	07.09.2018		created
0.2	01.11.2019	Wermerskirch	added Introduction
0.3	02.11.2019	Mack	added Product Data
0.4	04.11.2019	Baitinger	added Product Requirements
0.5	04.11.2019	Mack	added Non-Functional Requirements
0.6	04.11.2019	Wermerskirch	added Product Environment and Use Cases

CONTENTS

1. Introduction

The goal of this project is to develop software that supports the conversion of a Fieldbus-specific device description file such as IODD, CSP+, and ESI to an AutomationML/CAEX V3 file. For this purpose, an already created conversion tool Names GSD2AML will be extended with the three mentioned conversion rules. The main function of the software is to convert one file into another. For this to be possible, the converter must automatically recognize the respective input format so that it can be converted according to the conversion rules. The software should include a library module that allows the software to be used in other projects to perform the conversions, a command-line program that directly converts the file, and a graphical user interface for better usability that has the same features but is more user-friendly.

1.1. Product Environment

The software is mainly used by manufacturers and other companies who need to manage various device description files. The device description files of the types GSD, IODD, ESI and CSP+ are supported, whereby an independent program only for the conversion of GSD files already exists, which is now extended by the transformation rules mentioned above.

An IODD (IO Device Description) file describes the sensors and actuators of a system or components. It also contains information about identity, parameters, process data, communication and more. It is written in XML format, like AML, which ensures a conversion.

The ESI (EtherCAT Slave Information) file is also written in XML. It is used in networking hardware to specify the information which is necessary for the use of the EtherCAT protocol.

CSP+ (Control & Communication Profile) is used to describe components of the CC-Link Family or those which are compatible with them.

When working with different device description files, it may be useful to normalize them to make it easier to continue working with them. In this case, the files could be converted to AML, an exchange format developed as an open standard. During conversion, all important data and information are read from the device description file and stored in AML.

AutomationML (AML) is short for Automation MarkUp Language and is used to describe parts of automation plants as objects. These objects can consist of multiple other objects and be part of a larger assembly of objects. That way AML can be used to describe a single screw or an entire robot with the necessary level of detail. AML makes use of various standards to describe plant components.

1. CAEX (Computer-Aided Engineering Exchange) to describe attributes of objects and their relations in a hierarchical structure. This is called a system topology. In this respect, CAEX forms the overarching integration framework of AutomationML.
2. COLLADA to describe the geometry and 3D models of objects
3. COLLADA also integrates motion planning. It describes the connections and relations of moveable objects, which is called Kinematics
4. PLCopen XML describes the logic. Internal behaviour and states if objects, action-sequences and I/O connections are implemented via this format.

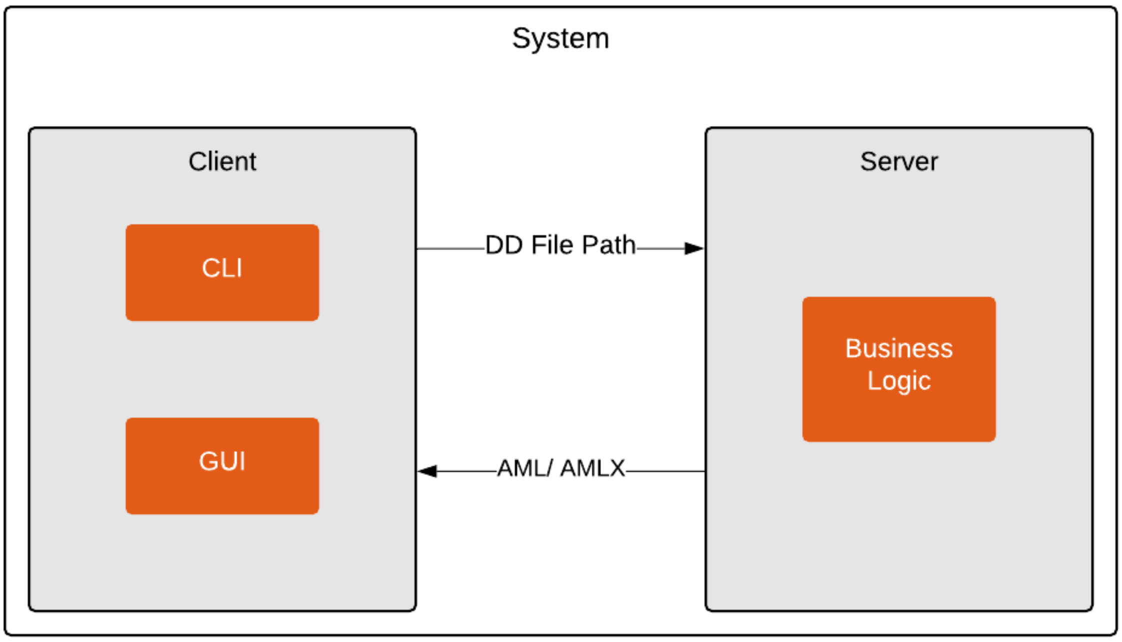


Figure 1: Product Environment

1.2. Use Cases

The software includes a library that can be integrated into other software, a command line interface and a graphical user interface, whereby the library and the command line interface are two independent programs.

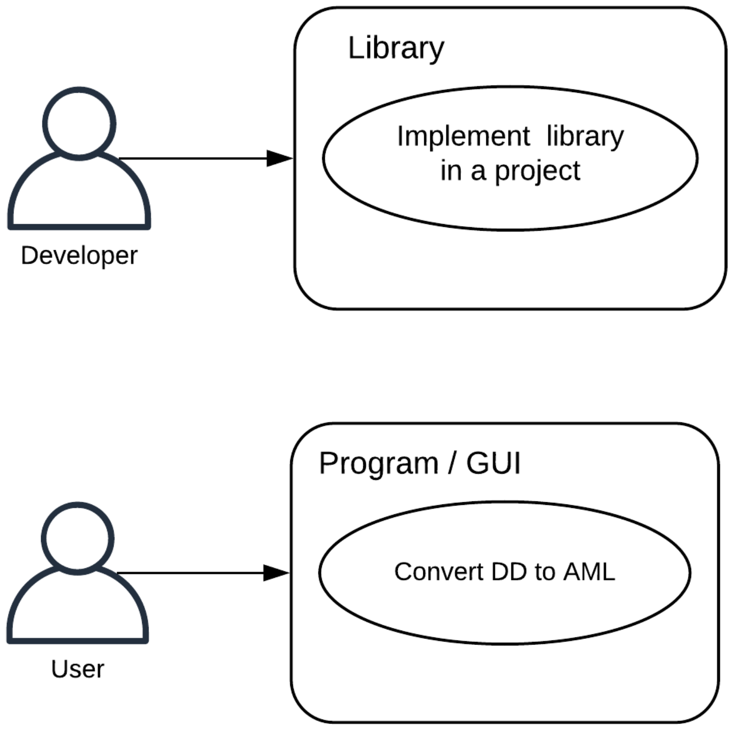
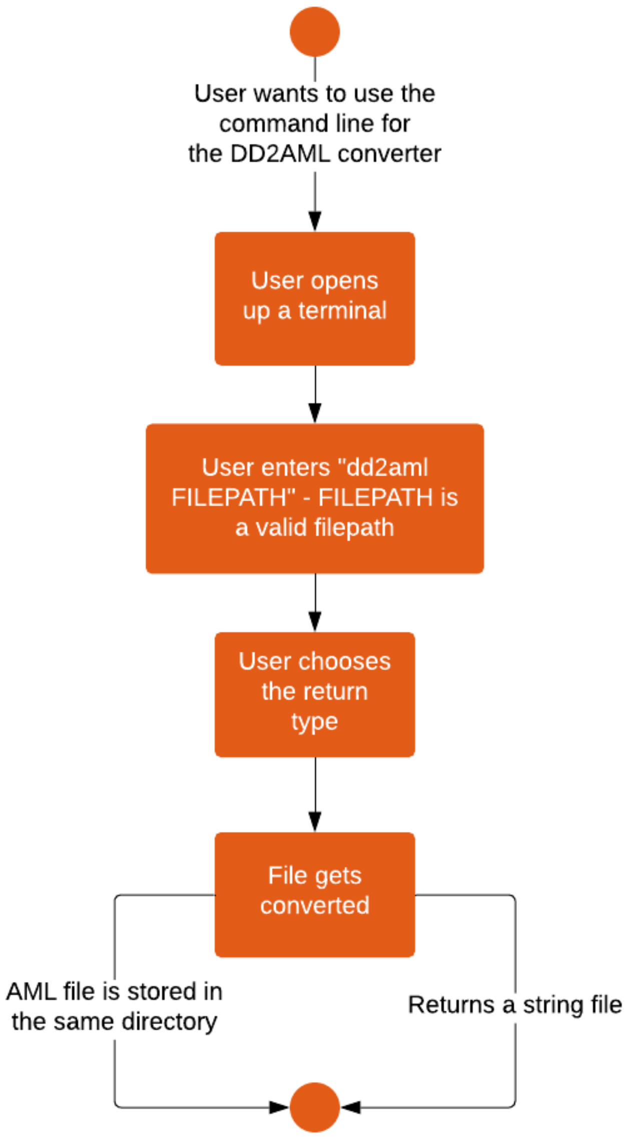


Figure 1.2: Use Case Overview Diagram

1.2.1. <UC.001> File conversion command line

Related Business Process:	<BP.001>: File conversion
Use Cases Objective:	User wants to convert his DD file to an AMLX file using the command line.
System Boundary:	Program itself is the system boundary.
Precondition:	• The program must be installed on the device. • "dd2aml" needs to be registered as an environment variable • The DD file must be without errors.
Postcondition on success:	The command line should not be closed before the conversion is finished.
Involved roles:	User and Converter
Triggering Event:	Typing dd2aml file with a valid FILEPATH in the command line

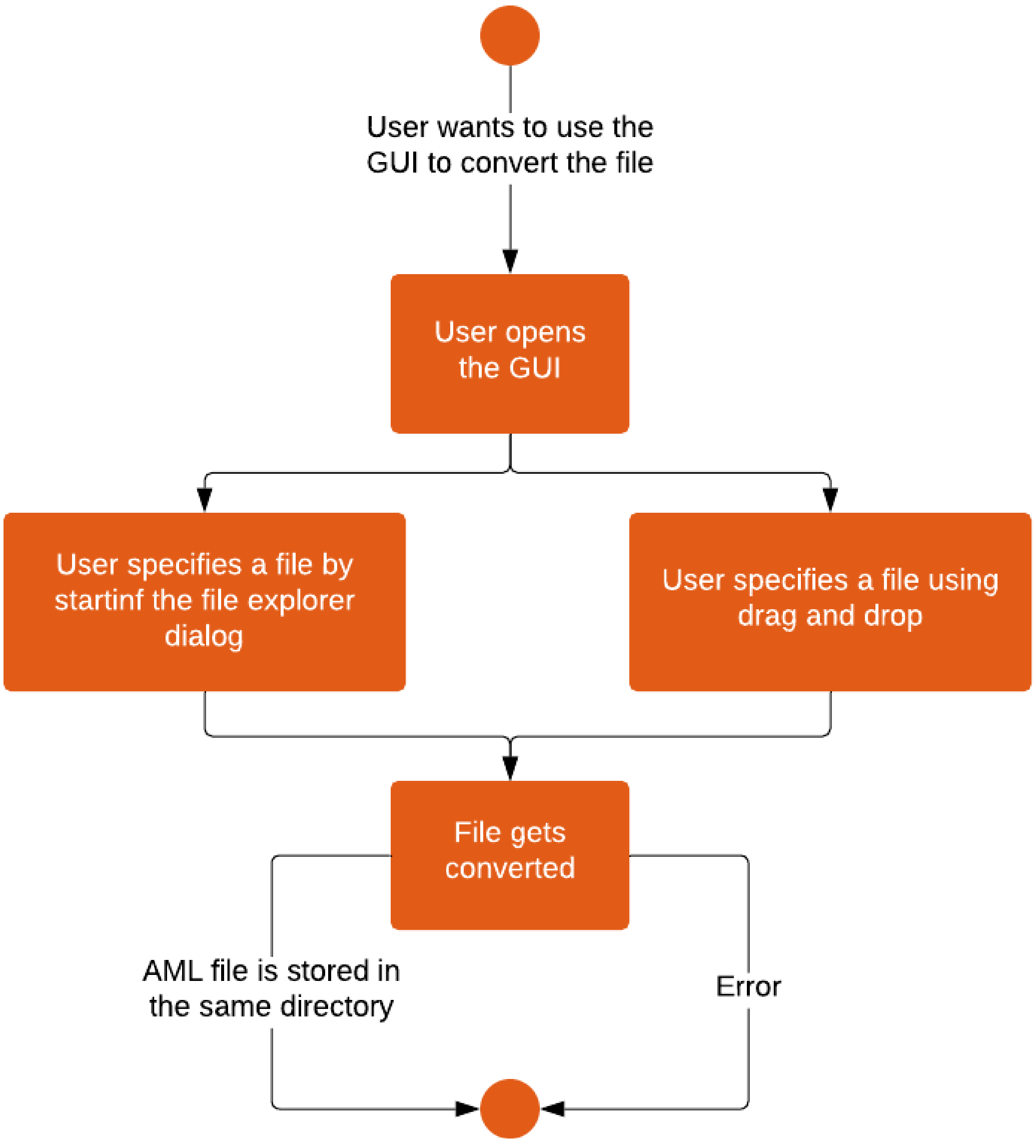


User wants to use the command line for the DD2AML converter
User opens up a terminal
User enters "dd2aml FILEPATH" - FILEPATH is a valid filepath
User chooses the return type
File gets converted
AML file is stored in the same directory
Returns a string file

Figure 1.2.1: <UC.001> File conversion command line

1.2.2. <UC.002> File conversion GUI

Related Business Process:	<BP.001>: File conversion
Use Cases Objective:	User wants to convert his DD file to an AMLX file using the GUI.
System Boundary:	Program itself is the system boundary.
Precondition:	• The program must be installed on the device. • The DD file must be without errors.
Postcondition on success:	The program should not be closed before the conversion is finished.
Involved roles:	User and Converter
Triggering Event:	Program is started.



User wants to use the GUI to convert the file
User opens the GUI
User specifies a file by startinf the file explorer dialog
User specifies a file using drag and drop
File gets converted
AML file is stored in the same directory
Error

Figure 1.2.2: <UC.002> File conversion GUI

1.2.3. <UC.003> Library usage

Related Business Process:	<BP.002>: Library implementation
Use Cases Objective:	Possibility for developer to implement this project in their software.
System Boundary:	Program itself is the system boundary.
Precondition:	The library needs to have a clearly documented interface and must be downloadable from GitHub or via NuGet. It needs to be published under the AGPL-license.
Postcondition on success:	The DD file was successfully converted into an AMLX package.
Involved roles:	Developer and library
Triggering Event:	Developer wants to implement a conversion function to his project.

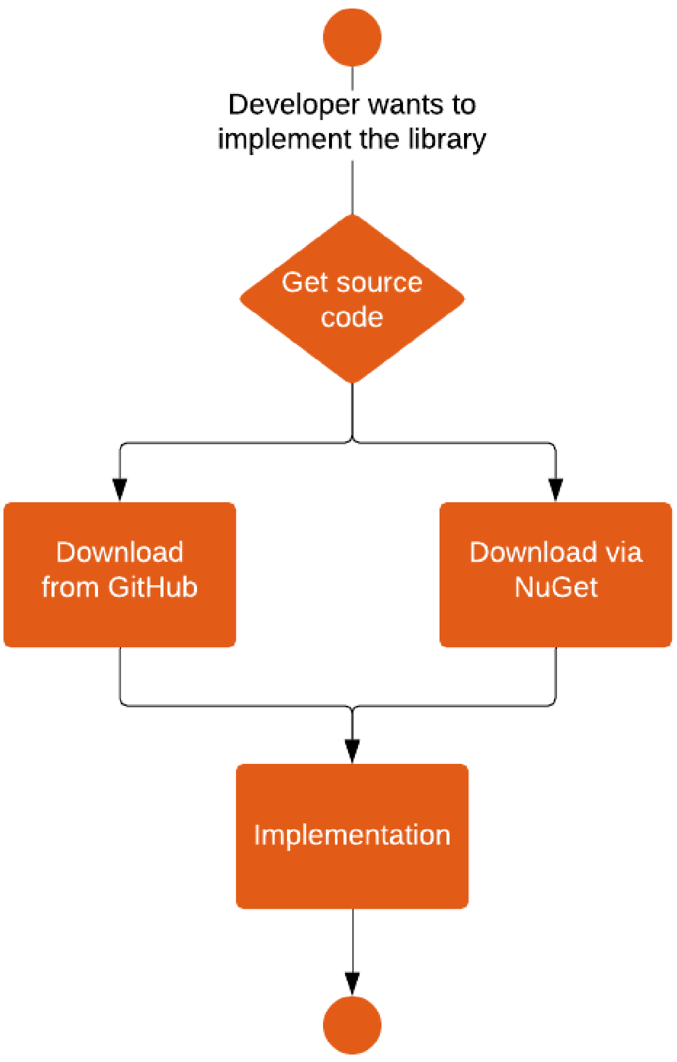


Figure 1.2.3: <UC.003> Library usage

2. Product Requirements

The following functionalities shall be supported by the system.

2.1. /LF10/File import

Before starting the conversion to AML, it is necessary to import the file the user wants to convert. Therefore, the user selects an IODD, ESI or CSP+ file. The component shall check if it is one of the listed formats and recognize it as one of the given formats. In case of violation, the user is asked to import another file. Otherwise, it will continue with the "Input validation".

Input field	**Value Range**
filepath	*absolute location of the import file*

2.2. /LF20/Input validation

The „Input validation" shall check if the import file is serializable. In case of a successful serialization, the import file is considered valid and the file shall be checked for essential data in regard to the conversion. If such data is missing an error shall be reported. Otherwise, the file will be accepted.

Input field	**Value Range**
filepath	absolute location of the import file

2.3. /LF30/Conversion

The main feature is to convert IODD, ESI and CSP+ files into AML-files. Therefore, the Conversion shall check the certain features of each file format and convert them into the AML notation. The conversion results are either returned as a string or stored with its dependencies in an AMLX package. This procedure is described in the component "Compressor".

Input field	**Value Range**
filepath	absolute location of the import file
filetype	the file is either IODD, ESI or CSP+
outputfilepath (optional)	location where the resulting AMLX package shall be stored
filename (optional)	filename of resulting AMLX package

2.4. /LF40/Conversion

The Compressor shall collect all files and resources and create the required AMLX package. The AMLX package will contain the generated AML file and all the corresponding files. The name and the output directory of the AMLX package are by default the same as those of the input file.

Input field	Value Range
filepath	absolute location of the import file
outputfilepath (optional)	location where the resulting AMLX package shall be stored
filename (optional)	filename of resulting AMLX package

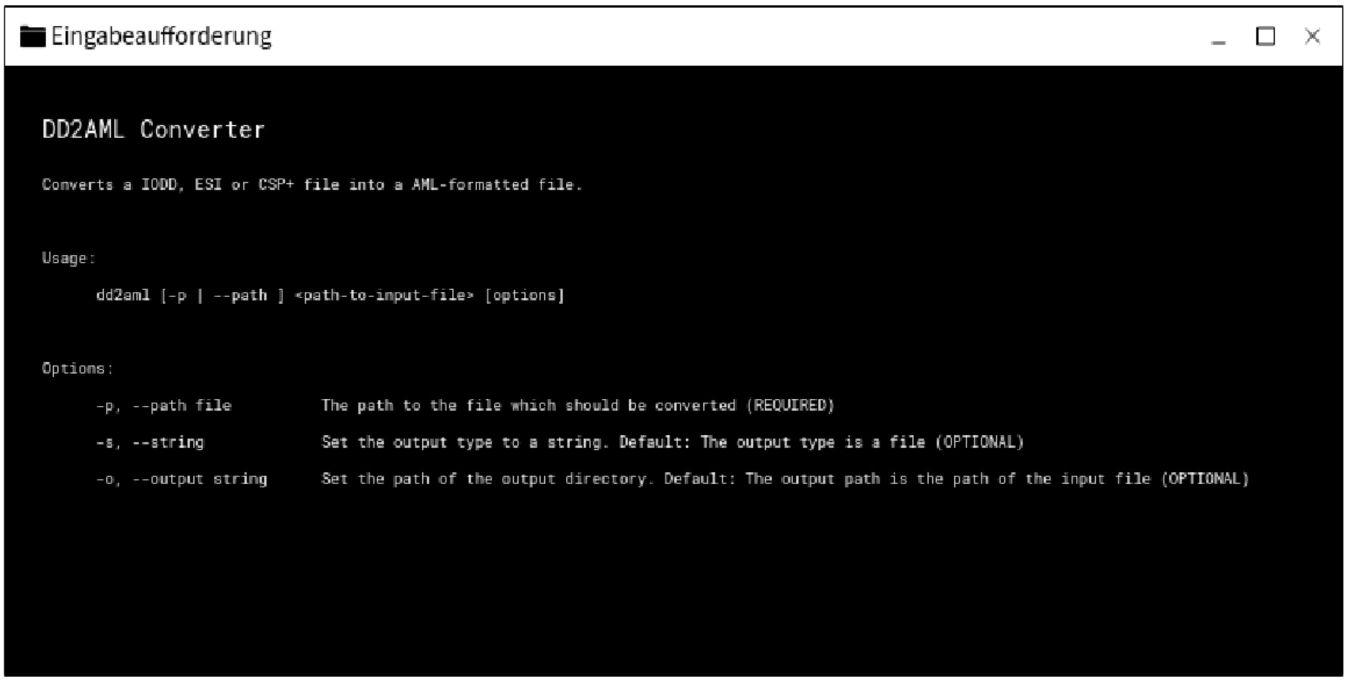
2.5. /LF50/File conversion library

The library shall provide an API for the file conversion. After acceptance, the conversion shall be performed. The resulting AML file will be either returned as a string or stored with its dependencies in an AMLX package. For this, the library uses the “File import” and the “Conversion” component and eventually the “Compressor” to create the AMLX package.

Input field	Value Range
filepath	absolute location of the import file

2.6. /LF60/File conversion command line

The user can start the conversion with the command-line interface to interact with the program. The conversion will use the API of the library. The command-line tool shall provide the requests to the user for the needed inputs and provide error information to the user. The resulting AML file will be either returned as a string or stored with its dependencies in the AMLX package.

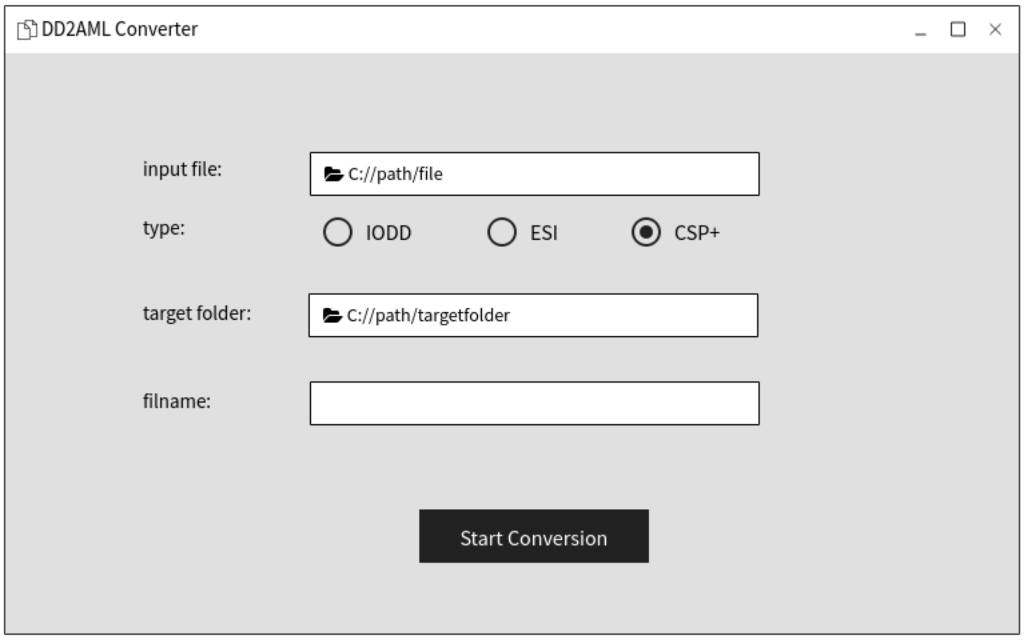


```
DD2AML Converter

Converts a IODD, ESI or CSP+ file into a AML-formatted file.

Usage:
      dd2aml [-p | --path ] <path-to-input-file> [options]

Options:
      -p, --path file          The path to the file which should be converted (REQUIRED)
      -s, --string             Set the output type to a string. Default: The output type is a file (OPTIONAL)
      -o, --output string      Set the path of the output directory. Default: The output path is the path of the input file (OPTIONAL)
```

Figure 2.6: command line Interface

2.7. /LF70/File conversion GUI

The second option for the user to interact with the program is the GUI tool. It also uses the API of the library for the conversion. It shall provide the user with the necessary requests to start the conversion and give error information when needed.



input file: C://path/file

type: IODD ESI [✓ CSP+]

target folder: C://path/targetfolder

filname:

Figure 2.7: GUI

3. Product Data

This section describes the initial Logical Data Model from the previously described functionalities and the context level diagram. The data requirements describe the (business) data needed by the application system via its interfaces.

3.1 /LD10/AML-DD

The system shall create an AML root file with versioning header and a "SystemUnitClass" with the descriptions, identification and interface information of the Profinet Device, also an external Reference to the original file.

3.2 /LD20/AMLX Package

The system shall store an AMLX file containing the AML root file, the original file and the device picture file.

3.3 /LD30/File types

The system shall work with a valid IODD, ESI or CSP+ file.

4. Non-Functional Requirements

This section describes the non-functional requirements for the product.

4.1 /NF10/System Enviroment

The software requires Windows 7 or higher. It also requires the Framework to be installed in Version x or higher. The framework is automatically included in the required Windows version.

4.2 /NF20/Portable Program

The software shall exist as a portable program. This means, that there is no installation required. This can be achieved by packing the necessary files and the runnable program in a ZIP file.

4.3 /NF30/Installation wizard

The software shall be installed using an installation wizard (File Conversion GUI).

4.4 /NF40/License

The software shall be published under the AGPL v3.0 license.[1]

5. References

[1] Free Software Foundation: GNU AFFERO GENERAL PUBLIC LICENSE Version 3, 19 November 2007. retrieved 05.11.2019

6. Glossar

AML Automation Markup Language is an open standard data format for storing and exchanging plant planning data.

GUI Graphical User Interface