Moduldokumentation

(MOD)

(TINF18C, SWE I Praxisprojekt 2019/2020)

Modul

DD2AML.Cli

Project: *DD2AML-Converter*

Customer: *Rentschler & Ewertz*
Rotebühlplatz 41
70178 Stuttgart

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

Version	Date	Author	Comment
0.1	03.04.2020	Lara Mack	First draft
0.2	17.04.2020	Lara Mack	Scope, Modul Requirements
0.3	24.04.2020	Lara Mack	Added Analysis, Design and Requirements
0.4	09.05.2020	Lara Mack	Added tests
1.0	09.05.2020	Lara Mack	Final version

1 Content

2 Scope

The Module Documentation (MOD) describes the architecture, the interfaces and the main features of the module. It also describes the module/component test including the results. It can also serve as a programming or integration manual for the module. If there are some risks related to the module itself, they shall be noted and commented within this document.

3 Definitions

AML	Automation Markup Language
AMLX	Automation Markup Language Package
CLI	Command Line Interface
GSDML	General Station Description Markup Language
IODD	IO Device Description

4 Figures

5 Module Requirements

5.1. User View

From the user's point of view, the module should make it possible to convert a GSD, IODD or CSP+ file into an AML file. This is done by passing the path of the source file with a command line flag. Optionally, the user can specify the output directory of the converted AMLX file. It is also possible for the CLI to output the contents of the AML file in the command line.

5.2. Requirements

The module has the task of fulfilling the requirements under /LF60/ in the SRS.
Following features are part of this requirement:

1. The CLI must be able to create an AML file in an AMLX package or output the contents of the AML file on the command line.
2. The user shall be able to specify the output directory via a -o (--output) flag. If an explicit output directory is omitted, the output directory shall be the directory of the GSDML file.
3. The user must be able to decide what the name of the resulting AMLX package is. This is done using -o (--output) flags, where a filename plus extension can be specified at the end of the file path.
4. The CLI shall provide input validation.
5. The CLI shall provide insightful error/help messages to the user.

5.3. Module Context

This module will use the DD2AML conversion library (DD2AML.Lib) to convert the DD file into an AML file. The DD2AML Conversion Library consists of the following modules:

- Compressor
- Converter
- Logger
- Parser
-

The CLI implements the logger module to receive error messages from DD2AML.Lib.
The Logger Module is implemented by passing a Logger object to the library, so the error message can be displayed correctly. It communicates with the converter by passing the path to the DD file and optionally the output path to the library via the provided interface of DD2AML.Lib in the form of public methods.

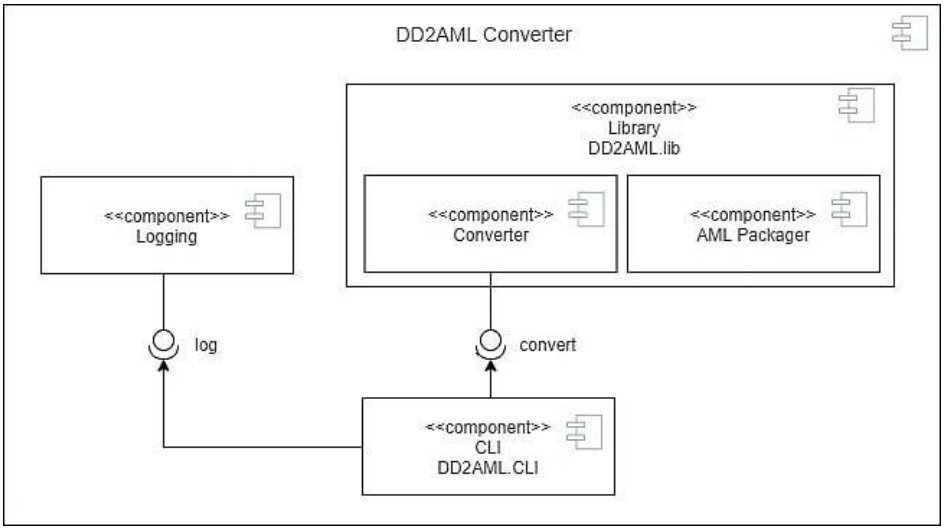

Figure 1 Component diagram

6 Analysis

The main task of DD2AML.Lib is to convert a DD-formatted file into an AMLX file. Here an interface for the function of the conversion is provided. The user can use this with a suitable interface.

The module DD2AML.Cli provides a suitable interface. This module is used from a command line by connecting to the DD2AML.Lib interface. For this to work, the CLI must first validate the input. This is necessary to verify that the data provided by the user is valid.

A division into several phases is useful to achieve this.

1. Syntactical Analysis
Analysis whether the formal use of the CLI flags is correct.

2. validation of the input
This checks whether the specified DD-formatted file and the specified output directory exist. If the output directory does not exist, it should be created. The system then checks whether the user has specified an output name for the AMLX package. If this file already exists, the user is asked whether the file should be overwritten.

3. triggering the conversion
Start the conversion using the DD2AML.Lib module

7 Design

The module is split into multiple classes to handle this task. Each of them has their own responsibility.

Class: Program

The main class is located at ./SOURCE/src/DDAML.Cli/Program.cs. This is the entering point for the application.

Main

The main method starts the parsing process for the command line arguments and triggers the conversion.

For this purpose, a Settings object is created which contains the parsed command line arguments. Through this object a trigger object is then used to trigger the conversion process.

Class: Settings

The purpose of the Settings class is the parsing and storage of the command line arguments. The source file is located at ./SOURCE/src/DD2AML.Cli/Settings.cs.

CheckCliArguments

This method checks the use of CLI flags for formal correctness. This is done by first iterating over the command line arguments and checking whether the short and long versions of a CLI flag have been used simultaneously. If this is the case, an error message is displayed. Secondly, the uniqueness of each flag used is confirmed. In case this cannot be confirmed, the method "PrintMultipleParameterError" is called to generate an error message. Finally, the simultaneous use of the flags -o (--output) and -s (--string) is checked, which is explicitly not allowed. It also checks which CAEX version should be used.

ParseCliArguments

This method will map the command line arguments into properties of the settings class.

CheckGsdmlExistence

This method will check whether the specified DD file exists and show an error message if the file cannot be found.

PrintMultipleArgumentError

This method will find CLI flags, that were used multiple times, and display an error message.

Class: Trigger

The main responsibility of this class is to prepare and start the conversion process. The source file is located at ./SOURCE/src/DD2AML.Cli/Trigger.cs.

Convert

This method will trigger the conversion process by using the Convert method of the DD2AML library.

If the output of the conversion will be an AMLX package, and not just a string, the output path will be verified, and the output directories will be created if they don’t exist.

GetOutputFile

This method determines the final output directory and file name of the AMLX package. There are three main branches in this function:

1. the user can specify the output using the -o (--output) flag, and the output flags end in the form of "string.string".
In this case the output path can be returned unchanged.

2. the user specified the output using the -o (--output) flag, but the output flags do not end in the form of "string.string".
The file name is [timestamp].amlx and will be appended to the output string specified by the user.

3. the user has not specified the output.

The directory of the DD file is used as output directory. The file name is [timestamp].amlx.

CheckOutput

This method will check whether the output file already exists on the file system. The user will be asked whether to overwrite the AMLX package if it already exists. The conversion will be aborted if the file should not be overwritten.

Class: Util

This class holds utility function, that are used by multiple other classes.

PrintHelpText

This method will simply print the predefined help text to the command line and end the process.

Class: Logger

This class is responsible for logging messages. It will implement the logger interface, which is provided by the DD2AML.Lib module.

7.1. Risks

With the CLI module, there is a risk that the user's input is not sufficiently checked and therefore no corresponding error messages are issued. Although this is covered by corresponding commands and an attempt is made to handle each possible case individually, there is a small change that a case is not handled sufficiently or not at all.

To minimize this risk, the test team performs manual tests against the CLI.

8 Implementation

The implementation followed the design described in chapter 7. Tests were repeatedly carried out against DD2AML.Cli to ensure that the program behaved correctly.

The .NET System.IO library was used for file system and command line interaction.

The logger module DD2AML.Lib was used for logging.

The code was documented with C# XML documentation comments.

Program

```
private static void Main(string[] args)
```

- args are the arguments passed to the program

This is the entry point for the application.

Settings

```
internal Settings() { }
```

Empty constructor for testing purposes.

```
public Settings(IList<string> args)
```

- args are the arguments passed to the program

Used to create a Settings object with an argument list.

```
internal void CheckCliArguments()
```

Checks if the user passed:

1. the same argument multiple times
2. the short and the long version of an argument
3. the output and the string flag

```
internal void ParseCliArguments()
```

Parses the CLI arguments and maps them to Settings properties.

```
private void CheckGsdmlExistence()
```

Checks whether the specified GSDML file exists.

internal void PrintMultipleArgumentError()

Prints an error message if the same argument is used multiple times.

Trigger

public Trigger(Settings settings)

- settings is a Settings object with the CLI arguments

Used to create a Trigger object with an Settings object.

public void Convert()

Trigger the Convert() method from the GSD2AML.Lib module based on the specified output type.

internal void GetOutputFile()

Checks if the output file is valid. If not, it changes the string:

1. outputFile is empty directory of input file + timestamp.amlx
2. outputFile not ending in filename.amlx outputFile string + timestamp.amlx (e.g. …/test …/test/timestamp.amlx)

Util

public static void PrintHelpText()

Prints the help text and exits the program.

Logger

public void Log(LogLevel level, string message)

- level is the log level
- message is the message to log

Logs the message according to the log level.

9 Module Test

9.1. Component Testplan

Test-ID	Feature ID	Test Specification
TS-001	LF50	This test is done from a black-box-view. This test verifies that the Trigger.GetOutputFile() method works correctly.
TS-002	LF50	This test is done from a black-box-view. This test verifies that the Trigger.GetOutputFile() method works correctly with on output parameter.
TS-003	LF80	This test is done from a black-box-view. This test verifies that theSettings.PrintMultipleArgumentError() method correctly throws an error when given an invalid input.
TS-004	LF80	This test is done from a black-box-view. This test verifies that the Settings.PrintMultipleArgumentError() does not throw an error when the input is valid.
TS-005	LF80	This test is done from a black-box-view. This test verifies that the Settings.CheckCliArguments() method throws an error if the input is invalid.
TS-006	LF80	This test is done from a black-box-view. This test verifies that the Settings.CheckCliArguments() method does not throw an error if the input is valid
TS007	LF50	This test is done from a white-box-view. This test verifies that the Settings.ParseCliArguments() method correctly maps the input arguments into the Settings properties.

9.2. Component Testreport

Test-ID	Pass / Fail	It failed: Test Observation	Date	Tester
TS-001	PASS		09.05.2020	Bastiane Storz
TS-002	PASS		09.05.2020	Bastiane Storz
TS-003	PASS		09.05.2020	Bastiane Storz
TS-004	PASS		09.05.2020	Bastiane Storz
TS-005	PASS		09.05.2020	Bastiane Storz
TS-006	PASS		09.05.2020	Bastiane Storz
TS-007	PASS		09.05.2020	Bastiane Storz

10 Summary

The main task of this module is the validation of user input. Invalid user inputs are handled by displaying an error message. It tries to cover every possible invalid input.

It is divided into several classes, which have a clear and defined responsibility. This makes the module easily extendable, as well as the selection of the CAEX version.

11 Appendix

11.1. References

[1] SRS: https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/wiki/System-Requirements-Specification

[2] STP: https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/wiki/Systemtestplan

[3] STR: https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/wiki/Systemtestreport

11.2. Code

The unit tests can be found at:
https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/tree/master/SOURCE/src/Dd2Aml.Test

The source code can be found at:
https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/tree/master/SOURCE/src/Dd2Aml.CLI

11.3. Module Test Cases

11.3.1. <TC-002-001> (View CLI help text)

Testcase ID:	TC-001-002	
Testcase Name:	View CLI help text	
Req.-ID:	LF60	
Description:	The test case verifies that the converter displays all possible functions as help text as soon as the passed argument contains "--help". All other passed arguments are ignored.	
Test Steps		
Step	**Action**	**Expected result**
1	Install the DD2AML tool and open the CLI by typing cmd in the windows search.	The DD2AML tool is installed on the system. The CLI is open.

2	Run the DD2AML CLI with valid arguments and use the help flag.	Regardless of the other valid arguments, only "--help" is executed and a help text is displayed, showing all possible functions.

Testdata:	TD-002-002			
Dataset	**File**	**Validation**	**Permission Input**	**Permission Output**
1	Balluff-BNI_IOL_355_S02_Z013-20170315-IODD1.1.xml	valid	given	given
2	0x1099_BNI CIE-508-105-Z015_2.0_en.cspp	valid	given	given
3	GSDMLV2.33.xml	valid	given	given

11.3.2. <TC-002-002> (Converting without output flag)

Testcase ID:	TC-001-002
Testcase Name:	Converting without output flag
Req.-ID:	LF60
Description:	The test case verifies that a conversion is also possible without a given output path.

Test Steps		
Step	**Action**	**Expected result**
1	Install the DD2AML tool and open the CLI by typing cmd in the windows search.	The DD2AML tool is installed on the system. The CLI is open.
2	Run the DD2AML CLI with valid input flag, for example: dd2aml –input /filePathTo/Balluff-BNI_IOL_355_S02_Z013-20170315-IODD1.1.xml -v 2	The conversion is executed successfully. Since no output path is given, the output file is saved in the file path of the input file.

Testdata:	TD-002-002			
Dataset	**File**	**Validation**	**Permission Input**	**Permission Output**
1	Balluff-BNI_IOL_355_S02_Z013-20170315-IODD1.1.xml	valid	given	given
2	0x1099_BNI CIE-508-105-Z015_2.0_en.cspp	valid	given	given
3	GSDMLV2.33.xml	valid	given	given