**Modul documentation**

(MOD)

(TINF18C, SWE I Praxisprojekt 2019/2020)

Sub-Modul

**Logging**

Project: DD2AML Converter

Customer: Rentschler & Ewertz

Rotebühlplatz 41

70178 Stuttgart

Supplier: by Nora Baitinger and information from TINF17C/GSD2AML-Converter Team   
- Team 3 (Nora Baitinger, Antonia Wermerskirch, Lara Mack, Bastiane Storz)

Rotebühlplatz 41

70178 Stuttgart

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comment** |
| 0.1 | 07.09.2018 |  | created |
| 0.1 | 13.04.2020 | Nora Baitinger | Filled with information |
| 1.0 | 08.05.2020 | Nora Baitinger | Added Design and Implementation |
| 1.0 | 10.05.20 20 | Nora Baitinger | Added Module Tests  Final version |

# Content

[1 Content 2](#_Toc40004730)

[2 Scope 3](#_Toc40004731)

[3 Author of this document 3](#_Toc40004732)

[4 Definitions 3](#_Toc40004733)

[5 Figures 3](#_Toc40004734)

[6 Module Requirements 4](#_Toc40004735)

[6.1 User View 4](#_Toc40004736)

[6.2 Requirements 4](#_Toc40004737)

[6.3 Module Context 4](#_Toc40004738)

[7 Analysis 5](#_Toc40004739)

[8 Design 5](#_Toc40004740)

[8.1 Risks 5](#_Toc40004741)

[9 Implementation 6](#_Toc40004742)

[10 Module Test 7](#_Toc40004743)

[10.1 Component Testplan 7](#_Toc40004744)

[10.2 Component Testreport 7](#_Toc40004745)

[11 Summary 8](#_Toc40004746)

[12 Appendix 9](#_Toc40004747)

[12.1 References 9](#_Toc40004748)

[12.2 Code 9](#_Toc40004749)

[12.3 Module Test Cases 9](#_Toc40004750)

# 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.

# Author of this document

This document is based on the tool GSD2AML-Converter developed by the TINF17C course. Some of the sections in this document are identical to their module documentation. These sections will be tagged with *\*TINF17C/GSD2AML-Converter Team*. All other sections are written and developed by Nora Baitinger, Antonia Wermerskirch, Lara Mack, Bastiane Storz.

# Definitions

**AML** Automation Markup Language

**AML DD** AML Device Description

**AMLX** AML Package

**CSP+** Control and Communication System Profile

**IODD** Input/Output Device Description

**PN** Profinet

**SAS** System Architecture Specification

**SRS** System Requirements Specification

# Figures

[Figure 1 - Logging in the module context 4](#_Toc40018179)

# Module Requirements

## User View

From the users point of view the Logging module should implement the following aspects:

1. Log information, warnings and error messages regarding the GUI tool in a log file.
2. Log information, warnings and error messages regarding the CLI tool in a log file and as a console output.
3. Providing a logging interface for the library.

## Requirements

The Logging module is linked with the requirements described in LF80.

It states that the module requires to set up a possibility to provide the user with error information to handle problems. Therefore the GUI and the CLI need to be able to display this information.

The information regarding the library shall be accessed by other developers too.

## Module Context

The Logging module interacts with most of the other modules. It is called every time an error occurs, a waring must be displayed or information about the program sequence should be logged.

There is a difference between the logging mechanism regarding the library, the GUI tool and the CLI.

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Figure 1 - Logging in the module context

# Analysis

This module deals with warnings, error messages and logging information about the program. This information shall be understandable to users and developers and should be sorted in one log file per execution. The logfiles should be available for the users and therefore be stored locally to a concise directory.

# Design

The logging module is divided into three different parts. There is the logging interface used in the library and the logger for the command line interface and for the graphical user interface.

First of all the design for the logging interface: It has to provide a log function that includes an enumeration for the different log levels. These different log levels have the values “Info”, “Warning”, “Error”, “Trace”, “Debug”, “Fatal” and “Off”.

All the logs that are provided by this interface contain the log level and a message. Both are written as a new line into the log file for the current conversion execution.

In order to work properly, the library needs a variable to hold a static logger that can be used by other modules. This variable should be set by all programs (in this case the CLI or the GUI) that want to use the library.

The logger of the GUI and the CLI will be implemented as two different classes, one for each of the interfaces. They both define the function called log that is used by the logging interface of the library.

These functions need to fulfil some tasks:

1. Create a log file at the beginning of each execution.
2. Write all the log levels into this file.
3. Add the log messages.
4. Provide the user with important information about the conversion process and eventually occurring problems.

To achieve these goals an external free and open-source logging platform “NLog” can be used. It is a NuGet package available for the .Net platform. The advantage of this platform is that it is widely spread and can be used easily.

## Risks

The Logging module uses the platform “NLog” as an external reference. Only the main function of “NLog” is used so the risk of bugs or other changes in this interface is not very high.

Another problem that could arise is missing privileges regarding the operating system. Missing rights for accessing the file system will endanger the need to write log files and therefore prevent the module from fulfilling its task. However that risk can be eliminated by using administrator privileges.

# Implementation

The tasks described in chapter 8 shall all be implemented by the Logger. Therefore the three different parts mentioned will be implemented as follows:

The library interface is implemented using the example of the “NLog” log function. The log levels correspond to the log levels of “NLog”.

The “Converter” class of the library gets a public static property to provide the other modules of the library with the same logger.

The “Logger” classes of the Dd2Aml.CLI and the Dd2Aml.Gui defines an “NLog.Logger” on which only the log function is called with the corresponding log level and message. All described parts were set public, because they are only used by other modules and programs.\*

*\*TINF17C/GSD2AML-Converter Team*

# Module Test

Two parts of the Logger module can be tested. The logging interface implemented in the Library module can’t be tested because it doesn’t implement its own functions.

However the Logger classes “Logger.cs” of the CLI module and the GUI module are tested with Unit Tests and White-Box view.

The source code this these Unit Tests can be found: [*https://github.com/WAntonia/TINF18C\_Team\_3\_DD2AML-Converter/blob/master/SOURCE/src/Dd2Aml.Test/LoggingTest.cs*](https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/blob/master/SOURCE/src/Dd2Aml.Test/LoggingTest.cs)

## Component Testplan

|  |  |  |
| --- | --- | --- |
| **Test-ID** | **Feature ID** | **Test Specification (Description or TCS)** |
| 1 | LF80 | This test verifies if the CLI creates log files and writes logs into this file. |
| 2 | LF80 | This test verifies if the GUI creates log files and writes logs into this file. |

## Component Testreport

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test-ID** | **Pass/Fail** | **If failed: Test Observation** | | **Date** | **Tester** |
| 1 | Pass | - | 10.05.2020 | | N.Baitinger |
| 2 | Pass | - | 10.05.2020 | | N.Baitinger |

# Summary

The module implements the requirements completely. If an error during the logging process occurs, no information can be shown to the user, because this module is responsible for error documentation.

However this module implements a state-of-the-art logging mechanism for desktop environments and therefore no enhancements are currently needed.

# Appendix

## References

1. SRS: [*https://github.com/WAntonia/TINF18C\_Team\_3\_DD2AML-Converter/wiki/System-Requirements-Specification*](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*](https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/wiki/Systemtestplan)
3. STR: [*https://github.com/WAntonia/TINF18C\_Team\_3\_DD2AML-Converter/wiki/Systemtestreport*](https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/wiki/Systemtestreport)

## Code

The code for this module can be found at four different locations:

* CLI Logger: [*https://github.com/WAntonia/TINF18C\_Team\_3\_DD2AML-Converter/blob/master/SOURCE/src/Dd2Aml.CLI/Logger.cs*](https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/blob/master/SOURCE/src/Dd2Aml.CLI/Logger.cs)
* GUI Logger: [*https://github.com/WAntonia/TINF18C\_Team\_3\_DD2AML-Converter/blob/master/SOURCE/src/Dd2Aml.Gui/Logger.cs*](https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/blob/master/SOURCE/src/Dd2Aml.Gui/Logger.cs)
* Library Logging Interface: [*https://github.com/WAntonia/TINF18C\_Team\_3\_DD2AML-Converter/blob/master/SOURCE/src/Dd2Aml.Lib/Logging/ILoggingService.cs*](https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/blob/master/SOURCE/src/Dd2Aml.Lib/Logging/ILoggingService.cs)
* Logging Variable Converter: [*https://github.com/WAntonia/TINF18C\_Team\_3\_DD2AML-Converter/blob/master/SOURCE/src/Dd2Aml.Lib/Converter.cs*](https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/blob/master/SOURCE/src/Dd2Aml.Lib/Converter.cs)

## Module Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test-ID** | **Name** | **Description** | **Test Steps** | | |
| **Step** | **Action** | **Expected Result** |
| 1 | TestCLILogging | This test verifies if the CLI creates log files and writes logs into this file. | 1 | Create new dd2Aml.CLI logger. |  |
| 2 | Log messages with log levels ‘Info’, ‘Warning’, ‘Error’ and ‘Fatal. | Log file is created and log messages appear in the file. |
| 3 | Log messages with log levels ‘Off’, ‘Debug’ and ‘Trace’. | These log messages do not appear in the log file. |
|  | | | | | |
| 2 | TestGuiLogging | This test verifies if the GUI creates log files and writes logs into this file. | 1 | Create new dd2Aml.CLI logger. |  |
| 2 | Log messages with log levels ‘Info’, ‘Warning’, ‘Error’ and ‘Fatal. | Log file is created and log messages appear in the file. |
| 3 | Log messages with log levels ‘Off’, ‘Debug’ and ‘Trace’. | These log messages do not appear in the log file. |