Hochschule
Augsburg University of
Applied Sciences

Major Project 2020

Master Industrial Security

University of Applied Sciences Augsburg

Project Report

Table 1: Student authors

Student	**Student ID**
Alexander Holzmann	2081881
Markus Kamm	2060929
Michael Wager	2081894

Abstract

Today's enterprises in the producing industry are subjected to continuous change and are required to adapt their production environment capabilities in order to remain competitive. This causes continuously increasing complexity and connectivity of modern production systems. To mitigate potential safety and security incidents as well as optimization of the companies' business case, a configuration management solution represents an effective instrument.

In context of the master course in *Industrial Security* at the University of Applied Sciences, Augsburg, a configuration management project is conducted on a real-world production facility by a group of three students.

Based on a fictional scenario concerning a manufacturer of smartphones (Yeskia Inc.), an initial analysis of the enterprise environment regarding potential risks is performed and appropriate mitigations are derived. Based on the mitigations a comprehensive concept phase is proposing various strategies for realization. Following the V-Model approach, three individual software solutions are designed, implemented and validated. A training course with qualified training material regarding the developed solutions concludes the project.

Contents

1. Initialization Phase

As first step of the project the initialization phase defines the environment and the scope of the project, including the scenario definition and limitations. It is clustered into four subsections, describing the project team setup, the project definition, the project infrastructure and the fictional scenario.

1.1. Project Setup

The team consists of three students, Alexander Holzmann, Markus Kamm and Michael Wager, studying the master course "Industrial Security". The planned kick off meeting represents the official start of this project (06.10.2020). Besides our team "Configuration Management", three other teams work in parallel on different topics:

- **Team Network Security**
  Concerned with network penetration testing and vulnerability mitigation
- **Team Secure Cloud**
  Implementation of a secure cloud environment, collecting production facility data
- **Team Safety**
  Development of functional safety extensions related to the production facility

The cross team collaboration is initialized within this early phase of the project to foster knowledge sharing and synergy leverage. Initiated by the configuration management (CM) team, a weekly synchronization online meeting is scheduled. Further, a shared booking calendar for laboratory reservation provides efficient planning of timeslots, required at the physical facility infrastructure.

1.2. Project Definition

The predefined goal, set by the principal for this project, is to evaluate and setup a suitable CM system for a virtual real world scenario, represented by the education facility (figure 1), available in the laboratory (HSA - room H4.12). This system has the following objectives, among others:

- Ensuring the integrity of the product
- Restoration of previous configurations
- Content and differences of configurations are known
- Traceability of product artifacts
- Tracking of changes

- Costs and time frames are adhered to
- Determining the status of artifacts

The configuration of the system should be known at all times. In the context of the project, this should also enable the restoration of previous configurations.

The most important tasks are:

- Search for free tools to document the configuration.
- Set standards for the other groups regarding documentation of the configurations.
- Train all other groups on the topic of configuration management, so that all configurations are documented.

Figure 1: Education Facility - Lab H4.12

1.3. Project Infrastructure

The team is scheduled for regular work sessions every Monday, Wednesday and Friday afternoon with a planned amount of 20 hours a week. Due to the lack of

a real contracting authority for this project, Prof. Dr. rer. nat. Peter Richard is substituting this role in combination to his position as supervisor. Besides the weekly cross team synchronization meetings, status report calls are established, aligned to suitable milestones for the purpose to keep the customer updated regarding the project status and potential deviations.

To enable seamless collaborative work on any content (e.g. documentation, application code, ...) regarding this project, a *git* [1] repository on *Atlassian BitBucket* [2] is initialized. The accumulated commit statistic, outlined in figure 2, indicates the intense usage of the cloud based infrastructure.

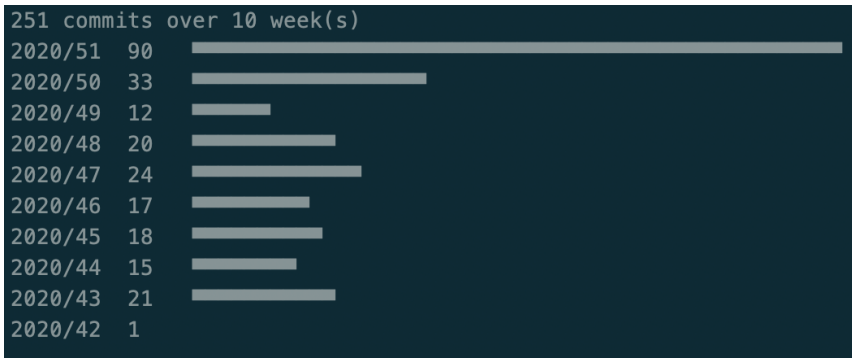


Figure 2: Git repository commit statistic

1.4. Scenario Definition

As the master course is intended to simulate an assignment in a real world setting, the CM project team derived an extensive project scenario, described in `CM_scenario_definition.pdf`, appendix A. The scenario simulates the virtual company Yeskia Inc., with a yearly revenue of 8.3 Billion Euro within the domain of smartphone production. The facilities distribute ten equal production lines producing the A-Phone. Hence the developed CM system could easily be deployed on all ten production lines with almost no additional effort.

Aligned to a realistic scenario, the development and validation phases are divided into three different environments:

1. **Development environment**
   The analysis, concept, design and implementation phase is progressed exclusively using the development environment. This environment covers the notebooks of the team members in combination with online available cloud services.

2. **System test environment**
   To install and validate the developed solutions in a realistic scenario, a system test environment is required. In addition to the development environment,

real facility hardware (HW) components and production software (SW) applications are available at the HSA laboratory educational facility. This environment provides a limited amount of representing components of a production line.

3. **Customer production environment**
   For final deployment and validation of the CM system infrastructure, access to the customer environment is required. Due to the virtual character of this project, a fully equipped production line is not available for final customer integration of the CM solution. Hence the system test environment at the HSA laboratory is representing the customer facility in a reduced manner.

The described scenario definition forms the base for all subsequent phases of this project.

2. Analysis Phase

Following the project initialization phase, the analysis phase represents the entry point for working on the subject of the assignment. The analysis phase is split into three main chapters, processed in parallel: risk assessment, requirements specification and infrastructure analysis.

2.1. Risk Assessment

A risk assessment identifies and covers all potential risks of a defined scenario area. Further it contains specific values regarding likelihood as well as the potential negative impact. A comprehensive risk assessment is mandatory for analyzing and selecting proper mitigations with the goal of decreasing the overall risk, making this project a success. Without the availability of this extensive information basis, the selected and possibly implemented mitigations will not ensure the desired risk reduction and protection effectively. Considering the business case target, investing in a comprehensive risk assessment is reasonable from an economic perspective. In conclusion, the described risk assessment builds up the basis for developing the subsequent roadmap throughout the whole project. As this project scopes on mitigations, achievable with introduction of a CM system, further possible solutions, mitigating the mentioned risks (e.g. hardening of devices), are not considered here. Other projects should go ahead, investigating other areas regarding potential mitigations, decreasing the overall risk of the facility.

The risk assessment is developed according to the international norm *ISO31000:2018 Risk management - Guidelines* [3]. As described within the mentioned norm, the risk assessment is divided into three process steps, *risk identification*, *risk analysis* and *risk evaluation*.

The following sections extensively explain the structure and content of the developed risk assessment.

2.1.1. Step 1 - Risk Identification

Starting the risk assessment, identifying potential risk assets is the first important step of this process. It is mandatory to discover any possible risk scenario before going ahead with the subsequent steps of the analysis. To avoid a premature rating of the assets during identification, the analysis of their impact and likelihood is postponed to the second step. To capture all potential relevant risks, the layout analysis of the HW and SW infrastructure is processed in parallel. This structured approach ensures covering all relevant assets to the extent possible. Interviews with experts from customer side would have supported this step further in a real world scenario. Business case irrelevant rated assets are filtered in later stages of the assessment. As a conclusion, the risk identification list represents an unbiased collection of any identified possible asset. The complete risk assessment document (filename: `Risk Assessment.xlsm`) is available within the project submission folder.

Figure 3 shows an extract of the created risk assessment document, focusing on step 1 - *Risk Identification*.

Step 1 - Risk Identification						
ID	Object	Vulnerability	Actor	Threat	Motivation	Impact
1 - Festo MES4 System						
1.1	Database	No content change logging	Operator (internal) SE Attacker (external) Attacker (internal)	Modification or corruption of database content lead to unwanted side effects. Unauthorized access to sensitive production data.	Unintentional: Improve, adapt and maintenance of production process work plans (functions). Intentional: Production data modification or theft.	Downtime, Non-conforming Production, Data theft
2 - Siemens Simatic S7 ET200SP CPU1515-SP						
2.1	FW	No firmware version logging	Siemens Online Support SE Attacker (external) Attacker (internal)	Different FW versions result in incompability with application software / other components. Firmware downgrade with the objective to revert security related patches.	Unintentional: Patches for security and functional improvements Intentional: FW downgrade attack	Downtime, Non-conforming Production, Safety incident, Data theft
2.2	APP SW	No application SW version logging	Dev, SE	Changes in application SW lead to unwanted side effects	Unintentional: Improve, adapt and maintenance of software- or safety functions	Downtime, Non-conforming Production, Safety incident
2.3	APP SW	Missing continous surveillance of application SW binary integrity	Dev, SE Attacker (external) Attacker (internal)	Changing application code without version increment to hide manipulation.	Intentional: Bypass release management, Injection of malicous code	Downtime, Non-conforming Production, Safety incident, Data theft
2.4	APP SW	Missing traceability of application SW content-related changes	Dev, SE	Missing traceability of content-related changes in application SW lead to excessive efforts for rollback or incident investigation.	Unintentional: Improve, adapt and maintenance of software- or safety functions	Downtime, Non-conforming Production, Safety incident
2.5	Para	No paramenter value change logging	Dev, SE Attacker (external) Attacker (internal)	Missing traceability of parameter changes lead to unwanted effects, excessive efforts for rollback or incident investigation required.	Unintentional: Improve, adapt and maintenance of parameter values. Intentional: Manipulation of functionality	Downtime, Non-conforming Production
3 - I/O Link ET200SP IM155-6PN-HF						
3.1	FW	No firmware version logging	Siemens Online Support SE Attacker (external) Attacker (internal)	Different FW versions result in incompability with other components. False analog calibration data can lead to incorrect sensor indications.	Unintentional: Patches for security and functional improvements Intentional: FW downgrade attack	Downtime, Non-conforming Production, Safety incident, Data theft
3.2	Config	No configuration data value change logging	Siemens Online Support SE Attacker (external) Attacker (internal)	Missing traceability of configuration changes enables manipulation of calibration data, leading to deviation of sensor/actor input and output values.	Unintentional: Configuration data change for fast testing or setup. Intentional: Manipulation of production process.	Downtime, Non-conforming Production, Safety incident

Figure 3: Risk Assessment - Step 1 - Risk Identification

The following columns of the table build up the relevant data identifying and describing each asset:

1. **ID**
   A unique ID for identification of the described asset, used as reference throughout the project report and its additional documents. The Id (X.Y) is structured as follows.
   - X: Identifying the HW component of the production facility as hook for identifying the embedded objects
   - Y: Asset referenced to the component X
2. **Object**
   The specific artifact of the component, subjected to the described risk. The objects cover any SW artifact as well as HW items.
3. **Vulnerability**
   This field states the specific vulnerability regarding the component and the related artifact. This information supports later evaluation of possible mitigations.
4. **Actor**
   Potential actors to consider regarding each asset. The list of actors differs between assets, e.g. remote attackers are not relevant for local access limited threats.
5. **Threat**
   The threat executed by an actor, potentially leading to a negative impact.
6. **Motivation**
   The specific motivation or indirect reason (e.g. development of new functionality), triggering the actor to execute the threat.
7. **Impact**
   The potential impact scenario classification of the assets without impact values.

The comprehensive amount of information, evaluated during the identification phase, enables a better estimation of the assets likelihood and the potential negative impact value in phase 2 of the risk assessment.

2.1.2. Step 2 - Risk Analysis

After identification of relevant assets, step 2 of the risk assessment covers the analysis of their likelihood and the potential negative impact as concrete values. The given numbers are best guess estimations, based on the evaluated infrastructure information and the scenario definition. The result builds up the input data for filtering the

identified assets regarding their criticality in consideration with the specific business case threshold of Yeskia Inc. The mentioned values are related to one single production line.

Figure 4 shows an extract of the created risk assessment document, scoping step 2 - *Risk Analysis*.

			Step 2 - Risk Analysis (one production line)								
ID	Object	Vulnerability	Events (year)	Probability of occurrence (relative)	Impacts (year)	Impact: Downtime	Impact: NCP	Impact: SI	Impact: DT	Impact sum	Risk w/o CM
1 - Festo MES4 System											
1.1	Database	No content change logging	7	0,125	0,875	31.511,42 €	693.033,32 €	0,00 €	1.500.000,00 €	2.224.544,74 €	1.946.476,65 €
2 - Siemens Simatic S7 ET200SP CPU1515-SP											
2.1	FW	No firmware version logging	4	0,25	1	73.526,64 €	126.006,06 €	9.000,00 €	200.000,00 €	408.532,70 €	408.532,70 €
2.2	APP SW	No application SW version logging	4	0,125	0,5	378.137,03 €	504.024,23 €	50.000,00 €	0,00 €	932.161,26 €	466.080,63 €
2.3	APP SW	Missing continous surveillance of application SW binary integrity	1	0,5	0,5	472.671,29 €	399.019,18 €	25.000,00 €	600.000,00 €	1.496.690,47 €	748.345,24 €
2.4	APP SW	Missing traceability of application SW content-related changes	4	0,0625	0,25	819.296,90 €	945.045,44 €	9.000,00 €	0,00 €	1.773.342,33 €	443.335,58 €
2.5	Para	No paramenter value change logging	4	0,2	0,8	42.015,23 €	378.018,17 €	0	0	420.033,40 €	336.026,72 €
3 - I/O Link ET200SP IM155-6PN-HF											
3.1	FW	No firmware version logging	4	0,0625	0,25	73.526,64 €	115.505,55 €	9.000,00 €	200.000,00 €	398.032,20 €	99.508,05 €
3.2	Config	No configuration data value change logging	0,5	0,5	0,25	68.274,74 €	189.009,09 €	9.000,00 €	0,00 €	266.283,83 €	66.570,96 €

Figure 4: Risk Assessment - Step 2 - Risk Analysis

The subsequent described columns outline the required index values:

1. **Events (year)**
   The estimated number of event occurrences per year. The event describes all executed actions (e.g. SW development, Firmware (FW) update) not necessarily leading to an incident.

2. **Probability of occurrence (relative)**
   The probability of a negative impact for each occurred event

3. **Impacts (year)**
   The calculated number of events with negative impact on year base (no. of events * probability of occurrence)

4. **Impact: Downtime**
   The absolute impact value for the downtime scenario within each asset and impact occurrence

5. **Impact: NCP**
   The absolute impact value for the non-conforming production time within each asset and impact occurrence

6. **Impact: SI**
   The absolute impact value for the safety incident scenario within each asset and impact occurrence

7. **Impact: DT**
   The absolute impact value for the security data theft scenario within each asset and impact occurrence

8. **Impact sum**
   The sum of all four impact scenarios. This value represents the expected impact amount for each impact occurrence.

9. **Risk w/o CM**
   This value represents the yearly expected absolute impact amount for one production line. This value considers the impact and the likelihood of incidents.

For estimation of the specific potential impact values for a possible impact occurrence, the business scenario of Yeskia Inc. is developed and summarized within a reference table, shown in figure 5. The calculations covered in the risk analysis step of the risk assessment are directly linked to these values to ensure consistent and comparable results.

Impact Estimation

Downtime	
Basic production cost(year)	50.000.000 €
Loss of sales (hour)	20.552 €
Impact (hour)	26.260 €
Non conforming production	
Basic production cost(year)	50.000.000 €
No. of devices (hour)	105 €
Prod costs (device)	249 €
Impact (hour)	52.503 €
Safety incident	
Small physical damage	180.000 €
Large physical damage	500.000 €
Human harm	150.000 €
Human death	1.500.000 €
Data theft	
Small	4.000.000 €
Large	10.000.000 €

Figure 5: Risk Assessment - Impact Estimation

1. **Downtime**
   In case of facility downtime, the factory is not able to produce devices using the affected production line.
   - *Basic production cost (year)*
     The basic production cost represents the amount of budget still required, even without active production (e.g. facility costs, personnel costs, IT infrastructure costs, ...)
   - *Loss of sales (hour)*
     The number of devices not sold due to unavailability within the market
   - *Impact (hour)*
     The resulting impact value per hour (hourly basic costs + loss of sales)
2. **Non-conforming production (NCP)**
   Malfunction of the production facility can lead to non conforming production (NCP) resulting in (partial) defective devices produced.
   - *Basic production cost (year)*
     As the devices are not sellable, the first expense equals the production line downtime costs.
   - *No. of devices (hour)*
     The production capacity of one production line in number of devices per hour
   - *Production costs (device)*
     The material investment costs for each individual device (lost with each defective device)
   - *Impact (hour)*
     The calculated impact value per hour (hourly downtime costs + material expense of defective produced devices per hour)
3. **Safety incident**
   Besides the production factors, potential safety incidents leads to undesired costs, too.
   - *Small physical damage*
     Constant defined value for a small physical damage of the facility
   - *Large physical damage*
     Constant defined value for a large physical damage of the facility
   - *Human harm*
     Constant defined value for a human harm scenario
   - *Human death*
     Constant defined value for a human death scenario

4. **Data theft**
   Security issues could lead to potential data theft scenarios.
   - *Small*
     Constant defined amount for a small data theft scenario
   - *Large*
     Constant defined amount for a large data theft scenario

The developed analysis values are mandatory for evaluating the assets (step 3) and selecting the most critical tasks to implement effective mitigations. The relevant result of the risk analysis phase is visually summarized in figure 7.

2.1.3. Step 3 - Risk Evaluation

After identification and risk analysis of the assets, step 3 of the risk assessment covers the potential risk reduction implementing suitable mitigations. As mitigations are able to reduce but usually not completely eliminate the risk, the result of step 3 contains the residual risk considering CM relevant mitigations. In a real world scenario, the risk reduction of each asset is individually compared against the effort implementing the linked mitigation, resulting in the return on invest (ROI) ratio. The implementation effort for mitigations, covering multiple assets, could be distributed. The mentioned values are related to one single production line.

Figure 6 shows an extract of the created risk assessment document, scoping step 3 - *Risk Evaluation*.

The subsequent described columns outline the required index values:

1. **Residual impact with CM**
   The resulting absolute residual impact value with a suitable mitigation implemented
2. **Impact reduction**
   The absolute impact value reduction achievable with a suitable mitigation
3. **Impact reduction (%)**
   Relative ratio reducing the absolute impact value
4. **Impacts (year) with CM**
   Number of negative impact occurrences with a suitable mitigation implemented
5. **Impacts (year) reduction (%)**
   Relative ratio reducing the impact occurrence number

			Step 3 - Risk Evaluation						
ID	Object	Vulnerability	Residual impact with CM	Impact reduction	Impact reduction (%)	Impacts (year) with CM	Impacts (year) reduction (%)	Residual Risk (year) with CM	Risk Reduction
1 - Festo MES4 System									
1.1	Database	No content change logging	667.363,42 €	1.557.181,32 €	70	0,525	40	350.365,80 €	1.596.110,85 €
2 - Siemens Simatic S7 ET200SP CPU1515-SP									
2.1	FW	No firmware version logging	122.559,81 €	285.972,89 €	70	0,7	30	85.791,87 €	322.740,84 €
2.2	APP SW	No application SW version logging	466.080,63 €	466.080,63 €	50	0,35	30	163.128,22 €	302.952,41 €
2.3	APP SW	Missing continous surveillance of application SW binary integrity	299.338,09 €	1.197.352,38 €	80	0,45	10	134.702,14 €	613.643,09 €
2.4	APP SW	Missing traceability of application SW content-related changes	532.002,70 €	1.241.339,63 €	70	0,1	60	53.200,27 €	390.135,31 €
2.5	Para	No paramenter value change logging	147.011,69 €	273.021,71 €	65	0,4	50	58.804,68 €	277.222,04 €
3 - I/O Link ET200SP IM155-6PN-HF									
3.1	FW	No firmware version logging	99.508,05 €	298.524,15 €	75	0,175	30	17.413,91 €	82.094,14 €
3.2	Config	No configuration data value change logging	186.398,68 €	79.885,15 €	30	0,125	50	23.299,84 €	43.271,12 €

Figure 6: Risk Assessment - Step 3 - Risk Evaluation

6. **Residual Risk (year) with CM**
   This value represents the yearly expected residual impact amount for one production line with a suitable mitigation implemented. This value considers the residual impact amount and the residual likelihood of incidents.

7. **Risk Reduction**
   The overall amount (likelihood and impact value) of risk reduction achievable with a suitable mitigation

The relevant result of the risk evaluation phase is visually summarized in figure 8.

2.1.4. Result - Assets

The figure *Risk Matrix without CM* summarizes the results of step 2 (chapter 2.1.2), considering all identified assets of the risk assessment, linked using the individual unique ID. The threshold representing the business case target is drawn as a line, separating the critical sector (top-right) from the non-critical area (bottom-left). All assets above the threshold are considered in the subsequent phases of the project. Assets below the defined threshold are postponed in the current project work and could possibly be handled in follow-up projects. All identified assets are placed within the chart, depending on two parameters:

1. *Impact*
   The estimated absolute value related to one impact occurrence

2. *No. impacts / year (likelihood)*
   The expected occurrences of impacts on yearly base

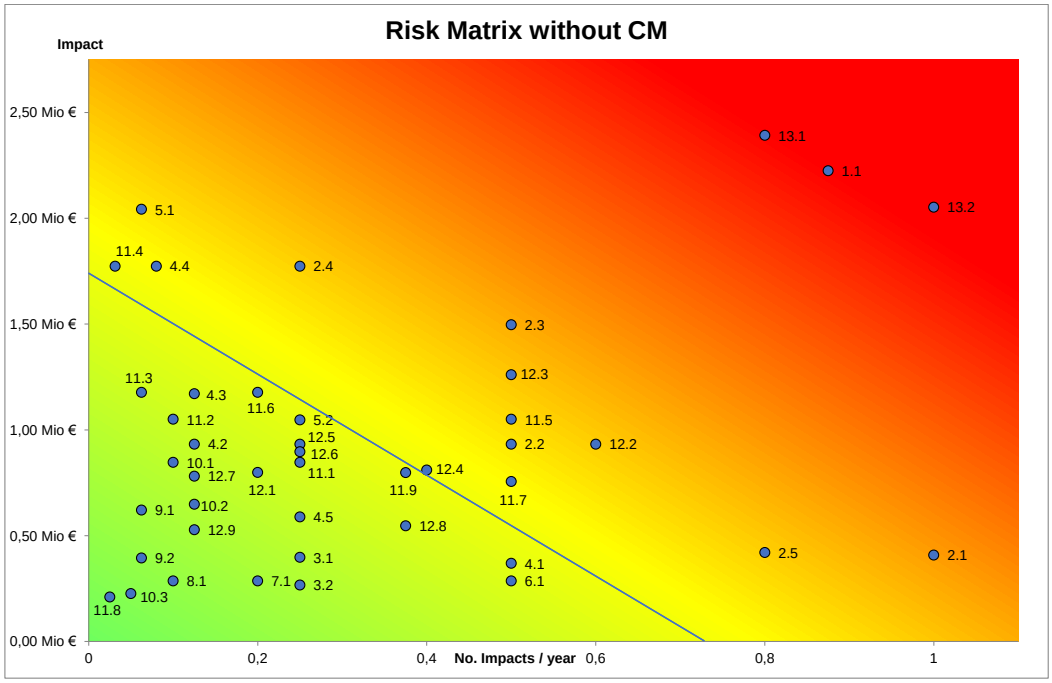


Figure 7: Risk Assessment - Assets without CM mitigation

The following listed assets are placed above the target threshold limit, hence classified as critical:

- **1.1** Festo MES4 System - Database - No content change logging
- **2.1** Siemens Simatic S7 ET200SP CPU1515-SP - FW - No firmware version logging
- **2.2** Siemens Simatic S7 ET200SP CPU1515-SP - APP SW - No application SW version logging
- **2.3** Siemens Simatic S7 ET200SP CPU1515-SP - APP SW - Missing continous surveillance of application SW binary integrity
- **2.4** Siemens Simatic S7 ET200SP CPU1515-SP - APP SW - Missing traceability of application SW content-related changes
- **2.5** Siemens Simatic S7 ET200SP CPU1515-SP - Para - No parameter value change logging

- **4.4** HMI TP700 - User MGMT - No user management logging
- **5.1** D-Link DAP-1665 - Config - No configuration logging
- **11.4** Festo SBOC-Q-R3C Camera - APP SW - Missing traceability of application SW (SPS) content-related changes
- **11.5** Festo SBOC-Q-R3C Camera - APP SW - No application SW version logging (optical verification program)
- **11.7** Festo SBOC-Q-R3C Camera - APP SW - Missing continuous surveillance of application SW (optical verification program) binary integrity
- **12.2** KUKA KRC4 Robot Control - APP SW - No robot control application SW in "KUKA Language" version logging
- **12.3** KUKA KRC4 Robot Control - APP SW - Missing continuous surveillance of robot control application SW in "KUKA Robotic Language" binary integrity
- **12.4** KUKA KRC4 Robot Control - APP SW - Missing traceability of application SW (robot control application SW in "KUKA Robotic Language") content-related changes
- **13.1** Remaining hardware infrastructure - Missing physical access protection
- **13.2** Remaining hardware infrastructure - No hardware component replacement or modification logging

2.1.5. Result - Mitigations

The identified critical assets are investigated regarding possible mitigations, implementing a CM system within the production facilities, to reduce the overall risk below an acceptable value. As the assets differ widely with regard to their threats, the development of various suitable mitigations is required. The following enumeration lists all evaluated mitigations, linked to the assets applied. A short description outlines the mode of action for each mitigation.

- **MIT1**: Automatic firmware version logging
  FW updates are applied on various HW components due to several reasons (e.g. security update, new functionality, downgrade attack, ...). Currently, no tracking of any FW update is available, leading to unknown configurations and a missing update history. As FW is providing basic low-level functionality and application programming interfaces (APIs), malfunction or security issues are not excludable. This mitigation automatically tracks any FW version changes caused by desired updates or security attacks, sending notification mails to

the maintenance administrator and finally logging the history of changes for continuous traceability.
Relevant assets: 2.1

- **MIT2**: Automatic application version logging
  Analogous to MIT1, similar scenario applies to application updates on any component. This mitigation automatically monitors the installed application versions, immediately notifying and logging any changes (desired and malicious) leading to potential malfunction or security issue.
  Relevant assets: 2.2, 11.5, 12.2

- **MIT3**: Detection and logging of unauthorized artifact changes
  Service engineers have the necessity to change code inside applications. Due to respecting the complete development process requires time-consuming release test procedures, the process is often bypassed and the modifications are installed "stealthy" without increasing the application version. As MIT2 will not apply for unchanged versions, another mitigation is required. A background surveillance service periodically verifies the installed executable application binary of each HW component connected to. In case of compromised content is detected, the service alarms by sending a notification mail and finally tracks all occurrences within a detection history database. As attackers pursue the same strategy with the target of modifying application runtime code for installing backdoors, this mitigation applies here as well.
  Relevant assets: 1.1, 2.3, 11.7, 12.3

- **MIT4**: Comprehensive development process
  Software has a high level of relevance for production facilities as available in this case. To ensure flawless quality of the produced goods, the quality and consistency of complex software is a very important factor. Unfortunately, many companies, like Yeskia Inc., still have not invested in developing and implementing such a required development process, supported by suitable tooling. This mitigation develops and installs a comprehensive development process implementing tool framework, guiding the users throughout the process and tracking all relevant change requests as well as modifications. To provide rollback possibility in case of production malfunction or component compromise, the system keeps track of all released states, also called baseline freezes. The release packages are available for download with a few clicks.
  Relevant assets: 1.1, 2.4, 11.4, 12.4

- **MIT5**: Detection and logging of parameter changes
  This mitigation is similar to MIT1 and MIT2, but tracking all changes of

parameters, configuration settings and the user management content.
Relevant assets: 2.5, 4.4, 5.1

- **MIT6**: Hardware infrastructure design modification tracking
  Analogous to MIT4, the hardware design is also relevant regarding a proper modification tracking. The mitigation approach is equal to MIT4 but covers any HW design relevant documentation (e.g. schematics, bill of material (BOM), ...) Relevant assets: 13.1, 13.2

- **MIT7**: Implementation of computer aided spare part replacement process
  Replacing defective devices with adequate spare parts by maintenance technicians defines a special but very important scenario. As complex devices, as used within Yeskia´s production facilities, are very sensible regarding interface compatibility, replacing parts with newer revisions could lead to undesired behavior of the whole system. Sometimes, the malfunction is not directly visible after replacement. In this case, it is mandatory to keep track of any replacement process, providing the required information for later investigation. This mitigation implements a computer aided spare part replacement process, leading the maintenance technician throughout the steps, avoiding submission mistakes and keeps tracking of any processed component replacements.
  Relevant assets: 13.2

This list only considers assets classified as critical. All described mitigations are applicable to further, non-critical classified HW components as well.

The figure *Risk Matrix with CM mitigations* illustrates the risk reduction potential of all assets rated above the business case threshold. The graphic represents the result values of step 3 (chapter 2.1.3). Each arrow indicates the specific impact and occurrence reduction for an individual asset. Even if the residual risk of some assets is still above the threshold level, mitigations by implementing a CM system massively helps to reduce the risk towards the limit. In these cases, further mitigations beyond the CM project are suggested (e.g. hardening of devices, implementing processes, physical access control, ...). In combination with the CM mitigations, the desired residual risk values could easily be achieved.

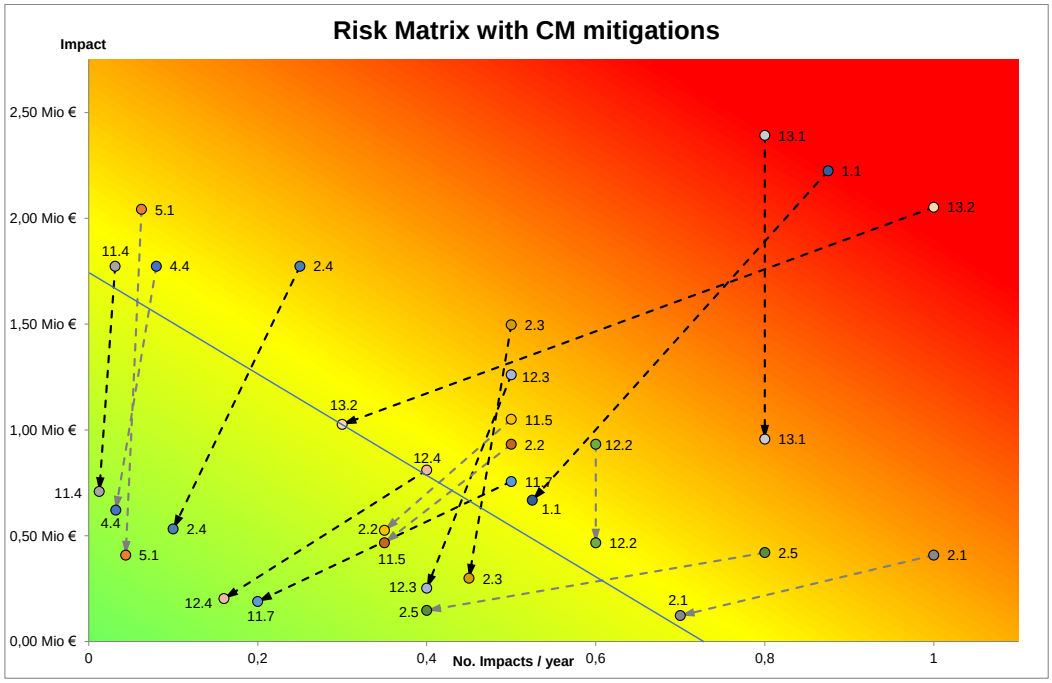


Figure 8: Risk Assessment - Potential Risk Reduction with CM mitigation

As the project is based upon a virtual scenario, all values are estimated to the best of the project teams knowledge. In a real world scenario, the values would be developed together with the contracting client, by initiation of negotiations and workshop meetings. Nevertheless, the developed values are the result of intense evaluations and discussions within the project team, hence they are close to reality.

2.2. Requirements Specification

This chapter describes the process of developing the Software Requirements Specification (SRS). Requirements define the complete demand regarding the system and its software implementation of a product. A recommended practice is defined in *IEEE 830-1998 IEEE Recommended Practice for Software Requirements Specifications* [4]. In order to achieve a target-oriented implementation of the mitigations, defined in the risk assessment, extensive requirements are essential.

The SRS `CM_system_software_requirements_specification.pdf` in appendix B describes the structure of documentation and defines specific requirements to all elaborated mitigation strategies. Due to the limited nature of this project, instead of extensive tooling, a document-based approach is used. In real projects however, good tooling is absolutely mandatory, in order to manage the list of requirements intelligently and ensure traceability of the content. Below we distinguish between business- and non-business requirements. Each requirement can be of type *business*, *security*, or *safety* but also be a combination of those.

2.2.1. Business Requirements

Primary goal of business requirements is to maximize the earnings before interest and taxes (EBIT) of a company. This is accomplished by reduction of potential risk leading to financial negative impact. The required effort implementing the specific mitigations must not exceed the financial outcome of the reduction effect. In conclusion, the return on investment (ROI) value is targeted as high as possible.

2.2.2. Safety Requirements

In addition to business requirements, the safety aspect is also considered and represent through proper requirements. Safety, according to *IEC-61508 Functional Safety* [5], means the protection of humans regarding harm, the environment and the machinery. Therefore, the target of safety requirements is to minimize potential harm and to fulfill the demands described in the stated norm.

2.2.3. Security Requirements

Security defines another area, driven by multiple requesting parties, which needs to be respected within the requirement specification. According to *IEC-62443-3 Security for industrial process measurement and control – Network and system security* [6], the protection regarding industrial cybersecurity have to be considered. The norm specifies four different security levels (SLs), each describing individual requirements. The norm *ISO 27001 Information technology – Security techniques – Information security management systems – Requirements* [7] addresses similar security requirements, but with primary focus on information security and security management. As both norms overlap regarding the main content, and the project is related to an industrial environment, *IEC-62443-3* is chosen for application.

2.3. Infrastructure Analysis

In order to identify components and artifacts relevant to CM, a HW and SW analysis is conducted on the production line in the laboratory. Supplementary to a received hands-on training and practical experience of the project group, a multitude of information sources such as manuals, technical specifications and electrical drawings are used to support the analysis. The following section covers the findings of the analysis, by describing each relevant HW component and its corresponding SW artifacts.

In brief, the concerned exemplary production line is composed of two independent, yet interconnected Festo CP Factory modules. The purpose of the set-up is the emulation of a production line in the domain of a smartphone production. The core task is the correct installation of one or multiple fuses in a workpiece, representing a smartphone circuit board. The production cycle includes storage

and transport of blank and equipped circuit boards, fuse installation and optical verification (automatic optical inspection (AOI)) of the installation. Communication between components is established using standardized communication protocols

Each cell is serving a distinct purpose as outlined below:

- CP-F-ASRS32
  The CP-F-ASRS32 [8] module is responsible for the storage of the circuit boards. The circuit boards are stored in a stack-rack layout and inserted/removed by a two-axis picker arm. Each board is separately carried on a pallet, which can be identified utilizing an attached radio frequency identification (RFID) tag.
- CP-F-RASS-KUKA
  The CP-F-RASS-KUKA [9] module is responsible for the installation of the fuses and optical verification. The installation of the fuses is accomplished within an assembly rack by a KUKA KR-6 robot. The verification of a successful installation is determined by optical inspection using a camera.

Both modules utilize conveyor belts to transport the pallets within each module and between them. The latter is achieved via a transfer port on the housing of each module.

2.3.1. Hardware Structure

Figure 9 shows the top level layout of the production line with the identified CM relevant components.

- **Festo-PC**
  The Festo-PC is situated within the laboratory and contains the relevant SW components which are used for production line. This includes a variety of SW for the application code development within the programmable logic controllers (PLCs), SW for the motor control setup and other SW relevant for the setup and maintenance of the production line. A central SW component is the Festo-MES4 [10], which acts as manufacturing execution system (MES) framework. The Festo-MES4 will be further discussed in section 2.3.2.
- **Siemens Simatic S7 CPU1515-SP**
  The core element of the production line control system is the Siemens Simatic S7 CPU1515-SP [11], responsible for computing the instructions for the application execution, processing input data, subsequent output signal generation and monitoring of safety functions within the production line. The Simatic S7

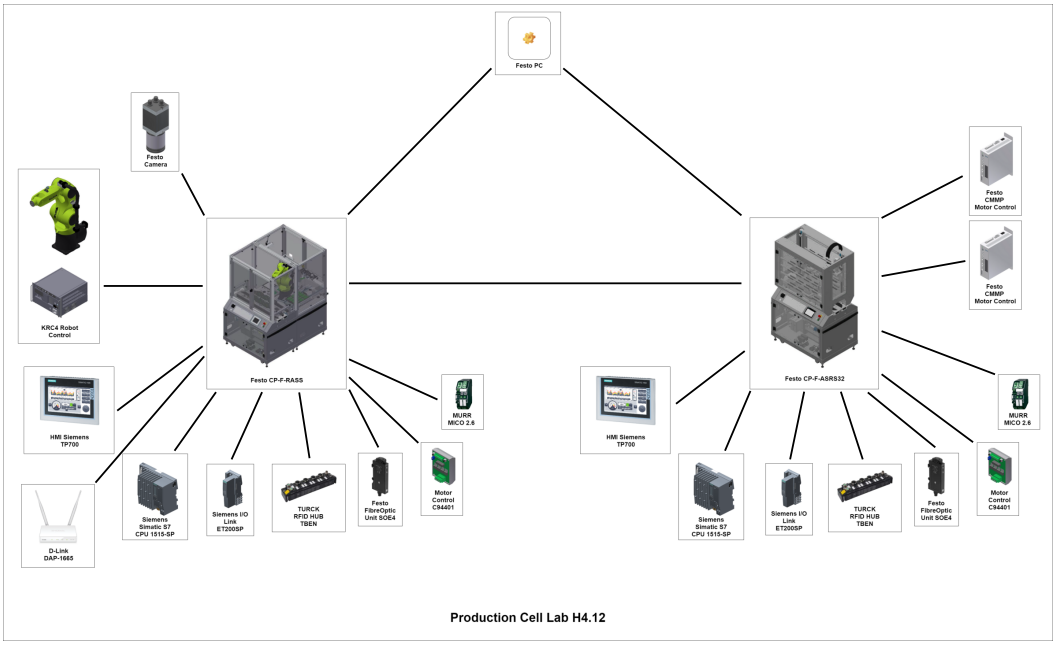


Figure 9: Hardware layout CP-F-RASS-KUKA / CP-F-ASRS32 in Lab H4.12

further performs management of attached device data, controls the communication of those devices and regulates the temporal order of communication channel access. Monitoring of safety functions is required in order to achieve the necessary safety integrity level (SIL) for ensuring the safety of the production line. In case of a malfunction or complete loss of the diagnosis, the SIL could decrease and appropriate safety in the plant is not available anymore.

- **Siemens I/O Link ET200SP**
  The Siemens I/O Link ET200SP [12] provides the actual connection between the PLC and the attached sensor and actor devices within the utilized communication protocol. Further, it serves as a serial communication adapter and performs conversion of analog to digital signals and vice versa. Multiple I/O Link devices can be utilized within a control system to accommodate for available sensor/actor devices.

- **Siemens HMI TP700**
  The production cells contain a Siemens HMI TP700 [13] device which acts as a human machine interface (HMI) to facilitate user interaction and display relevant production data. User inputs are captured using a multi-color touchscreen display. Within the concerned production line, the HMI is used to acknowledge error messages, modify parameters and manually initiating the configuration of RFID tag data.

- **D-Link DAP-1665**
  A D-Link DAP-1665 [14] access point is used for providing wireless network access to a mobile robot for material supply. The mobile robot is not utilized in the production line at the moment and therefore it is out of the scope of the CM project. Yet, the access point is considered relevant, as it remains an active device and provides access to the production network.

- **Festo FibreOptic Unit SOE4**
  Workpiece detection on the conveyor belts, assembly mount and fuse supply rig is achieved by using an optical detection system. The system consists of the Festo FibreOptic Unit SOE4 [15] acting as a controller as well as a light source and optical fibers distributed towards their relevant detection areas. When a workpiece enters a detection area, a light beam consisting of visible infrared, is interrupted which will yield a signal from the controller to the PLC. The setup of light thresholds is done via a physical button on the SOE4 device itself, triggering an internal teach-in function.

- **Motor Control C94401**
  The conveyor belt drive motors are controlled by a C94401 drive motor control [9] device. The device provides electric load control and offers a dynamic breaking function, as well as motor drive speed adjustments. The latter is achieved by altering a potentiometer on the device front.

- **MURR MICO 2.6**
  The MURR MICO 2.6 [16] acts as a central electronic circuit protection device. In a power-on event of the production cell, the device splits the operating voltage into two loads and switching the loads to on-state with a time shift of 75ms to avoid system overload. The system specific operating current can be set by switches on the front side of the device.

- **TURCK RFID HUB TBEN**
  The RFID tag data on each workpiece carrier is accessed via the TURCK RFID HUB TBEN [17] device, consisting of a hub and connected read/write heads. When a carrier is aligned with the read/write head, multiple data variables can be accessed and persistently written on the tag.

- **Festo CMMP-AS**
  The drive motor of each axis of the picker-arm in the storage cell is controlled and monitored by a dedicated Festo CMMP-AS servo motor controller [18]. The device supplies the motor drive with electrical energy, provides start/stop and break functions. To enable proper addressing and communication with the Fieldbus protocols, dual in-line package (DIP) switches are used on the front of the device.

- **Festo SBOC Camera**
  The Festo SBOC Camera [19] performs the AOI of workpieces during the production process and approves or rejects workpieces for further processing. The optical verification checks if the color and orientation of the observed workpiece concur with a predefined target condition. The workpieces are placed on a verification pad for optical examination. The focus of the lens of the camera can be manually adjusted for improving image quality and to adjust for changed environment settings.

- **KUKA KRC4**
  The KUKA KRC4 [20] is the control component for the KUKA robot. In the CM project, the control device and the robot itself are considered as one device. The robot is responsible for workpiece handling from the conveyor to the optical verification rig and to the assembly mount and vice versa. In addition, it is performing the fitting of fuses into the workpiece. The KRC4 controller provides electricity and control signals to the robot.

2.3.2. Software Structure

In parallel to the HW structure, the analysis of the SW artifacts is described in this chapter. Safety related functions, other than diagnosis, are not considered here, as those functions are implemented on a hardware base only. Each of the components listed below, has certain SW artifacts. However, the component specific artifacts are often of the same SW type and are responsible for resembling tasks. To address this, the different SW types are explained in general below and if necessary, more specific details concerning the respective component is given.

- **Firmware** The fimware (FW) is responsible for low-level control of the component HW, security and basic communication with other components. FW is generally updated only to improve functionality, address security issues or provide bug fixes. As the FW is a central part of the component, the manufacturers ensure integrity of the FW by utilizing internal checksums or other measures.

- **Application software** In contrast to the FW, the application SW is provided by the operator/user of the equipment. In the context of production systems, such SW artifacts contain for example execution instructions for the PLC in order to produce parts, configuration data or diagnosis frameworks. Depending on the type of application SW, the frequency of changes to the code is variable. In the concerned production infrastructure, two types of application SWs are prevalent, the so called SPS-Software contains all data relevant to execution instructions for the PLC and connected components. In addition,

some components utilize further application SW, in order to perform internal configuration and operation.

- **Parameter** Parameter values are related to the application SW and describe variables within the execution instructions that are designed to be adjusted in order to accommodate for setup, service or changing production environments. The parameter settings, however remain within the boundaries specified by the application SW instructions.
- **Configuration** Information related to component identification for networking and component baseline settings are represented by the configuration artifacts.

The majority of component SW artifacts can be found within the above stated categories, see table 2.

Table 2: Component software artifact relation

	Firmware	**App. SW**	**Parameter**	**Configuration**	**Other SW**
CPU1515-SP	+	+	+	o	o
I/O Link ET200SP	+	o	o	+	o
HMI TP700	+	+	o	o	+ User Mgmt
DAP-1665	+	o	o	+	o
FibreOptic Unit SOE4	o	o	o	+	o
RFID HUB TBEN	+	o	+	o	o
CMMP-AS	+	o	o	+	o
SBOC Camera	+	+	o	o	+ CheckOpti/Kon
KUKA KRC4	+	+	o	o	+ KUKA KRL

In addition to the above described artifacts, some components implement extended specific SW artifacts, which are described here:

- **Siemens HMI TP700 - User Management**
  The Siemens HMI offers the option to register multiple accounts with different sets of permissions and access. This option is realized using the user management framework of the base operating system.
- **Festo SBOC Camera - CheckOpti**
  To perform the AOI, the camera utilizes the CheckOpti SW, in which all data related to the predefined target condition is specified.
- **KUKA KRC4 - KRL**
  Internal communication and execution instructions are realized within the KUKA robotic language (KRL) framework.
- **Festo - MES4**
  As previously stated, the Festo-MES4 serves as the main SW component for

production control within the production line. Further, it is used as a reporting tool for quality related production data, and performs tasks associated with supervisory control and data (SCADA) systems. All data of the MES4 is stored in a Microsoft Access database (DB), where all data can be accessed using SQL commands. The data contains all information regarding the products to produce, machinery, work plans and production orders [10]. The MES4 communicates with the relevant PLC via acyclic communication to provide production data and via cyclic communication to acquire machinery status information and error messages [21].

3. Concept Phase

As described in chapter 2.1.4, the result of the risk evaluation, aligned to the business case threshold, defines the relevant assets for further consideration in the scope of this project. In consequence, the concept phase focuses on the following identified relevant mitigations (refer to chapter 2.1.5 for detailed description)

- **MIT1**: Automatic firmware version logging
- **MIT2**: Automatic application version logging
- **MIT3**: Detection and logging of unauthorized artifact changes
- **MIT4**: Comprehensive development process
- **MIT5**: Detection and logging of parameter changes
- **MIT6**: Hardware infrastructure design modification tracking
- **MIT7**: Implementation of computer aided spare part replacement process

3.1. Concept Development

The subsequent developed concepts describe the top level architecture design approach and abstract structure of functionality. These concept specifications define the base for the design and implementation phases. The concept phase refers to different actor roles related to the CM domain, available at Yeskia Inc.

- **Administrator**
  The Administrator is responsible for the installed server infrastructure regarding configuration, maintenance and user management.
- **Release Manager**
  The release manager is responsible for defining and creating projects within the CM system. Further, planning, coordination and final approval of project releases describe his main tasks based on defined projects.

- **Service Engineer**
  As direct interface to the customer, the service engineer is one of the primary sources for required product changes. Contributing issues using the CM system, customer related change requests are also taken into account for planned future releases.

- **Developer**
  Besides creating tickets for change requests, the developer mainly contributes with implementing assigned tickets for planned releases.

- **Maintenance Engineer**
  The maintenance engineer maintains the facilities by replacing defective components with adequate spare parts.

3.1.1. Concept MIT1 & MIT2

Automatic firmware & application version logging

The CM provides a service for automatic software version surveillance of control & monitoring components, integrated in the production facility. Monitoring all herein installed software sections enables detection of artifact individual version change events.

Mitigation objectives:

1. **Periodic acquisition of running artifact versions within control and monitoring components of the production system**

   The CM System uses external available communication interfaces of the monitoring components to query the desired version information from the component. An individual communication adapter handles the communication between the CM back end and the component. The component-individual received data is translated in a generic format for further processing.

2. **Detection of version value changes compared to last known state and version change event triggering in case of detected changes**

   The last known state of surveillance relevant components is available even in case of reboot or plant shutdown. Therefore, the CM manages a persistent storage. For each queried event the version is compared to the past version, and, in case of different values, a version change event is triggered. In case of equal version values, no further action is needed.

3. **Persistent storage of each detected version change event including relevant event data**

On incoming version change events, the new version value, including relevant data, should be stored in a history based persistent storage. The stored timestamp should identify the precise point in time related to other events.

4. **Sending notification email for each detected version change event including relevant event data**

   On incoming version change events, the system dispatches a notification email containing the previous and current version, a timestamp and relevant data.

Architecture specification:

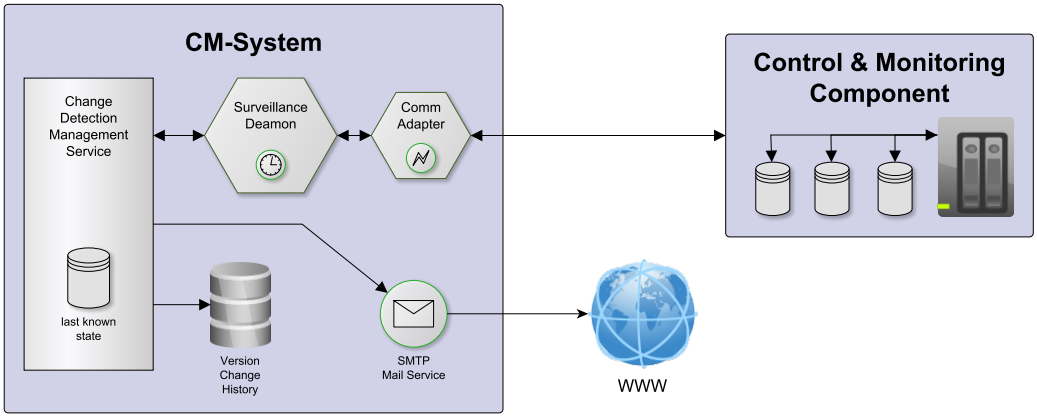


Figure 10: MIT1 & MIT2 - Architecture Concept

Surveillance Daemon:

The surveillance daemon represents the central controlling component, scheduling the surveillance tasks. The implemented scheduler, inside this component, periodically triggers the acquisition process to obtain the current version information from the respective control & monitoring component, installed within the production facility. The acquisition interval values are adjustable defined inside a configuration file. After receiving the requested version data, the surveillance daemon passes the information towards the change detection management service for further processing.

Communication Adapter:

As the production facility is build upon a heterogeneous combination of control & monitoring components, provided by various manufacturers, each individual communication path could differ regarding used communication protocol and/or data format. Therefore, several communication adapter implementations harmonize the specific connection requirements towards a generic API including a standardized data format.

Change Detection Management Service:

The change detection management service is responsible for maintaining the last known version states of all relevant control & monitoring components. To achieve this functionality, an internal persistent storage keeps track of the required version information. The stored values can be restored at configuration management service startup. Using the known state information, the detection service is able to compare the received version value against the last known value and determine whether the version has changed. In case of a change event, the service executes the following tasks:

- Create new version change record: Create a new data record to append the event related data to the history based persistent storage database.
- Send notification email: Send a notification email using a suitable simple mail transfer protocol (SMTP) service implementation in combination with a mail server connection to notify a predefined recipient regarding the version change event. The destination email address is defined within a global configuration file.
- Update last known version state entry: Update the correlating data record within the local persistent known state storage container, using the processed version value, currently running on the control & monitoring component.

3.1.2. Concept MIT3

Detection and logging of unauthorized artifact changes

Analogous to MIT1 & MIT2 (Automatic firmware & application version logging), the CM provides a service for automatic software binary content surveillance of control & monitoring components, integrated in the production facility. Monitoring all herein installed software sections enables detection of artifact individual binary content modification events.

Mitigation objectives:

1. **Periodic acquisition of running artifact binary content or content representing checksum within control and monitoring components of the production system**

   The CM system uses external available communication interfaces of the monitoring components to query the desired content information from the component. An individual communication adapter handles the communication between the CM back end and the component. The component-individual received data should be translated in a generic format for further processing.

2. **Detection of binary content modifications compared to last known state and modification event triggering in case of detected modifications**

   The last known state of surveillance relevant components is available even in case of reboot or plant shutdown. Therefore, the CM manages a persistent storage. For each queried event the content representing data is compared to the past content state, and, in case of dissimilarity, a modification event is triggered. In case of unchanged binary content, no further action is needed.

3. **Persistent storage of each detected content modification event including relevant event data**

   On incoming content modification events, the new content representing dataset, including relevant data, should be stored in a history based persistent storage. The stored timestamp should identify the precise point in time related to other events.

4. **Sending notification email for each detected content modification event including relevant event data**

   On incoming content modification events, the system dispatches a notification email containing information regarding the affected component and artifact, a timestamp and further relevant data.

Architecture specification:

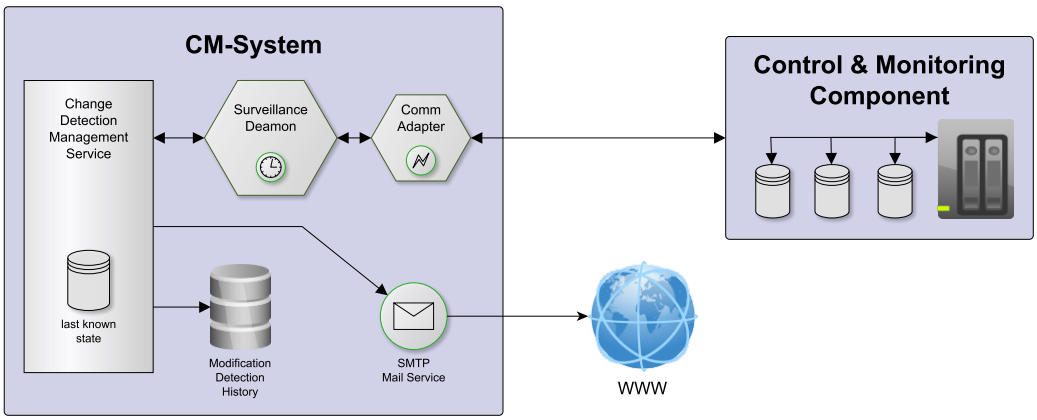


Figure 11: MIT3 - Architecture Concept

Surveillance Daemon:

The surveillance daemon represents the central controlling component, scheduling the surveillance tasks. The implemented scheduler, inside this component, periodically triggers the acquisition process to obtain the current binary content dataset

from the respective control & monitoring component, installed within the production facility. The acquisition interval values are adjustable defined inside a configuration file. After receiving the requested dataset, the surveillance daemon passes the information towards the modification detection management service for further processing.

Communication Adapter:

As the production facility is build upon a heterogeneous combination of control & monitoring components, provided by various manufacturers, each individual communication path could differ regarding used communication protocol and/or data format. Therefore, several communication adapter implementations harmonize the specific connection requirements towards a generic API including a standardized data format.

Modification Detection Management Service:

The modification detection management service is responsible for maintaining the last known artifact content states of all relevant control & monitoring components. To achieve this functionality, an internal persistent storage keeps track of the required content representing information. The stored datasets could be restored at configuration management service startup. Using the known state information, the detection service is able to compare the received content dataset against the last known state and determine whether the content has been modified. In case of a modification event, the service executes the following tasks:

- Create new content modification record: Create a new data record to append the event related data to the history based persistent storage database.
- Send notification email: Send a notification email using a suitable SMTP service implementation in combination with a mail server connection to notify a predefined recipient regarding the content modification event. The destination email address is defined within a global configuration file.
- Update last known content state entry: Update the correlating data record within the local persistent known state storage container, using the processed content dataset, representing the currently running artifact on the control & monitoring component.

3.1.3. Concept MIT4 & MIT6

Comprehensive development process

A comprehensive development process is absolutely essential for providing sufficient quality of release artifacts in context of highly complex production facilities. The CM supports the application of such a development process in a substantial

manner. All phases of the V-model based development process are covered by the implementation of a suitable CM. The CM considers any change request concerning software related artifact changes and hardware infrastructure modifications.

Mitigation objectives:

1. **Development life-cycle monitoring**

   The CM is monitoring the development life-cycle of each functional change request to provide a guided development process.

2. **Transparent development documentation**

   The CM tracks documentation for each functional change request in a history based manner.

3. **Development history traceability**

   The CM tracks state changes of each functional change request for the complete time of development to enable detailed process traceability.

4. **Ticket state change email notification**

   The CM sends an email notification for each ticket state change to keep responsible parties informed regarding the development process.

5. **Source code transparency**

   The CM framework enables source code (incl. HW design schematics) transparency to avoid hidden design changes.

6. **Implementation history traceability**

   The CM tracks code and design changes during implementation phase to provide history based traceability and comparability.

7. **Release planning**

   The CM supports the release manager regarding strategic release planning, considering business case and market requirements.

8. **Release life-cycle monitoring**

   The CM keeps track of the actual release life-cycle state to trigger appropriate actions.

9. **Release build process**

   The CM implements automatic build process functionality to reduce release generation efforts and to streamline continuous delivery.

10. **Release history traceability**

    The release management system within the CM facilitates release-comparability, -traceability and changelog generation.

11. **Release email notification**

    The CM sends an email notification for each release life-cycle state change to keep responsible parties informed regarding the development process.

Architecture specification:

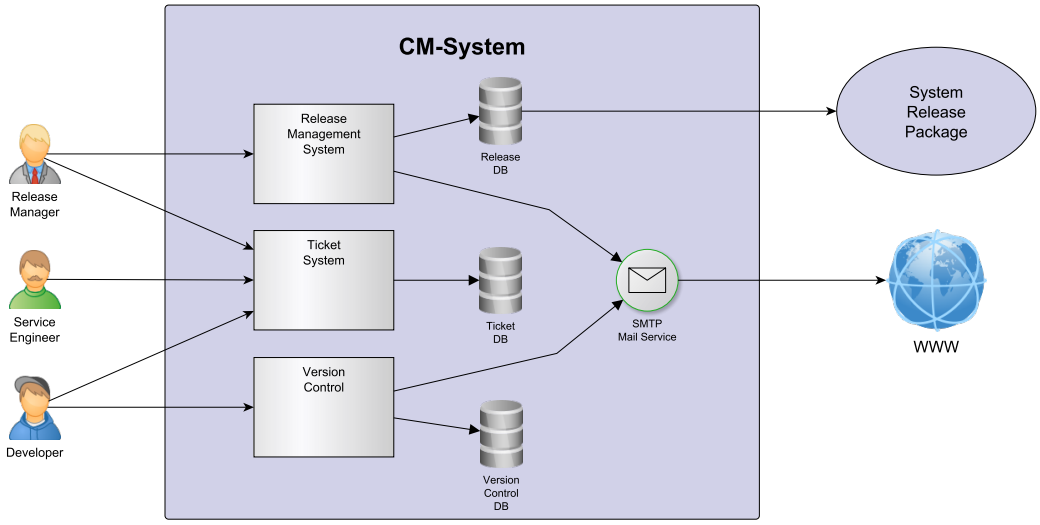


Figure 12: MIT4 & MIT6 - Architecture Concept

Ticket system

The ticket system is a central component, managing change requests for SW and HW releases, targeting the HW infrastructure or specific control & monitoring components of the system. It provides a user interface, allowing different actors (release manager, service engineer, developer) the creation and administration of these change requests. The ticket system implements the objectives 1 - 4.

Version control

The version control system is a central component, managing changes to source code, documentation files and other related source code level content. Every change to the version control system is directly linked to a ticket. Hence, no source code changes are possible without a change request. The version control system implements objectives 5 & 6.

Release management

The release management supports the release manager regarding the planning and realization of system releases for the HW infrastructure or specific control & monitoring components. After evaluation of pending functional change requests within the change control board (CCB), the release is directly aligned to a desired set of functional changes. The system keeps track of the release life cycle state to benchmark the progress against the underlying release plan. Further, a tool framework within the release management provides automatic system release integration, verification and validation prior to the release deployment. In consequence, the system supports automatic quality assurance for the release packages. The release management system is responsible for objectives 7 - 11.

3.1.4. Concept MIT5

Detection and logging of unauthorized parameter changes

Contrary to MIT1 & MIT2 (Automatic firmware & application version logging) and MIT3 (Detection and logging of unauthorized artifact changes), the CM provides logging of parameter value changes of control & monitoring components, integrated in the production facility. Due to value based logging depth, extensive traceability can be achieved. To provide the necessary synchronized timing requirements, a polling approach is not sufficient. Hence, the parameter value logging service is realized using an event based implementation.

Mitigation objectives:

1. **Immediate parameter value change messages, triggered from the control & monitoring component for each parameter value change event**

   Control & monitoring components are responsible for actively sending out parameter value change messages in case values are changed. Each message contains a key for parameter instance identification and its adjusted value. To avoid extensive message bursts due to high frequent parameter changes from the operator, an inhibit time filter is necessary.

2. **Listening for reception of parameter value change messages within the CM**

   The network message server passively listens for incoming parameter value change messages. Each identified parameter value change message is instantly extended by a local timestamp.

3. **Persistent storage of all received parameter value change events including relevant event data**

All passed parameter value change events from the network message server are processed within the change monitoring service. On incoming content modification events, the new content representing dataset, including relevant data, should be stored in a history based persistent storage. To provide versatile capabilities related to reporting, a suitable storage strategy is required.

4. **Sending report email containing summarized parameter value change information for a given time period**

   To avoid extensive email flood caused by value change granular email triggering, a summarized report email is generated and dispatched at a certain, configurable point in time. This report information regarding the affected component and artifact, a timestamp and further relevant data.

Architecture specification:

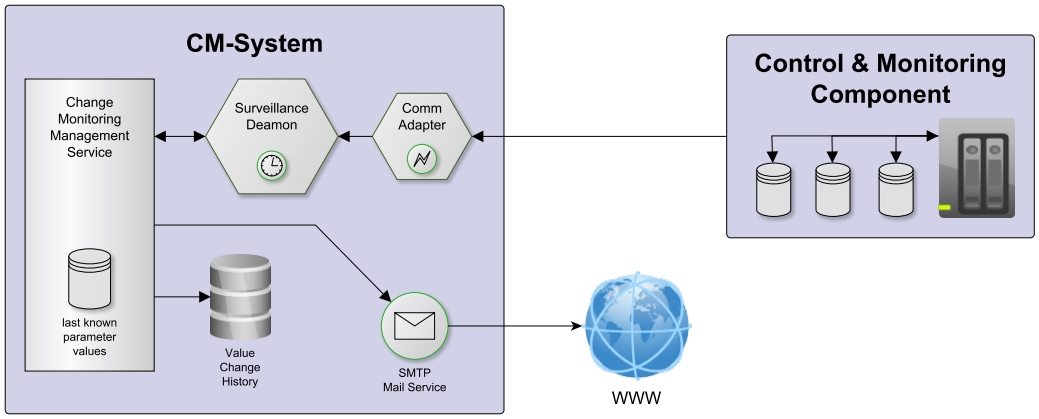


Figure 13: MIT5 - Architecture Concept

Surveillance Daemon:

The surveillance daemon implements the monitoring server component, listening for incoming parameter value change notification messages. After receiving the notification message from the respective control & monitoring component, the surveillance daemon passes the information towards the change monitoring management service for further processing.

Communication Adapter:

As the production facility is build upon a heterogeneous combination of control & monitoring components, provided by various manufacturers, each individual communication path could differ regarding used communication protocol and/or data format. Therefore, several communication adapter implementations harmonize the

specific connection requirements towards a generic API including a standardized data format.

Change monitoring management service:
The change monitoring management service is responsible for tracking incoming events. In case of a modification event, the service executes the following tasks:

- Create new parameter value change record: Create a new data record to append the event related data to the history based persistent storage database.

The stored data is provided towards the surveillance daemon on request for creating the content for the summary email.

3.1.5. Concept MIT7

Computer aided spare part replacement process
The CM implements the spare part replacement process to support maintenance personal executing maintenance tasks. This enables a guided process for electronic replacement documentation as a primary functionality.

Mitigation objectives:

1. **Web based portable frontend interface**

   The CM provides a web based frontend interface for maintaining personal for the documentation of a completed spare part replacement task. To optimize usability, the possibilities of selection are guided and checked for incorrect entries.

2. **Persistent storage of replacement task**

   The CM creates a persistent record for each replacement task from the entered data to store the data for later usage.

3. **Update of persistent inventory database**

   The CM updates the facility-specific BOM based on replacement task input data. Due to the CM updating the inventory database, the BOM represents the current state of the HW infrastructure.

4. **Spare part replacement email notification**

   The CM sends an email notification for each spare part replacement task to keep responsible parties informed regarding the replacement process.

5. **Maintenance task report**

After completion of the replacement task, the CM generates a maintenance task report containing all relevant information for usage by the maintaining technician.

6. **Spare part replacement journal**

   On request, the CM provides a chronologically sorted overview of all stored spare part replacement tasks to gain traceability.

7. **Inventory status report**

   On request, the CM provides a comprehensive overview representing the complete state of the plant, including relevant additional information (e.g. date of last replacement).

Architecture specification:

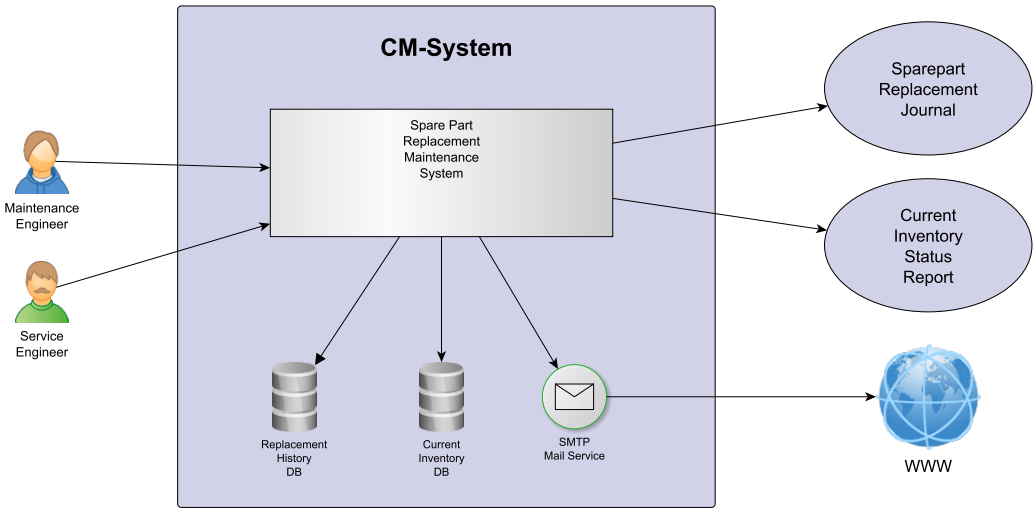


Figure 14: MIT7 - Architecture Concept

Spare Part Replacement Maintenance System: The spare part replacement maintenance system is the central component, responsible for managing spare part replacement tasks. Herein implemented DB connector modules provide the necessary persistent storage of the complete inventory list, containing all relevant information regarding installed HW components of the production system. Further, the connector persistently keeps track of all replacement tasks in a chronological order. The maintenance technician uses the provided frontend interface to conduct the documentation of executed spare part replacement tasks. For interoperability reasons, the frontend implementation is realized using state-of-the-art web based technology. The current inventory DB contains all elements of the facility BOM covering the following information:

- Material equipment identifier (primary key)
- Serial number (if applicable)
- Revision status (if applicable)
- Date of last replacement

The replacement history DB contains the following data items for each replacement task:

- Task ID
- Maintenance personal ID
- Timestamp
- Material equipment identifier
- Serial number - removed part
- Serial number - installed part
- Revision status - removed part
- Revision status - installed part
- Task comment

For each replacement task, the component appends an individual task record to the replacement history DB. The desired entry within the inventory database is updated to ensure the current status of the DB content. Finishing the documentation process, the system will send a task notification to a recipient, predefined in a global configuration file.

Additionally, the frontend provides the possibility for generating spare part replacement journals and current inventory status reports on request.

4. Design Phase

Within this phase, a detailed design is derived from the underlying mitigation concept, satisfying the linked requirements. The developed design approaches specify the implementation of the mitigation realization. For each mitigation, the design is composed of an initial tool research, a software architecture description and a final functional specification. During tool research phase, free available software solutions are evaluated regarding suitability for integration within the CM system. Based on the selected framework, the software architecture design illustrates the integration

structure incl. component interface interaction. The functional specification describes each component and its APIs in detail as documentation for the subsequent implementation phase.

4.1. Comprehensive Development Process

The following sections are related to the mitigation concept MIT4 & MIT6 - Comprehensive Development Process, described in chapter 3.1.3.

4.1.1. Tool Research

Release Management System
The release management system provides functionality for automatic continuous integration, deployment, and delivery of release packages.

1. **Jenkins**
   Jenkins[22] is one of the most popular open source build and release management tools, available for a wide range of operating system environments. It implements a server-client approach enabling flexible distributed build agent infrastructures. The framework is easily extendable with custom written or third party plug-ins, and it supports build scheduling based on the cron expression. Jenkins provides seamless integration of major version control and ticket system solutions. Notification services are included by default. The toolset is freely available and open source licensed under MIT[23].

2. **GoCD**
   GoCD[24] implements a free, open source continuous integration and build system, available for Windows, Linux and OSX platforms. It is licensed under the Apache License Version 2.0[25]. The complexity of the framework is rather simple, which supports fast time to initial function, but bears the risk of limitations in later realization phases. Its server-agent infrastructure, combinable with container based environments, enables flexible distribution approaches. The system is usually installed on premise, but cloud solutions are possible via third party service providers, too. The tool offers some interesting features like build comparison, automatic notification service and graphical dashboard frontend user interface. The tool documentation is of poor quality.

3. **GitLab**
   GitLab[26] is a free and open source tool suite, providing release management, issue tracking and version control within one solution package. It is available as offline version, installable on premise or on distributed cloud based infrastructures. The release management system provides a seamless link to the

internal available version control and issue tracking system if used in combination. GitLab is widely used in the software development sector as an all-in-one solution, providing a rich feature set like user authentication or analytic reporting. GitLab is licensed under MIT Expat[23].

4. **Atlassian Bamboo**
   Bamboo[27] is a proprietary solution, provided by Atlassian, usable as cloud service or local installation. Besides its core functionality as release management system, Bamboo provides distributed build environments integrated in containerized solutions, parallel builds and automatic triggering of builds, based on detected changes or time schedules. It best integrates with other tools of the Atlassian software suite. Release life cycle status reporting is available via web based frontend or file report generation. Notification mail service is fully included. The license model grants a trial period for 30 days.

Table 3: Comparison of Release Management System solutions

	Jenkins	**GoCD**	**GitLab**	**Atlassian Bamboo**
Availability	+ free	+ free	+ free	- 30 days trial
Infrastructure	+ on premise, cloud	+ on premise, cloud	+ on premise, cloud	+ on premise, cloud
Architecture	+ client-server	+ server-agent	+ distributed builds	+ distributed
UI/UX	**o**	+	+	+
Flexibility	+ plug in concept	- limited	**o**	**o**
Support	+ community	- poor support	+ community	**o** proprietary
Feature set	+ extendable	- limited	+ rich feature set	**o** fixed feature set
License	+ MIT	+ Apache 2.0	+ MIT Expat	- proprietary

Ticket System The ticket system implements functionality supporting a change request based development processes. Each change request is represented by a ticket within the system, equipped with a process aligned life cycle. The ticket system is aiming towards the following objectives: Release planning, development progress monitoring, link between version control and build system.

1. **Atlassian Jira**
   Jira [28], another component from the Atlassian tool suite, implements a state-of-the-art ticket system. A free license of the proprietary product is available, but limited to max. 10 users. As all Atlassian products, Jira provides an intuitive user interface (UI) and sophisticated user experience (UX).

2. **GitLab**
   As described above, the GitLab suite implements a ticket system called *issue tracking*.

3. **Redmine**
   Redmine [29] is a free and open source solution, implementing web based cross-platform support. It is licensed under GNU General Public License v2 (GPL)

[30]. The toolset has been available for a long time, hence no longer reflecting the current state of the art. An email notification service and seamless link to various version control solutions are integrated within the package.

Table 4: Comparison of Ticket System solutions

	Atlassian Jira	**GitLab**	**Redmine**
Availability	**o** free, max. 10 user	+ free	+ free
Infrastructure	+ on premise, cloud	+ on premise, cloud	+ on premise, cloud
UI/UX	+	+	**o**
Flexibility	+ customizable	**o** limited	+ customizable
Support	**o** proprietary	+ community	**o** basic free support
Feature set	**o** fixed feature set	+ rich feature set	+ rich feature set
License	**-** proprietary	+ MIT Expat	+ GNU GPLv2

Version Control

1. **Apache Subversion**
   Apache subversion (SVN) [31] represents a free available and open source version control solution, implementing a centralized oriented single server approach. SVN was class-leading for many years, since it was first developed in the year 2000. The tool is licensed under Apache License 2.0 [25]. Due to the single server architecture, SVN should be considered deprecated.

2. **Git**
   Git [1] is a free and open source distributed version control solution, fulfilling the latest requirements concerning speed, efficiency and flexibility. Git was developed by Linus Torvalds[32] to eradicate the disadvantages of existing solutions available at that time. Git is licensed under GNU General Public License v2 (GPL) [30]. It is widely used in the software development sector and represents the current dominating version control solution. GitLab supports GIT as default version control system providing full integration functionality.

Table 5: Comparison of Version Control System solutions

	Apache Subversion	**Git**
Availability	+ free	+ free
Infrastructure	+ on premise, cloud	+ on premise, cloud
Architecture	**-** single server	+ distributed
UI/UX	+ optional	+ optional
Flexibility	**o**	+ extensions available
Support	+ community	+ community
Feature set	+	+
License	+ Apache 2.0	+ GNU GPLv2

Release DB

Integrated implementation within release management solution.

Ticket DB
Integrated implementation within ticket system solution.

Version Control DB
Integrated implementation within version control solution.

SMTP Mail Service
Integrated implementation within release management and ticket system solution.

Tool Research Evaluation
As conclusion of the described comprehensive tool research phase, GitLab proves to be the best suited solution for release management and ticket system implementation. Besides the result being the highest scoring solution within the criteria based comparison, the GitLab tool suite provides the ticket system and release management within one closely linked package. Further, GitLab is freely available, licensed under open source, provides high sophisticated and modern UI/UX and is widely distributed within the software development sector. Compared to GitLab, Jenkins provides higher level of flexibility, especially for future extensions. Nevertheless, the GitLab integrated release management system is used in the first stage of the project, migrating to Jenkins is a potential option for later enhancement. Git is the undisputed preferred solution for use as version control within the desired environment. GitLab easily integrates Git without additional integration effort providing seamless interoperability.

4.1.2. Architecture Description

The following chapter describes the software architecture design from an integration level perspective. The implementation is realized using already available software modules from the GitLab framework, including the core components as well as an integrated database and frontend web service. Hence, the integration is expected to be straight forward without significant issues. The integration design approach is closely aligned with the recommended default architecture of GitLab as shown in figure 15.

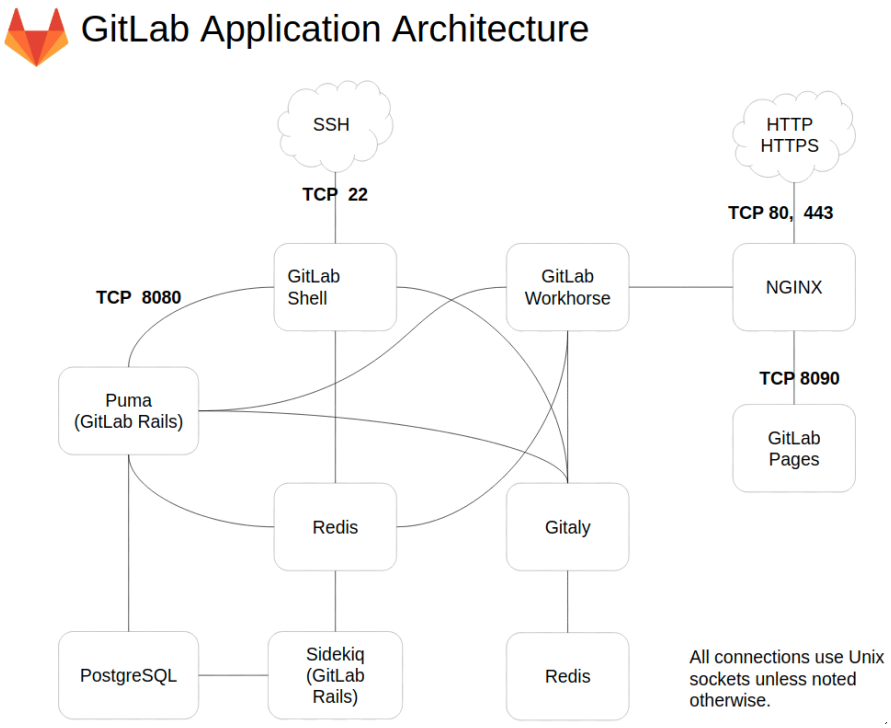


Figure 15: GitLab simplified integration architecture, from [33].

To realize the concept MIT4 & MIT6, described in chapter 3.1.3, the following components of GitLab, outlined in figure 15 are used:

1. **Puma (GitLab Rails)**
   Puma represents the main component of the GitLab tool framework, implementing the core functionality of the ticket system and the release management system. It includes the primary business logic and proper interface adapter for connecting surrounding software modules.

2. **PostgreSQL**
   A PostgreSQL database instance serves as persistent storage for history based records (tickets, releases), application metadata and user information (user accounts, roles, authentication information).

3. **Redis**
   Redis is a fast caching database solution, used to store frequently requested items like session data, temporary cache information or background job queues to improve speed and efficiency while reducing reply latency.

4. **Gitaly**
   Gitaly is responsible for communication between Puma and the GIT repository using remote procedure call (RPC) technology. It serves as an adapter for integrating the GIT version control implementation.

5. **SMTP Gateway**
   GitLab provides a SMTP gateway for sending email notifications, implemented inside Puma.

6. **NGINX webserver**
   NGINX, a popular open source webserver, serving the frontend implementation for user machine interaction. Restricting the transmission control protocol (TCP) access to port 443 (HTTPS only), combined with proper installed certificates, the communication layer is secured. The certificate based authentication and traffic data encryption permits separation of the server instance from the clients, even using non-secured network infrastructures.

7. **GitLab Shell**
   GitLab implements an internal secure shell (SSH) interface, providing SSH-based git sessions as an alternative to the described web based access to the system.

Installation Environment

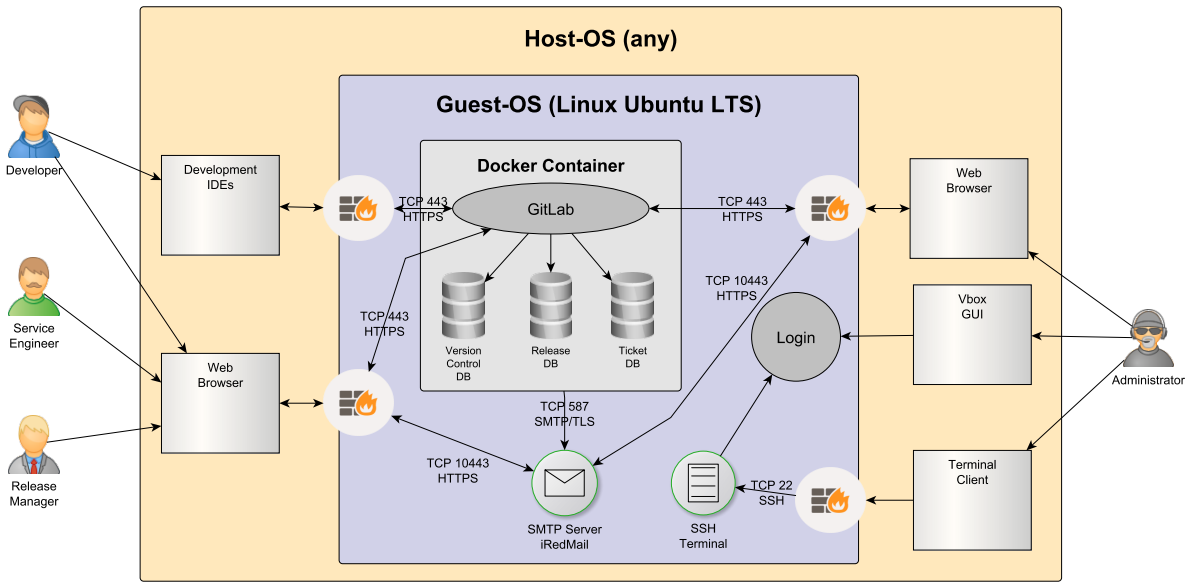


Figure 16: Installation environment architecture

The described tool integration architecture is installed within a virtualized environment as depicted in figure 16. Virtualization provides the possibility to migrate the CM server implementation on almost any available machine, including virtual cloud based server instances, allowing scalability as business grows (multiple production lines). Persistent storage database instances are integrated within the container

as well.
Further a high level of security can be achieved, following this strategy:

- **Environment encapsulation**
  Encapsulated and isolated from the host environment, the guest OS will not be negatively affected in case of a compromised host OS. Individual user authentication and access policies can be applied, limiting direct operating system (OS) access to administrative users only. The virtualization is realized two-staged: As first layer of encapsulation, Virtual Box is used with its guest OS sandboxing features. Canonical Ubuntu LTS[34] is installed inside the virtual machine, acting as guest OS. The CM GitLab server infrastructure is further encapsulated using Docker[35] realizing the second encapsulation layer.

- **Network access control**
  A firewall implementation filters access from external networks, permitting connections on selected ports for required services only. This directly minimizes the potential attack vectors, exposed to the untrusted host environment (e.g. Festo PC). Any further connections are limited to access from inside the GitLab container to prevent potential security related issues.

Finally, a locally installed SMTP server simulates the required mail server, usually available within enterprise networks. Users can gain access to mailboxes using the provided webmailer frontend of the service.

4.1.3. Functional Specification

Besides the wide spectrum of available features, provided by this CM infrastructure, the most essential functions are described here by means of specific use cases:

Release Management

- **Release Project Definition**
  The release manager creates a project within the CM, defining the scope of the release. Projects are typically closely linked to the hardware infrastructure, where the release artifacts are installed. The tasks, described below, are applied on project level.

- **Release Planning**
  The CM provides release planning capabilities with definition of release content by assigning change requests (issues) to specific target releases. Multiple target releases are processable in parallel, e.g. a major release for long term development, a minor release for midterm development and short terminated bug fix releases. GitLab represents target releases as release branches. Target

releases contain all required information, like a name, description and further documentation. The CM design covers planning on content level, based on decisions by the change control board (CCB), not considering business case planning regarding resources or budget. Project planning on business case level is provided by the controlling department of Yeskia Inc.

- **Progress Monitoring**
  GitLab provides the possibility to consider the progress of each assigned change request (issue). The progress is realized using a life cycle on issues as shown in figure 17.

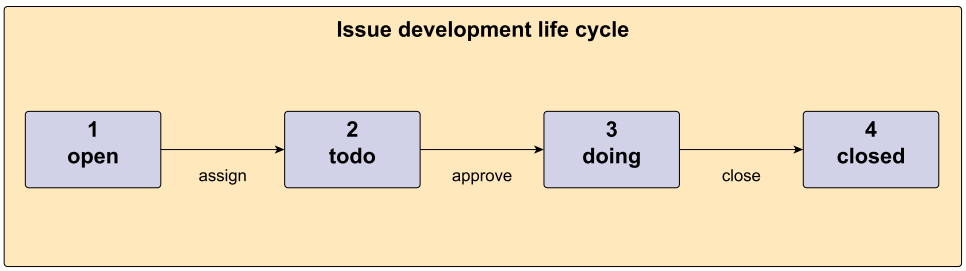


Figure 17: Issue development life cycle

1. `open`
   A new created issue, not assigned for implementing
2. `todo`
   Issue approved and assigned by the CCB, ready to implement
3. `doing`
   Issue in implementation phase
4. `closed`
   Issue closed by developer, ready to integrate into assigned release milestone

The life cycle is fully adaptable to custom requirements, e.g. extending with states for reviewing or testing stages.

- **Release Approval Process**
  For each potential release candidate, a linked milestone is created. Milestones represent release packages for the subsequent release validation process. In case of failing validation, a new successor milestone is created as base for the next release candidate loop iteration. The milestone passing the validation process represents the final content for the release freeze, published as delivery package.

- **Release History Tracking**
  All published releases are stored within the release database. GitLab provides views, listing all closed releases including information. Further, the database archives all artifacts for each release, extractable for roll-back installation, recovering prior states of the facilities. Release content comparison supports effective investigation of negative impacts related to release deployment.

Ticket System The ticket system is integrated within the GitLab framework as core functionality. Issues are created as tickets, representing change requests implementing a suitable life cycle as described in chapter 4.1.3. Issues are created by different actors due to various reasons:

- Development feature requests
- Development bugs
- Production feature requests
- Production malfunction notifications

Submitted tickets act as initial starting point for the release planning, described in chapter 4.1.3. Hence, the ticket system is closely linked to the release management.

Version Control Each defined project is connected to an individual git repository as backend interface to the version control system. The repository stores any kind of artifact related data (e.g. source code, application parameters, technical specifications, integration development environment (IDE) solutions). Any modification of data inside repositories for development purpose is linked to referenced issues. Transparency and traceability of code modifications is fully ensured. Version control supports effective investigation by providing history based difference comparison. Parallel check-out combined with merging check-in enables the required collaborative development workflow.

4.2. Detection and logging of unauthorized artifact changes

The following sections are related to the mitigation concept MIT3 - Detection and logging of unauthorized artifact changes, described in chapter 3.1.2.

4.2.1. Tool Research

As main programming language, JavaScript is chosen, executing the solution inside the node.js [36] runtime. JavaScript is widely used, simple to use, dynamically typed scripting language, offering comprehensive library support (e.g. GitLab, OPC-UA, Logging, ...). Availability of runtime implementations for various OS environments,

provides flexible portability. The adapter concept supports seamless adaption of libraries, written in different programming languages. Hence, available communication adapter implementations could easily be integrated within the solution (e.g. native C/C++ libraries). The GitLab environment is already implemented and installed with MIT4 & MIT6, providing a suitable and available solution for persistent history data tracking and mail notification service.

4.2.2. Architecture Description

The following chapter describes the overall software architecture design with scope on integration level. Figure 18 depicts the software component structure, used as template for the implementation phase.

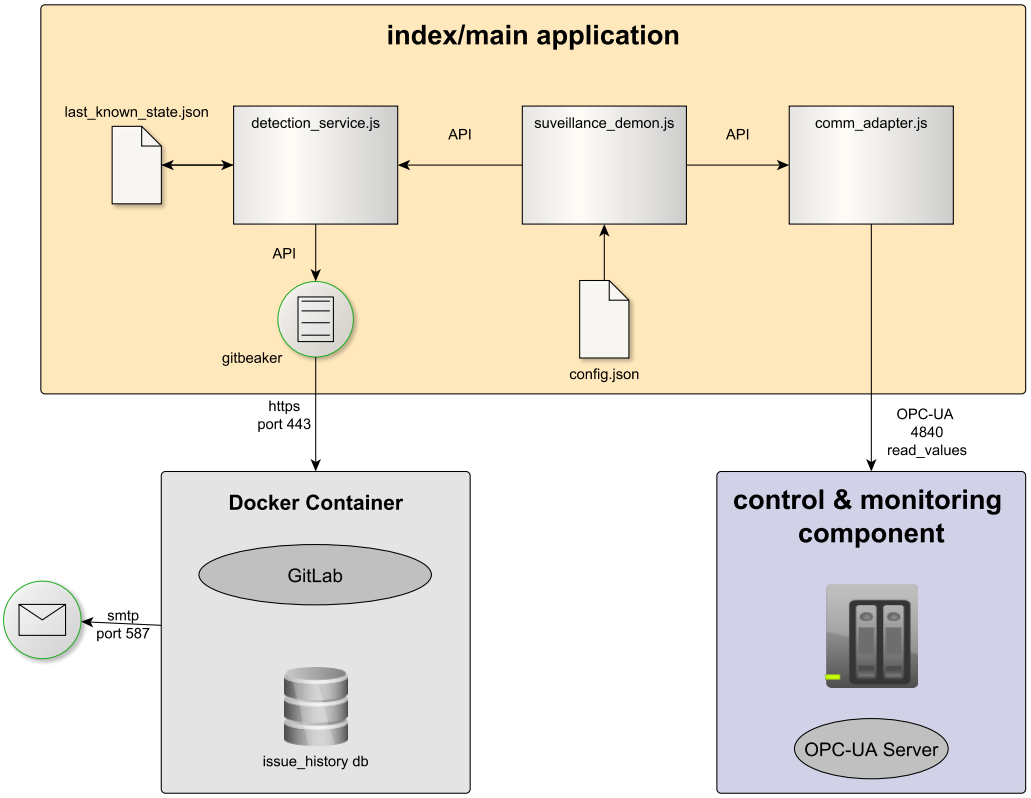


Figure 18: MIT3 - Architecture Design

Each software component is described regarding its architecture, API and functional behavior.

1. **Overall Architecture**
   The architecture of the mitigation is split into two sections, the main application and the GitLab service. The integration of all implemented SW components within the main application is structured in separate files to en-

able a modular approach for good maintainability and transparency. Executed with node.js runtime implementation, the connection to the GitLab service is realized using its provided HTTPS based API.

2. **Surveillance Deamon**
   The surveillance daemon represents the main component, controlling the process. The scheduler is realized using JavaScripts native method `setInterval()`. No additional components are required to implement the required functionality. The global service configuration is realized with a JSON [37] formatted persistent file, parsed at initialization phase of this component. All relevant configuration parameters are passed to sub-components during initialization. The functions of this component are called internally only, hence no external API is published. The execution context of the service is solely implemented within this component.

3. **Comm Adapter**
   The communication adapter represents the interconnection component between the surveillance service and its connected control & monitoring components of the production facility. It implements the specific communication protocols for acquiring the desired information from the HW components. Due to varying requirements for different protocol implementations, usage of suitable programming languages is supported. The API provides functions for requesting checksum values, representing the binary content of the controller artifacts. If not directly provided by the individual control & monitoring component interface, the comm adapter requires further implementation for calculating the desired checksum values.

4. **Change Detection Management Service**
   After acquiring representing checksum values from the control & monitoring component, it is passed to the change detection management service using internal API calls. Therefore, the component publishes the required function set as public interface. Persistent storage of the last known state table is realized using a JSON formatted file.

5. **GitLab**
   GitLab service is responsible for persistent history storage and export, as well as for email notification of detected changes. The provided issue system is used for both tasks. The interconnection to the GitLab service is realized calling its provided HTTPS based API, using an available and open source adapter library `gitbeaker` [38].

4.2.3. Functional Specification

At program initialization phase, the main routine of the application opens the global configuration file and initializes all required sub-components passing the related configuration parameters. The change detection service opens the file containing the last known checksum values for each monitored HW component. Once the system is successfully initialized, the surveillance daemon schedulers starts periodical triggering the acquisition of the artifact representing checksum values. Each control & monitoring component, internally represented with a unique component ID, is monitored regarding changes of the artifact content checksum values. In case of a detected value change, the change detection service updates the linked entry inside the JSON formatted file. Further an issue creation process and mail notification is initialized containing the following information:

- *Title*
  Stating that an artifact content of HW component (ID and name) has been changed
- *Description*
  Old and new artifact checksum value for verification

4.3. Implementation of computer aided spare part replacement process

The following sections are related to the mitigation concept MIT7 - Implementation of computer aided spare part replacement process, described in chapter 3.1.5.

4.3.1. Tool Research

As the main programming language, JavaScript is chosen for the frontend and the backend implementation, analogous to MIT3 4.2.1.

The GitLab environment is already implemented and installed with MIT4 & MIT6, providing a suitable and available solution for persistent history data tracking and mail notification service.

4.3.2. Architecture Description

The following chapter describes the overall software architecture design with scope on integration level. Figure 19 depicts the software component structure, used as template for the implementation phase.

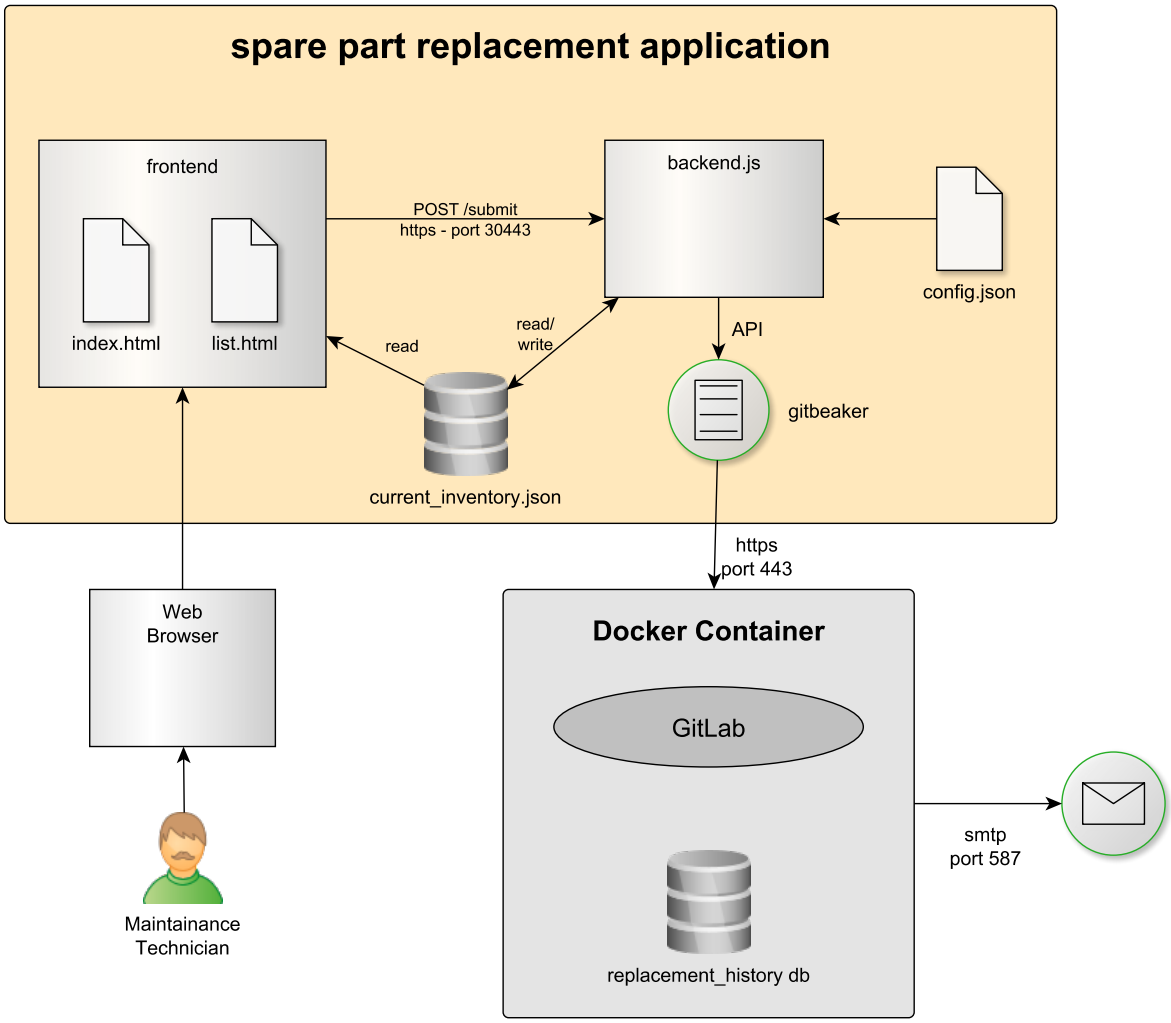


Figure 19: Spare part app simplified integration architecture

Each software component is described regarding its architecture, API and functional behavior.

1. **Overall Architecture**
   The architecture of the mitigation is split into two sections: the main application (i.e. frontend & backend), and the GitLab service. The integration of all implemented SW components within the main application is structured in separate files to enable a modular approach for good maintainability and transparency. The frontend is responsible for providing a guided user interface to the maintenance technician, while the backend is executed within the node.js runtime, and the connection to the GitLab service is realized using its provided HTTPS based API. Due to the web based approach, the service is accessible over internet protocol (IP) network infrastructure, using various device types (e.g. Laptop, Smartphone, ...).

2. **Frontend**
   The frontend implementation is split into two software components:
   a) `index.html`
      Implements the form for the replacement process, validating and submitting user input data to the backend.
   b) `list.html`
      Provides a table-based view of the complete BOM inventory, showing the current HW status of the facility.

3. **Backend**
   After validating and accepting the submitted user input data from the frontend, the backend updates the relevant spare part record in the current inventory database and finally creates a GitLab issue.

4. **Current Inventory Database**
   The database, representing the current status of all available HW components of the facility, is implemented within the file `current_inventory.json`. The implementation is prepared for future migration to large scale SQL database solutions.

5. **User Database**
   All users defined for accessing the service are listed within the file `users.json`. Besides the user name, the user ID of the GitLab service is defined here enabling automatic creation of GitLab issues.

6. **GitLab**
   GitLab service is responsible for persistent history storage and export, as well as for email notification of detected changes. The provided issue system is used for both tasks. The interconnection to the GitLab service is realized calling its provided HTTPS based API, using an available and open source adapter library `gitbeaker` [38] inside backend.js.

4.3.3. Functional Specification

At program initialization phase, the backend opens a global configuration file and initializes all required sub-components passing the related configuration parameters. Once the system is successfully initialized, it listens for incoming HTTPS connections on TCP port 30443 in order to accept user input data from the frontend. In case of an incoming submit, the backend updates the relevant record inside the current inventory database. Further an issue creation process and mail notification is initialized containing the following information:

- *Title*
  Stating that spare part (BMK) has been replaced
- *Description*
  Summarizes the input data from the user, combined with the record data of the replaced part.

The frontend form for replacement submission includes the following listed elements:

1. *Header*
   The header of the website contains the company name as well as a short description of the purpose of this site.
2. *Link to BOM inventory list view*
   A link, redirecting to the webview containing the list displaying the current inventory database content.
3. *User (mandatory)*
   Dropdown containing all available user accounts for selecting the technician processing the replacement process.
4. *Material equipment identifier (BOM) (mandatory)*
   Dropdown, filled with all available HW components of the facility, able to select for the replacement process. The elements are displayed with the unique BOM and the component name.
5. *Serial number - removed part (if applicable)*
   If applicable, the serial number of the replaced part
6. *Serial number - installed part (if applicable)*
   If applicable, the serial number of the installed part
7. *Revision status - removed part (if applicable)*
   If applicable, the revision status of the replaced part
8. *Revision status - installed part (if applicable)*
   If applicable, the revision status of the installed part
9. *Date of replacement (mandatory)*
   HTML5 datepicker element for selection of the replacement date. This date could differ from the current date.
10. *Task comment*
    A comment filled by the technician, explaining the purpose of the replacement or other additional notes.

11. *Submit button*
    The button to submit the filled form to the backend service for validation and processing.

The following enumeration describes the functional behavior of the application:

1. *Frontend form index.html initialization*
   Accessing the `index.html` start page of the spare part replacement service, the frontend automatically fills the field `date of replacement` with the current date. The user is able to change the date of replacement value, as the replacement could be processed already days before the submission.

2. *User selection*
   As the dropdown list only contains all users, defined within the user database file, the selection is fail-safe.

3. *HW component selection*
   The dropdown list is prefilled with all available HW components of the facility. After selecting one item, the frontend fills further fields (`Serial number - removed part`, `Revision status - removed part`) with available information regarding the component. The user verifies the suggested values against the real numbers of the replaced part. In case of differences, the values can be overwritten, which leads to a additional notice within the created GitLab issue description. This scenario is necessary in case of non-consistent replacement history (e.g. replacements without submission).

4. *Submit replacement documentation*
   After filling all desired fields, the process is finished by pressing the submit button, located at the bottom of the page. The data is passed to the backend service for validation and further processing.

Processing the passed data, the backend service is performing the following validation checks:

- *Replacement date*
  The date representing the replacement timestamp shall not be in the future.

- *Serial number available*
  In case a serial number is available for the selected component within the current inventory database, the field shall be treated as mandatory.

- *Serial number consistent*
  Compare the passed value for the replaced part with the available value within the current inventory database. In case of differences, the backend adds an additional notice within the created GitLab issue description for transparency.

- *Revision status available*
  In case a revision status is available for the selected component within the current inventory database, the field shall be treated as mandatory.

- *Revision status consistent*
  Compare the passed value for the replaced part with the available value within the current inventory database. In case of differences, the backend adds an additional notice within the created GitLab issue description for transparency.

5. Implementation Phase

This chapter describes the implementation of the mitigations addressing the highest priorities, based on the design phase. The implementation phase is split into two release candidates, where release candidate 1 is concerned with MIT4 & MIT6 - comprehensive development process, whereas release candidate 2 covers MIT3 - detection and logging of unauthorized artifact changes and MIT7 - implementation of computer aided spare part replacement process. This section primarily focuses on the implementation tasks of the server framework, rather than its usage (user manual). Refer to the appended training course material for user hands-on usage description.

5.1. Release Candidate 1 (RC1)

This chapter covers the implementation of RC1, realizing the mitigation MIT4 & MIT6 - comprehensive development process.

5.1.1. Implementation (RC1)

The necessary steps for the installation, implementation and configuration are outlined in the subsequent section, structured by its comprising SW components.

Installation of Ubuntu 20.04 LTS in Oracle VirtualBox

A Linux Ubuntu 20.04 LTS [34] operating system is installed in a Oracle VirtualBox [39] environment to serve as the central server infrastructure for further tool integration. The following settings have to be adjusted for the installation, in order to ensure adequate system performance:

- Set storage disc space allocation to 30GB (dynamically allocated)
- Set processor count to 2
- Set base memory to 5120MB
- Enable 3D acceleration in display settings

- Ensure Network Adapter 1 is set to NAT (for complete network structure description, refer to Chapter 5.1.1)

Creation of a certificate chain with self-signed certificate authority (CA) SSL certificates are required to provide authenticated and secured connections to the services of the CM server infrastructure. Despite the very limited network access of the education facility within the laboratory, securing all network traffic prepares the server installation for later migration to extended IT-infrastructures, utilizing unsecured network areas as well. A hierarchical certificate structure, according to IETF X.509 [40] enables the verification of all further created certificates by only installing the self-signed root CA public certificate on client machines. The OpenSSL [41] framework is used for generation. In a real world scenario, this emulation would be replaced by using an already existing public key infrastructure (PKI) [42] of the companies IT-infrastructure. The created CA structure is illustrated in figure 20, the required installation steps are listed subsequently.

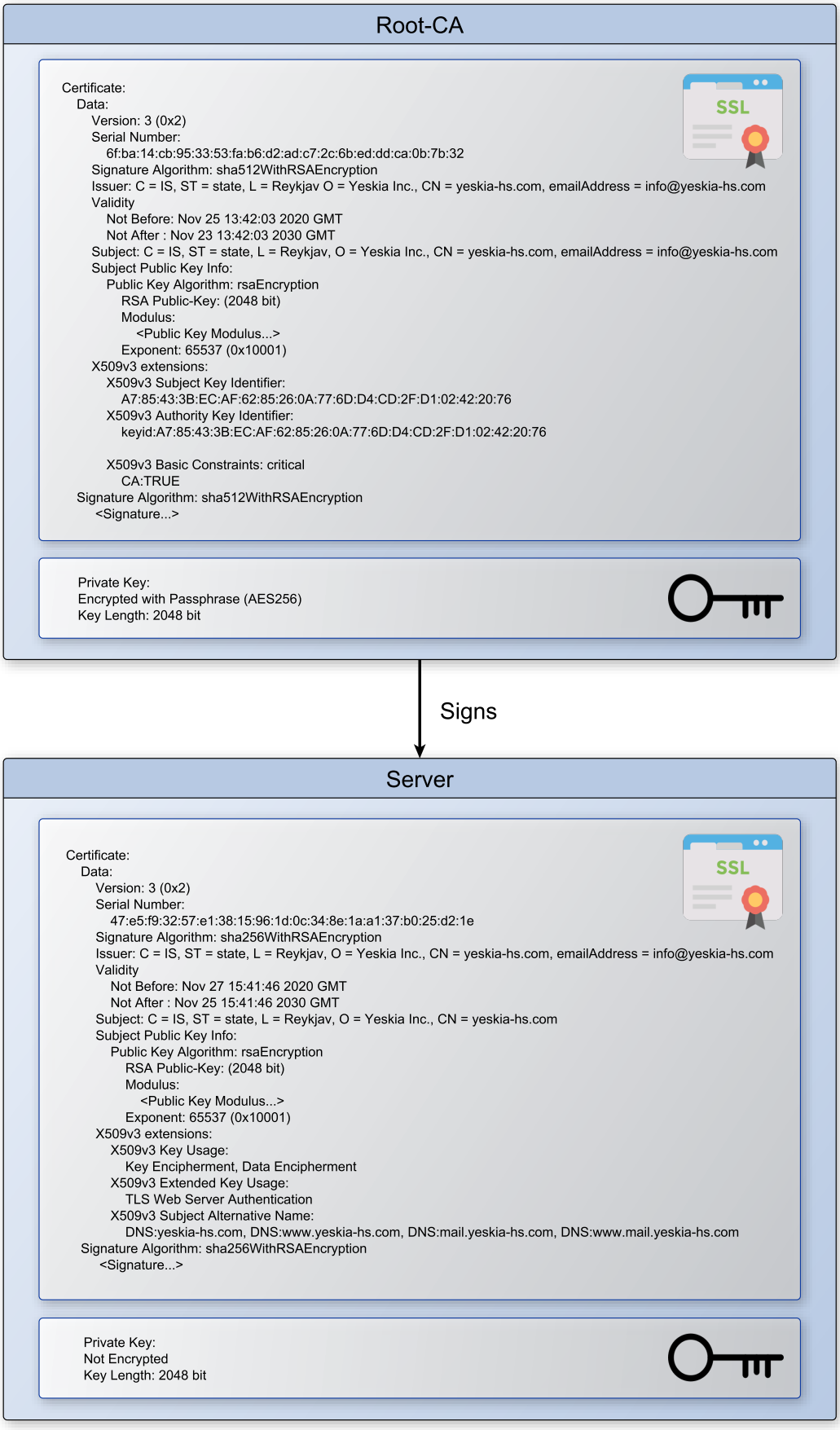


Figure 20: Root CA certificate chain

As first step, a root CA and its private key are created using the following commands:

1. `openssl genrsa -aes256 -out ca-key.pem 4096`
   The private key has a key length of 4096 bit. As the root key represents the secret with the highest criticality regarding confidentiality, it is additionally encrypted with AES256 algorithm to avoid unauthorized access in case of leakage.

2. `openssl req -x509 -new -nodes -extensions v3_ca -key ca-key.pem -days 3650 -out ca-root.pem -sha512`
   The public root CA certificate is self-signed with the root CA private key.

The public root CA certificate is installed on all relevant client machines (cert store of web browser, OS and git framework installation) to enable verification of all certificates, issued by the root CA. The issued server certificate with its private key is created and signed using the following steps:

1. create server certificate configuration file

Listing 1: server.cfg

```
[req]
distinguished_name = req_distinguished_name
req_extensions = v3_req
prompt = no
[req_distinguished_name]
C = IS
ST = state
L = Reykjav
O = Yeskia Inc.
CN = yeskia-hs.com
[v3_req]
keyUsage = keyEncipherment, dataEncipherment
extendedKeyUsage = serverAuth
subjectAltName = @alt_names
[alt_names]
DNS.1 = yeskia-hs.com
DNS.2 = www.yeskia-hs.com
DNS.3 = mail.yeskia-hs.com
DNS.4 = www.mail.yeskia-hs.com
```

2. `openssl genrsa -out server-key.pem 4096`
   The private key has a key length of 4096 bit. As this key is implicitly used

by webserver and mail services, and certificate revocation and replacement is achievable with little effort in case of leakage, the key is not encrypted.

3. `openssl req -new -out server.csr -key server-key.pem -config server.cfg`
   A certificate signing request (CSR) is generated, including all relevant information.

4. `openssl x509 -req -in server.csr -CA ca-root.pem -CAkey ca-key.pem -CAcreateserial -out server.pem -days 3650 -sha256`
   Finally, the CSR is signed by the root CA, using its secret private key. After completion of this step, the created server certificate is valid until revocation or expiration.

Table 6 summarize all resulting files of the CA generation process.

Table 6: CA - generated files

Filename	**Description**
ca-key.pem	private key of the root CA (PEM format)
server.cfg	configuration file for server CSR generation
server-key.pem	private key of the server certificate (PEM format)
server.csr	server CSR (PEM format)
server.pem	server certificate incl. public key (PEM format)

Install docker

Docker [35] is a platform enabling OS level virtualization to deliver software in so-called containers. Besides addressing security related topics, Docker provides the option for easy installation of preconfigured solutions. Additionally, Docker Compose extension [43] is installed to support defining and running containers easily configurable utilizing YAML files.

Install and configure GitLab using docker

On top of the Docker runtime environment, GitLab Community Edition [26] is installed as container using Docker Compose with a YAML configuration file printed in listing 2.

Listing 2: gitlab_compose.yaml

```
1  web:
2  image: 'gitlab/gitlab-ce:latest'
3  restart: always
4  hostname: 'yeskia-hs.com'
5  environment:
6  GITLAB_OMNIBUS_CONFIG: |
7  external_url 'https://yeskia-hs.com'
8  letsencrypt['enable'] = false
9  gitlab_rails['gitlab_email_enabled'] = true
```

```
10      gitlab_rails['gitlab_email_from'] = 'cm@yeskia-hs.com'
11      gitlab_rails['gitlab_email_display_name'] = 'Configuration
            Management'
12      gitlab_rails['gitlab_email_reply_to'] = 'noreply@yeskia-hs.com'
13      gitlab_rails['gitlab_email_subject_suffix'] = ''
14      gitlab_rails['smtp_enable'] = true
15      gitlab_rails['smtp_address'] = "mail.yeskia-hs.com"
16      gitlab_rails['smtp_port'] = 587
17      gitlab_rails['smtp_user_name'] = "cm@yeskia-hs.com"
18      gitlab_rails['smtp_password'] = "Passw0rd!"
19      gitlab_rails['smtp_domain'] = "yeskia-hs.com"
20      gitlab_rails['smtp_authentication'] = "login"
21      gitlab_rails['smtp_enable_starttls_auto'] = true
22      gitlab_rails['smtp_openssl_verify_mode'] = 'peer'
23      gitlab_rails['smtp_ca_file'] = '/etc/gitlab/ssl/ca-root.crt'
24      registry_nginx['ssl_certificate'] = "/etc/gitlab/ssl/yeskia-hs.
            com.crt"
25      registry_nginx['ssl_certificate_key'] = "/etc/gitlab/ssl/yeskia
            -hs.com.key"
26      ports:
27      - '443:443'
28      - '22:22'
29      volumes:
30      - '/srv/gitlab/config:/etc/gitlab'
31      - '/srv/gitlab/logs:/var/log/gitlab'
32      - '/srv/gitlab/data:/var/opt/gitlab'
33      extra_hosts:
34      - "mail.yeskia-hs.com:172.17.0.1"
```

The installation is carried out aligned to the extensive documentation available on the GitLab website [44]. External access to the GitLab server is secured using the generated server certificate to provide the identification of the server identity and encryption of all traffic, implementing transport layer security (TLS). For sending notification emails, the GitLab configuration is extended with custom SMTP settings. Web access and connections to the SMTP server is limited to secured communication channels (TLS) only.

Install and configure iRedMail mail server

iRedMail [45], an open source mail server, is installed to simulate a real company email infrastructure. Authentication and traffic encryption is enabled with installation of the server certificate within the NGINX webserver [46] and the provided SMTP/internet message access protocol (IMAP) server access. Relevant iRedMail users are created to grant access to the related mailboxes via the webmailer frontend.

Network infrastructure setup

As outlined in figure 21, the network structure is split into three separated, isolated IP based subnets:

1. Host-PC-Festo
   Windows host PC for development purpose and production control, additionally acting as host machine for the virtual CM server infrastructure.

2. Guest-OS-Ubuntu
   Virtual sandbox environment for the complete virtual CM server infrastructure, hosting all CM relevant services.

3. Docker GitLab Container
   Container based virtual sandbox, hosting the GitLab environment including its required database instances.

As the virtual server represents the emulated enterprise IT-infrastructure, the fully qualified domain names (FQDNs) `yeskia-hs.com` and `mail.yeskia-hs.com` are configured as primary domain names. The subnets are connected via network address translation (NAT), forwarding requests on dedicated ports towards registered destination IP interfaces. The NAT implementation provides security by rejecting any other requests besides the desired service endpoints. Extended configuration of the OS hosts files supports access to the subnets using FQDN resolution. Depending on the source domain, the domain name server (DNS) directs to the appropriate destination subnet where the target server is located.

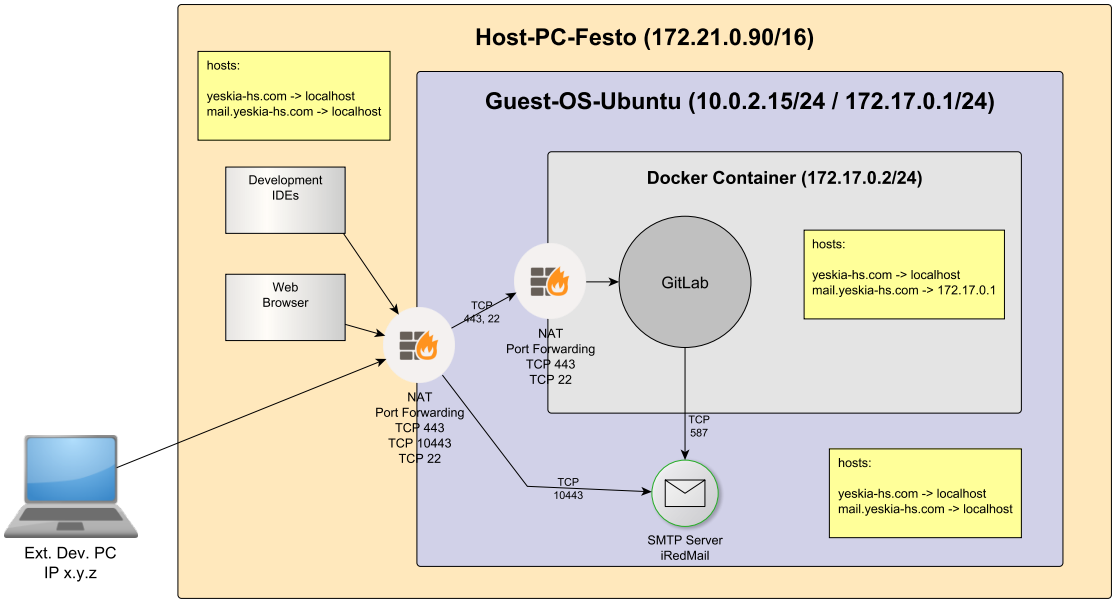


Figure 21: Network Structure

Due to lack of an enterprise DNS server within this project, the FQDN domain names are resolved locally by the respective OS name service, linking its hosts configuration file. The following routes describe the communication paths regarding their associated use cases:

- Ext.-Dev.-PC ⇒ Host-PC-Festo
  Access attempts from external located PCs are routed through the Festo host PC. Hence the external clients establish IP connections with the Festo PC IP address as destination. Once connected to the Festo PC, the routing mechanism of Windows is treating them analogous to local attempts for further processing.

- Host-PC-Festo ⇒ Guest-OS-Ubuntu
  Oracle VirtualBox provides port forwarding rules applied to NAT configured virtual interfaces. Searching for a listening IP port within the host environment, the NAT service retargets the requests to the Guest-OS in case of matching destination ports. Further processing is handled inside the virtual machine (VM).

- Guest-OS-Ubuntu ⇒ iRedMail server
  For incoming connection attempts, the Linux forwarding table distinguishes between access to the iRedMail webserver and the GitLab servers inside the Docker container. The selection is based on the destination port of the connection attempt:
  - 443, 22: forwarded towards the GitLab Docker container
  - 10443: routed to the local available iRedMail NGINX webserver instance

- Guest-OS-Ubuntu ⇒ Docker Container (GitLab)
  A second stage NAT layer is responsible for finally routing requests on port 443 and 22 into the GitLab Docker container subnet, reaching the desired services.

- Docker container (GitLab) ⇒ iRedMail server
  To provide container internal services the possibility to establish links to services running on guest OS level, backtrace routes are registered within the Docker environment. Any access to the domain name `mail.yeskia-hs.com` from services inside the container are forwarded to the guest OS subnet (iRedMail server access from GitLab).

Project initialization setup

Based upon the described server infrastructure installation, each project requires an initial setup for further usage. The following list outlines the project creation and configuration steps:

1. **User management - GitLab**
   As first step, user accounts for all relevant users are created within the GitLab user management system using the GitLab webadmin frontend. In addition to the user name, account contains additional information regarding the user.

2. **Group definition - GitLab**
   GitLab defines working groups, which contain references to created projects. Access to referenced projects is granted for member users of the group.

3. **Project creation - GitLab**
   Create a new project with the ability for importing an already existing git repository or to automatically generate a new, empty repository, directly linked to the project. Projects are assigned to defined groups.

4. **Project user role assignment - GitLab**
   The granted access level and correlating permissions are defined by assigning predefined roles to user accounts, individual configurable for each project.

5. **User management - iRedMail**
   For each user, a dedicated mailbox instance with a linked email address is created using the iRedMail administration web frontend.

5.1.2. Integration Test (RC1)

After successful integration of all required software components within the CM server infrastructure environment, the integration test procedure is executed according to the test specification `CM_integration_test_report` in appendix C. The test report documents the result of the test procedure, each test step is listed with detailed information regarding execution and individual result. The test documentation covers all required information for test reproducibility. Due to the seamless integration of the GitLab docker environment and standard interfaces toward the mail server, the test showed no irregularities.

The test report documentation is structured as described below:

1. **Report header**
   - *Report ID* - unique ID of the document
   - *Date* - date of test execution
   - *Tester* - person or group conducting the test execution
   - *Result* - overall test result
   - *Object under test (OUT)* - object to be examined
   - *Test environment* - test infrastructure and installation

- *Test result summary* - result summary, condensed for major test steps

2. **Test Steps**
   - *Test step headline* - unique ID and name of the major test step
   - *Minor test steps* - ID, instruction and excepted result for each minor test step
   - *minor test step result* - result for each minor test step

5.2. Release Candidate 2 (RC2)

This chapter covers the implementation of RC2, realizing the mitigation MIT3 - detection and logging of unauthorized artifact changes and MIT7 - implementation of computer aided spare part replacement process.

5.2.1. MIT 3 - Implementation (RC2)

The application code is written as outlined in the design architecture schematic. Functionality is fully implemented with the resulting solution. An online available GitLab cloud account is used for stubbed implementation during the development phase to minimize the time demand requiring the laboratory facility infrastructure. As communication with the real OPC-UA server is not provided in this environment, the different use cases of request replies are emulated within a stubbed communication adapter SW component. The service is executed within a node.js runtime environment, automatically started at OS boot time, infinitely monitoring for integrity checksum changes as background daemon. All required configuration parameters are defined inside the global configuration file `config.json` listed in figure 22.

```
{
  "GITLAB_API_HOST": "https://yeskia-hs.com:20443",
  "GITLAB_API_TOKEN": "8pZN_xaDPGesi3fAKU-_",
  "GITLAB_PROJECT_ID": "4",
  "GITLAB_USER_ID": "4",

  "OPC_ENDPOINT": "opc.tcp://172.21.1.1:4840",

  "SURVEILLANCE_ITERATION_INTERVAL_TIME": 60000
}
```

Figure 22: Configuration file - content

- `GITLAB_API_HOST`
  The URL of the GitLab server
- `GITLAB_API_TOKEN`
  A unique token, created by GitLab, granting access to the repository

- `GITLAB_PROJECT_ID`
  The ID of the gitlab project
- `GITLAB_USER_ID`
  The GitLab user account to access the repository and perform tasks within the ticket system
- `OPC_ENDPOINT`
  The URL of the monitored device within the facility infrastructure
- `SURVEILLANCE_ITERATION_INTERVAL_TIME`
  Periodic time span in milliseconds between monitoring loops

The scheduler, running withing the main thread loop of the surveillance application, periodically calls the function `iteration()`. A complete monitoring cycle is executed within this function call. As listed in figure 23 the main iteration loop executes the following steps:

1. `getChecksumFromDevice()`
   The current available checksum is requested using the proper communication adapter for the desired device. Error handling in case of communication failures are handled here.
2. `detectChange(checksumOfApp, checksumFromFile)`
   Pass the tuple, including the acquired checksum and the expected value from the persistent file `last_known_state.json`, calling the function `detectChange` for further processing.

```
/**
 * Surveillance iteration routine
 *
 * First, the comm adapter is asked for the checksum of the device application,
 * then the the last known checksum from a file is read and finnaly those values
 * are passed to the detection service for further processing.
 */
async function iteration() {
  logger.info('Starting new iteration...')

  const checksumOfApp = await getChecksumFromDevice()

  // facility down
  if (checksumOfApp === null) {
    logger.info('facility seems to be down...')
    return
  }

  const checksumFromFile = lastKnownState.component1.artifact1_checksum

  await detectChange(checksumOfApp, checksumFromFile)
}
```

Figure 23: `surveillance_deamon.js` - monitoring iteration loop function

The communication adapter exports its public function `getChecksumFromDevice`, listed in figure 24. Calling this function executes the subsequent steps.

1. `client.connect(endpointUrl)`
   Try to establish a TCP connection to the OPC-UA server of the desired HW device.

2. `client.createSession()`
   After successful connection, create a session for further data exchange on OPC-UA protocol layer.

3. `session.read(...)`
   Request the required checksum values, addressing the OPC-UA server with proper data.

Further code within the function transforms the checksum data into the harmonized format for passing back to the caller instance. Error handling for identification of communication failures is implemented here as well.

```
/**
 * Get checksum from device via OPC connection
 */
exports.getChecksumFromDevice = async function() {
  try {
    logger.info(`connecting via OPC to ${endpointUrl} ...`)

    await client.connect(endpointUrl)
    logger.info('connected via OPC')

    const session = await client.createSession()
    logger.info("OPC session created ")

    const dataValue3 = await session.read({
      nodeId: 'ns=3;s="plcChecksum".tmpNulled',
      attributeId: AttributeIds.Value
    });

    const checkumRaw = dataValue3.value.value
```

Figure 24: `comm_adapter.js` - get checksum from device

After checksum acquisition, the function `detectChange(checksumOfApp, checksumFromFile)` compares the passed checksum values for difference. In case of detected changes within the checksum values, a GitLab issue is created via the https based API, automatically triggering a notification email to the issue assigned recipient. Finally, the file `last_known_state.json` is updated with the new checksum value to suppress repeatedly notifications (false positives). Figure 25 lists the code of the described functionality.

```
/**
 * Routine contains logic for checksum change detection
 *
 * In case of detected change, a gitlab issue is created.
 */
exports.detectChange = async function(checksumOfApp, checksumFromFile) {
  let isEqual = true

  checksumOfApp = Array.from(checksumOfApp) // convert uint8 array to regular array

  for (let i = 0; i < 8; i++)  {
    if (checksumOfApp[i] !== checksumFromFile[i]) {
      isEqual = false
    }
  }

  if (isEqual) {
    logger.info('No changes detected')
    return
  }

  // --- checksum differs ---
  logger.info('>> Change detected! creating gitlab issue...')
```

Figure 25: `detection_service.js` - detect checksum changes

The complete inline commented source code is available within the project submission folder.

5.2.2. MIT 3 - Integration Test (RC2)

The implemented SW components are integrated as application, running within the development environment. To emulate the missing Yeskia Inc. GitLab server infrastructure, a freely available online could account is used. The communication adapter is temporarily replaced with a stub implementation, simulating the desired use cases of real communication scenarios. The integration test is processed as described in the test report `CM_integration_test_report_rc2` including the test results, available within appendix E. The structure of the test report is analogous to chapter 5.1.2.

5.2.3. MIT 7 - Implementation (RC2)

The application code is written as outlined in the design architecture schematic. Functionality is fully implemented with the resulting solution. An online available GitLab cloud account is used for stubbed implementation during the development phase to minimize the time demand requiring the laboratory facility infrastructure. The backend is executed within a node.js runtime environment, automatically started at OS boot time, serving the HTML files and listening for incoming user

input data. All required configuration parameters are defined inside the global configuration file `config.json` listed in figure 26.

```
{
  "GITLAB_API_HOST": "https://yeskia-hs.com:20443",
  "GITLAB_API_TOKEN": "8pZN_xaDPGesi3fAKU-_",
  "GITLAB_PROJECT_ID": "5"
}
```

Figure 26: Configuration file - content

- `GITLAB_API_HOST`
  The URL of the GitLab server
- `GITLAB_API_TOKEN`
  A unique token, created by GitLab, granting access to the repository
- `GITLAB_PROJECT_ID`
  The ID of the gitlab project

Frontend

In order to provide a modern, state-of-the-art user interface, the well known Bootstrap framework [47] is used to style web based content. The latest version of the hypertext markup language (HTML) markup language (HTML5) is used for future proof design.

Backend

The backend server is configured using HTTPS, providing secure and encrypted communication between the user machine and the webserver, as well as between the frontend implementation and the backend API. As listed in figure 27 the backend server is configured using the Yeskia Inc. TLS server certificate.

```
const SSL_PORT = 30443;
const key = fs.readFileSync('./server-key.pem', 'utf8');
const cert = fs.readFileSync('./server.pem', 'utf8');

const options = {
  key: key,
  cert: cert
};

https.createServer(options, app).listen(SSL_PORT);
```

Figure 27: Backend SSL setup

Figure 28 shows an extract of the submit listener implementation, including logic for finding the entity to update and serial number validation.

```
/**
 * Frontend form submit endpoint
 */
app.post('/submit', async (req, res) => {
  const dataFromClient = req.body
  const record = currentInventoryDB.find(r => r.bmk === dataFromClient.bmk)

  // clone old record
  const oldRecord = JSON.parse(JSON.stringify(record))

  // validation:
  if(record.serial_number && (!dataFromClient.sn || !dataFromClient.sn2)) {
    return res.status(500).send({
      error: `Please verify serial number, must not be empty for component: ${record.bmk}`
    });
  }
```

Figure 28: Backend submit listener

Figure 29 shows the current inventory database update routine. The inventory record is updated with the following data, submitted by the maintenance technician:

- Date of last replacement
- Serial number of installed part
- Revision status of installed part

Finally the file is written back to the hard disk for persistent storage.

```
/**
 * Update the inventory json file
 */
function updateInventory(record, currentInventoryDB, dataFromClient) {
  const newInventory = []

  record.date_of_last_replacement = dataFromClient.thedate
  record.serial_number = dataFromClient.sn2
  record.revision_status = dataFromClient.rs2

  // update the record and save inventory json file
  currentInventoryDB.forEach(rec => {
    if (rec.bmk === dataFromClient.bmk) {
      newInventory.push(record)
    }
    else {
      newInventory.push(rec)
    }
  })

  fs.writeFileSync('./current_inventory.json', JSON.stringify(newInventory, null, 2))
}
```

Figure 29: Backend update inventory routine

The complete inline commented source code is available within the project submission folder.

5.2.4. MIT 7 - Integration Test (RC2)

The implemented SW components are integrated as application, running within the development environment. To emulate the missing Yeskia Inc. GitLab server infrastructure, a freely available online could account is used. The integration test is processed as described in the test report `CM_integration_test_report_rc2.pdf` including the test results, available within appendix E.

6. Validation Phase

This section describes the validation steps of all implemented release candidates.

6.1. System Test Validation - RC1

Due to its inherent hardware autonomy, the performance of RC1 can be validated using the integration stage test and no further dedicated system integration test is necessary. After successful integration test, the framework can be installed at the customer facility for the subsequent customer integration test.

6.2. Customer Integration Test - RC1

The customer integration tests are conducted at the customer site on the production facility. Due to high test costs associated with partial or full production downtime during testing, a successful integration test and system integration test are a prerequisite. This ensures minimizing possible adverse effects during installation on the customer infrastructure.

The test is conducted analogous to the integration test phase, on the Festo-PC infrastructure.

Test iteration 1

As indicated in `CM_integration_test_report` iteration 1 in appendix D, the test execution failed at step 1 - *Login to GitLab web frontend*. The investigation discovered a duplicate use of the intended port 443 for the web server by another service on the Festo-PC. As this is a customer infrastructure specific behavior, this issue was not present in the previous integration test phases. An improved customer infrastructure specification is recommended for future projects, in order to avoid similar issues. This issue is solved by redirecting the initially specified port 443 to port 20443, which is available on the machine. A subsequent test iteration is necessary to evaluate all test steps within the test specification.

Test iteration 2

Within `CM_integration_test_report` iteration 2 in appendix C, the test failed again at later stage. In step 8 - *Create new merge request and new branch*, the generated branch was not visible. This behavior is traced back to well-known issues within the Microsoft Edge Browser, visualizing drop-down menu content. Firefox Browser is installed to solve the issue.

Test iteration 3

The implementation is again tested in a final iteration 3 passing all test steps, as listed in `CM_integration_test_report` iteration 3 in appendix C.

6.3. Customer Acceptance Test - RC1

The implementation is accepted by the customer after an extensive evaluation, which is documented in an acceptance protocol and signed by both involved parties. This stage is the conclusive step of the development of MIT4 & MIT6 and defines the state of the implementation for a partial delivery.

6.4. System Test Validation - RC2

Binary integrity surveillance

The integrated and pretested binary integrity surveillance service, refer to test report `CM_integration_test_report_rc2` in appendix E, is installed within the target representing system test environment. The emulating GitLab and communication adapter stubs are replaced with the productive variants. Additionally, to the already executed tests during the development integration test phase, the communication adapter with real communication scenarios and the interface to the productive GitLab server instance is the primary scope of this test stage. Due to extensive testing on the development environment following the mentioned stubbing strategy, no further issues occurred during the system test process. The procedure and results are documented within the test report `CM_system_integration_test_report_rc2` in appendix F.

Spare part replacement process

Analogous to the system test of MIT3 6.4, the release candidate is deployed to the system test environment. The stubbed GitLab account is replaced with the productive server infrastructure. Due to extensive testing on the development environment following the mentioned stubbing strategy, no further issues occurred during the system test process. The procedure and results are documented within the test report `CM_system_integration_test_report_rc2` in appendix F.

6.5. Customer Integration Test - RC2

As the system test and the customer facility are equivalent in our scenario of the virtual company Yeskia Inc., the customer integration test corresponds to already processed system test, described in chapter 6.4. As consequence, the execution of this identical test loop is skipped in this case.

6.6. Customer Acceptance Test - RC2

The implementation is accepted by the customer after an extensive evaluation, which is documented in an acceptance protocol and signed by both involved parties. This stage is the conclusive step of the development of MIT3 and MIT7 and defines the state of the implementation for the final delivery.

7. Training

The project concludes with a training course for selected employees of Yeskia Inc. The training covers usage, maintenance and administration of the developed CM system. The goal of this course is to enable efficient usage regarding different use cases.

7.1. Training Concept Development

The training concept is comprised as hands-on-training including practical live demonstration. Training duration is set to 45 minutes, covering the following topics in the outlined order:

1. Comprehensive Development Process (MIT4 & MIT6)
2. Detection of unauthorized artifact changes (MIT3)
3. Spare part replacement process (MIT7)

If applicable and beneficial, the training is carried out from the perspective of an administrator, a maintainer and a developer. Fellow students pose as employees of Yeskia Inc.

A teaching room, providing the following training facilities, is required:

- Projector
- Laptops
- Connection to the laboratory network

The training is delivered by an alternating presenter, supported by two training assistants guiding the trainees. The training material structure is aimed for reapplication in the future.

7.2. Training Content

The training material content is gathered from the actual solutions. It is easily comprehensible and does not require extensive prior knowledge of the used SW components. It is mainly composed of screenshots taken from the developed CM applications, including visually highlighted relevant areas.

The training course material is provided separately and not content of this report.

8. Outlook

Due to continuously rising complexity and connectivity of modern industrial infrastructures, the risks of production unavailability, non-conforming production and safety or security incidents increases proportional. This trend indicates the high demand concerning a suitable configuration management mitigating potential negative impacts.

The analysis and concept phase, including the extensive risk assessment, covers a wide range of potential mitigations. However, due to budget limitations, the design and implementation focuses on the four most important mitigations, which are fully realized and validated. The developed CM framework is prepared for deployment within a productive environment for professional use, representing a successful completion of this project.

An effective real world implementation exceeds the scope of this student project. Nevertheless, the process approach and the results of the analysis and concept phase provide a solid foundation for further development.

The real world scenario focus of the major project, combined with the multidisciplinary approach, results in growth on both personal and professional level.

Finally, the project group would like to express deep gratitude towards our supervisor Prof. Dr. rer. nat. Peter Richard and all fellow students for guidance and support.

9. Abbreviations

AOI automatic optical inspection

API application programming interface

BOM bill of material

CA certificate authority

CCB change control board

CM configuration management

DB database

DIP dual in-line package

DNS domain name server

EBIT earnings before interest and taxes

FQDN fully qualified domain name

FW Firmware

HMI human machine interface

HTML hypertext markup language

HW hardware

IDE integration development environment

IMAP internet message access protocol

IP internet protocol

KRL KUKA robotic language

MES manufacturing execution system

NAT network address translation

NCP non conforming production

OS operating system

OS operating system

PKI public key infrastructure

PLC programmable logic controller

RFID radio frequency identification

ROI return on investment

RPC remote procedure call

SCADA supervisory control and data

SIL safety integrity level

SL security level

SMTP simple mail transfer protocol

SRS Software Requirements Specification

SSH secure shell

SVN subversion

SW software

TCP transmission control protocol

TLS transport layer security

UI user interface

UX user experience

VM virtual machine

10. References

[1] Git, *Git*, 20.11.2020. [Online]. Available: https://git-scm.com/.

[2] Bitbucket, *Bitbucket | the git solution for professional teams*, 17.12.2020. [Online]. Available: https://bitbucket.org/product/.

[3] ISO, *Iso 31000:2018*, 17.12.2020. [Online]. Available: https://www.iso.org/standard/65694.html.

[4] IEEE, *Ieee recommended practice for software requirements specifications*, 1998. DOI: 10.1109/IEEESTD.1998.88286. [Online]. Available: https://ieeexplore.ieee.org/document/720574.

[5] H. Gall, "Functional safety iec 61508 / iec 61511 the impact to certification and the user.," in *AICCSA*, IEEE Computer Society, 2008, pp. 1027–1031, ISBN: 978-1-4244-1967-8. [Online]. Available: http://dblp.uni-trier.de/db/conf/aiccsa/aiccsa2008.html#Gall08.

[6] IEC, *Iec 62443*, 17.12.2020. [Online]. Available: https://webstore.iec.ch/publication/7029.

[7] ISO, *Din en iso 27001 - information security management*, Berlin, 7/2017.

[8] Festo Didactic SE, *Cp factory - cp-f-asrs32*, 6–2017.

[9] Festo Didactic SE, *Cp factory - cp-f-rass-kuka*, Festo Didactic SE, Ed., 2–2018.

[10] Festo Didactic SE, *Mes4 manual*, Festo Didactic SE, Ed., 11–2017.

[11] Siemens AG, *Simatic - et200sp open controller cpu1515-sp*, Siemens AG, Ed., 5–2017.

[12] Siemens AG, *Simatic - et200sp kommunikationsmodul io-link master: Gerätehandbuch*, Nürnberg, 11–2017.

[13] Siemens AG, *Simatic hmi tp700 comfort*, 27/11/2020. [Online]. Available: https://support.industry.siemens.com/cs/products/6av2124-0gc01-0ax0/simatic-hmi-tp700-comfort?pid=127118&mlfb=6AV2124-0GC01-0AX0&mfn=ps&lc=en-WW.

[14] D-Link, *D-link dap-1665 - user manual: Wireless ac1200 dual band access point*. [Online]. Available: ftp://ftp2.dlink.com/PRODUCTS/DAP-1665/REVA/DAP-1665_REVA_MANUAL_v1.10_EN.pdf.

[15] Festo AG & Co. KG, *Fibre optic unit soe4-fo-l - operating instructions*, Festo AG & Co. KG, Ed., 10/12/2020.

[16] MURR Elektronik, *Mico 2.6 electronic circuit protection - installation manual*, 10/12/2020. [Online]. Available: https://shop.murrelektronik.de/en/Electronics-in-the-Control-Cabinet/Intelligent-Power-Distribution/Modules/MICO-electronic-circuit-protection-2-CHANNELS-9000-41042-0100600.html.

[17] Hans Turck GmbH & Co. KG, *Tben-s2-2rfid-4dxp - compact rfid interface*, 11–2019.

[18] Festo AG & Co. KG, *Gdcp-cmmp manual: Gdcp-cmmp-m3-hw-en*, Esslingen, 10/12/2020. [Online]. Available: https://www.festo.com.cn/fox/net/SupportPortal/MobileDetails.aspx?documentId=380123&q=CMMP-AS&type=documentation.

[19] Festo AG & Co. KG, *Compact vision system manual - sbo...-q: Manual compact vision*, 2016. [Online]. Available: https://www.festo.com/net/da_dk/SupportPortal/default.aspx?cat=1995&tab=3&s=t#result.

[20] KUKA Deutschland GmbH, *Kr c4 compact - specification: Version v10*, Augsburg.

[21] Florian Wascher, *Mes4 communication with mes4: Communication between mes4 and plc*, Festo AG & Co. KG, Ed.

[22] Jenkins, *Jenkins*, 18.11.2020. [Online]. Available: https://www.jenkins.io/.

[23] MIT, *The mit license | open source initiative*, 18.11.2020. [Online]. Available: https://opensource.org/licenses/MIT.

[24] GoCD, *Open source continuous delivery and release automation server | gocd*, 7.07.2020. [Online]. Available: https://www.gocd.org/.

[25] Apache License, *Apache license, version 2.0*, 29.09.2020. [Online]. Available: https://www.apache.org/licenses/LICENSE-2.0.

[26] GitLab, *Gitlab.org / gitlab · gitlab*, 18.11.2020. [Online]. Available: https://gitlab.com/gitlab-org/gitlab.

[27] Atlassian, *Bamboo continuous integration and deployment build server*, 18.11.2020. [Online]. Available: https://www.atlassian.com/software/bamboo.

[28] Atlassian, *Jira | issue & project tracking software | atlassian*, 18.11.2020. [Online]. Available: https://www.atlassian.com/software/jira.

[29] Redmine, *Redmine*, 18.11.2020. [Online]. Available: https://www.redmine.org/.

[30] GPLv2, *Gnu general public license, version 2 - gnu-projekt - free software foundation*, 18.11.2020. [Online]. Available: http://www.gnu.org/licenses/old-licenses/gpl-2.0.html.

[31] Apache, *Apache subversion*, 20.11.2020. [Online]. Available: `https://subversion.apache.org/`.

[32] Encyclopedia Britannica, *Linus torvalds | finnish computer scientist*, 20.11.2020. [Online]. Available: `https://www.britannica.com/biography/Linus-Torvalds`.

[33] GitLab, *Gitlab integration architecture*, 23.11.2020.

[34] Canonical, *Focalfossa/releasenotes - ubuntu wiki*, 23.11.2020. [Online]. Available: `https://wiki.ubuntu.com/FocalFossa/ReleaseNotes?_ga=2.93303573.907068542.1606144533-95822307.1589614078`.

[35] Docker Documentation, *Docker overview*, 19.11.2020. [Online]. Available: `https://docs.docker.com/get-started/overview/`.

[36] Node.js, *Node.js*, 11.12.2020. [Online]. Available: `https://nodejs.org/en/`.

[37] json, *Json*, 5.12.2020. [Online]. Available: `https://www.json.org/json-de.html`.

[38] gitbeaker, *Jdalrymple/gitbeaker*, 11.12.2020. [Online]. Available: `https://github.com/jdalrymple/gitbeaker`.

[39] Oracle, *Oracle vm virtualbox*, 28/11/2020. [Online]. Available: `https://www.virtualbox.org/`.

[40] Tools.ietf.org, Rfcmarkup Version 1.129d On, *Rfc 5280 - internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile*, 22.11.2020. [Online]. Available: `https://tools.ietf.org/html/rfc5280`.

[41] OpenSSL Foundation, Inc., */index.html*, 30.11.2020. [Online]. Available: `https://www.openssl.org/`.

[42] BSI, *Public key infrastrukturen - public key infrastrukturen (pkien)*, 28/11/2020. [Online]. Available: `https://www.bsi.bund.de/DE/Themen/DigitaleGesellschaft/ElektronischeIdentitaeten/sicherPKI/sicherheitsmechanismenPKI.html`.

[43] Docker Compose, *Overview of docker compose*, 26.11.2020. [Online]. Available: `https://docs.docker.com/compose/`.

[44] GitLab Docker, *Gitlab docker images | gitlab*, 30.11.2020. [Online]. Available: `https://docs.gitlab.com/omnibus/docker/`.

[45] iRedMail, *Iredmail - free, open source mail server solution*, 8.05.2020. [Online]. Available: `https://www.iredmail.org/`.

[46] NGINX, *Nginx | high performance load balancer, web server, & reverse proxy*, 14.10.2020. [Online]. Available: `https://www.nginx.com/`.

[47] Twitter, *Bootstrap*, 16.12.2020. [Online]. Available: `https://getbootstrap.com/`.

A. Appendix: Project Scenario

Hochschule
Augsburg University of
Applied Sciences

Major Project 2020

Master Industrial Security

University of Applied Sciences Augsburg

Application Scenario

Table 1: Student authors

Student	Student ID
Alexander Holzmann	2081881
Markus Kamm	2060929
Michael Wager	2081894

1 Introduction

Yeskia Inc. constitutes a multinational enterprise in the field of telecommunications and a manufacturer of premium smartphone devices. With over 20 years of experience, the company managed to become one of the market leaders by utilising state-of-the-art technologies combined with low-cost production in Asia. Despite the great success story, Yeskia Inc. recently experienced an increasing adverse impact on production outcomes and revenues. An initial investigation, initiated by the management board, revealed that those incidents can be traced back to missing configuration management concerning the production domain. Past critical events like unintended machinery downtime, quality degradation of the final products and safety related incidents urge the board to react. Growing connectivity of the enterprise infrastructure poses additional challenges regarding cyber security. In order to resolve those issues and prevent future incidents, the board decided to task an external consulting firm with the development and implementation of an appropriate configuration management. The measures also include the establishment of a training course for Yeskia Inc.'s employees.

2 Company fact sheet

- Company: Yeskia Inc.
- Company Headquarters: Island
- Business sectors: broadband communication infrastructure, smart home IoT devices, premium smartphone devices
- Yearly overall revenue: 24,700,000,000 EUR
- Relevant sector for project assignment: production of smartphone devices
- Production: 10 production lines distributed across 3 sites located in China
- Operation: 24/7 - 365 days per year
- Product name: A-Phone
- Yearly revenue: 8,300,000,000 EUR
- Selling price: 899 EUR per device
- Manufacturing material costs: 249 EUR per device
- Units produced: 9,232,480 per year
- Basic production costs: 50,000,000 EUR per line per year (labour, production infrastructure)

B. Appendix: System and Software Requirements Specification

Hochschule
Augsburg University of
Applied Sciences

Major Project 2020

Master Industrial Security
University of Applied Sciences Augsburg

System and Software Requirements Specification

Table 1: Student authors

Student	Student ID
Alexander Holzmann	2081881
Markus Kamm	2060929
Michael Wager	2081894

1 Introduction

This document outlines the system and software requirements specification for the configuration management system for Yeskia Inc. It contains business requirements, security requirements and safety requirements.

Each Requirement contains:

- a unique identifier (ID) related to the corresponding risk
- a subject representing the component of the system
- a mitigation action title
- the requirements name
- a revision number for document version control
- a creation date
- a status indicating the approval state
- a description containing the actual requirement details
- a field for additional comments

2 Non-Functional Requirements

ID: 0.1.1		
Subject: General Requirement		
Category: Business/Security Requirement		
Mitigation: Ensure availability, safety and security of the plant		
Name: Limit side effects of CM		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The implementation of a Configuration management shall not have an adverse effect on availability, safety and security of the plant. Further, avoid any negative side effect regarding the production process.		
Comments:		

ID: 0.1.2		
Subject: General Requirement		
Category: Business/Security/Safety Requirement		
Mitigation: Minimize susceptibility to errors		
Name: Good usability		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: To reduce the probability of errors due to the usage of the configuration management, an intuitive usability shall be provided.		
Comments: Evaluate the target user group and adapt the human machine interface accordingly.		

ID: 0.1.3		
Subject: General Requirement		
Mitigation: Optimize implementation effort and cost factor		
Name: Tool selection		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Prefer already available state-of-the-art tools if applicable. If possible, free of charge software should be selected.		
Comments:		

3 Functional Requirements

ID: 0.1.4		
Subject: General Requirement		
Mitigation: Prevent unauthorized data access		
Name: Secured storage of data		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Sensitive data, acquired from the plant by the configuration management system, shall be stored securely. Unauthorized access to this data shall be prevented.		
Comments: Either use data encryption or proper access control.		

ID: 0.1.5		
Subject: General Requirement		
Mitigation: Prevent unauthorized data access		
Name: Secured communication channels		
Revision: 1	**Date:** 30.10.2020	**Status:** postponed
Description: Communication between the configuration management system and the plant components shall be implemented in a secured manner.		
Comments: Either use secure communication protocols or limit physical access to the network segments.		

ID: 1.1.1		
Subject: Festo MES4 System		
Mitigation: Automatic application version logging		
Name: Application version change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any artifact version change. Events shall only be triggered once each change.		
Comments: The artifact version could be acquired from the target device using appropriate communication protocols.		

ID: 1.1.2		
Subject: Festo MES4 System		
Mitigation: Automatic application version logging		
Name: Application version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The Application version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 1.1.3		
Subject: Festo MES4 System		
Mitigation: Automatic application version logging		
Name: Store records only on application change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall only contain records triggered by a change of the application version compared the current stored record.		
Comments: Avoid unnecessary redundant records.		

ID: 1.1.4		
Subject: Festo MES4 System		
Mitigation: Automatic application version logging		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 1.1.5		
Subject: Festo MES4 System		
Mitigation: Automatic application version logging		
Name: Notification in case of application version change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected application version change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 1.1.6		
Subject: Festo MES4 System		
Mitigation: Automatic application version logging		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 100 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 1.1.7		
Subject: Festo MES4 System		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Artifact binary integrity surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any changes within the artifact section. Changes shall only trigger once each change.		
Comments: The implementation using a state-of-the-art hashing algorithm with proper hemming distance is sufficient for detection reliability.		

ID: 1.1.8		
Subject: Festo MES4 System		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Artifact version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The application version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 1.1.9		
Subject: Festo MES4 System		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Logging of binary change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each binary integrity modification event shall trigger the system to automatically record the event persistently.		
Comments: Using a suitable ticket system could be used for logging.		

ID: 1.1.10		
Subject: Festo MES4 System		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 1.1.11		
Subject: Festo MES4 System		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Notification in case of binary change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected binary change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 1.1.12		
Subject: Festo MES4 System		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 50 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 1.1.13		
Subject: Festo MES4 System		
Mitigation: Comprehensive development process		
Name: Ticket system		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall implement a ticket based project management system in order to provide a guided development life cycle.		
Comments: The ticket system implementation should be directly linked with the version control and release management domains.		

ID: 1.1.14		
Subject: Festo MES4 System		
Mitigation: Comprehensive development process		
Name: Ticket system - Unique identifier		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each ticket shall be represented by a unique identifier.		
Comments: N/A		

ID: 1.1.15		
Subject: Festo MES4 System		
Mitigation: Comprehensive development process		
Name: Ticket system - User and role management		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The ticket system shall support a user and role oriented management. Users shall be linked to roles, the roles shall be configurable regarding authorization levels.		
Comments: N/A		

ID: 1.1.16		
Subject: Festo MES4 System		
Mitigation: Comprehensive development process		
Name: Ticket system - Process oriented ticket management		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The ticket system shall implement the well established enterprise development process. Following attributes are mandatory: • Lifecycle state • Owner • Severity/Priority • Type/Class • Target release		
Comments: N/A		

ID: 1.1.17		
Subject: Festo MES4 System		
Mitigation: Comprehensive development process		
Name: Transparent version control		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall implement a version control framework to trace detailed changes of the application content.		
Comments: The version control implementation should be directly linked with the ticket system and release management domains.		

ID: 1.1.18		
Subject: Festo MES4 System		
Mitigation: Comprehensive development process		
Name: Transparent version control - Longevity support		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Selection of version control system shall be conducted to ensure longevity support of at least 10 years.		
Comments: If applicable, internal maintenance of an open source solution should be preferred.		

ID: 1.1.19		
Subject: Festo MES4 System		
Mitigation: Comprehensive development process		
Name: Transparent version control - Artifact content differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Content changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 1.1.20		
Subject: Festo MES4 System		
Mitigation: Comprehensive development process		
Name: Roll out history of application releases		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall record any deployment of releases to the MES4 central SCADA system. Each record shall contain the following data: • Date • Version • Purpose		
Comments: The release management implementation should be directly linked with the ticket system and version control domains.		

ID: 1.1.21		
Subject: Festo MES4 System		
Mitigation: Comprehensive development process		
Name: Automatic release deployment roll back		
Revision: 1	**Date:** 30.10.2020	**Status:** postponed
Description: In case of a necessary roll back, the system shall provide an automated solution.		
Comments: Until integration of the automated system, the release deployment roll back shall be accomplished manually.		

ID: 2.1.1		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Automatic firmware version logging		
Name: FW version change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any artifact version change. Events shall only be triggered once each change.		
Comments: The artifact version could be acquired from the target device using appropriate communication protocols.		

ID: 2.1.2		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Automatic firmware version logging		
Name: FW version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The FW version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 2.1.3		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Automatic firmware version logging		
Name: Store records only on firmware change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall only contain records triggered by a change of the firmware version compared the current stored record.		
Comments: Avoid unnecessary redundant records.		

ID: 2.1.4		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Automatic firmware version logging		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 2.1.5		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Automatic firmware version logging		
Name: Notification in case of firmware version change event		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected firmware version change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 2.1.6		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Automatic firmware version logging		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 100 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 2.2.1		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Automatic application version logging		
Name: Application version change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any artifact version change. Events shall only be triggered once each change.		
Comments: The artifact version could be acquired from the target device using appropriate communication protocols.		

ID: 2.2.2		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Automatic application version logging		
Name: Application version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The Application version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 2.2.3		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Automatic application version logging		
Name: Store records only on application change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall only contain records triggered by a change of the application version compared the current stored record.		
Comments: Avoid unnecessary redundant records.		

ID: 2.2.4		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Automatic application version logging		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 2.2.5		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Automatic application version logging		
Name: Notification in case of application version change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected application version change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 2.2.6		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Automatic application version logging		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 150 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 2.3.1		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Artifact binary integrity surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any changes within the artifact section. Changes shall only trigger once each change.		
Comments: The implementation using a state-of-the-art hashing algorithm with proper hemming distance is sufficient for detection reliability.		

ID: 2.3.2		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Artifact version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The application version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 2.3.3		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Logging of binary change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each binary integrity modification event shall trigger the system to automatically record the event persistently.		
Comments: Using a suitable ticket system could be used for logging.		

ID: 2.3.4		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 2.3.5		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Notification in case of binary change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected binary change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 2.3.6		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 50 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 2.4.1		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Comprehensive development process		
Name: Ticket system		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall implement a ticket based project management system in order to provide a guided development life cycle.		
Comments: The ticket system implementation should be directly linked with the version control and release management domains.		

ID: 2.4.2		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Comprehensive development process		
Name: Ticket system - Unique identifier		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each ticket shall be represented by a unique identifier.		
Comments: N/A		

ID: 2.4.3		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Comprehensive development process		
Name: Ticket system - User and role management		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The ticket system shall support a user and role oriented management. Users shall be linked to roles, the roles shall be configurable regarding authorization levels.		
Comments: N/A		

ID: 2.4.4		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Comprehensive development process		
Name: Ticket system - Process oriented ticket management		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The ticket system shall implement the well established enterprise development process. Following attributes are mandatory: • Lifecycle state • Owner • Severity/Priority • Type/Class • Target release		
Comments: N/A		

ID: 2.4.5		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Comprehensive development process		
Name: Transparent version control		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall implement a version control framework to trace detailed changes of the application content.		
Comments: The version control implementation should be directly linked with the ticket system and release management domains.		

ID: 2.4.6		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Comprehensive development process		
Name: Transparent version control - Longevity support		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Selection of version control system shall be conducted to ensure longevity support of at least 10 years.		
Comments: If applicable, internal maintenance of a open source solution should be preferred.		

ID: 2.4.7		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Comprehensive development process		
Name: Transparent version control - Artifact content differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Content changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 2.4.8		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Comprehensive development process		
Name: Roll out history of application releases		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall record any deployment of releases to the MES4 central SCADA system. Each record shall contain the following data: • Date • Version • Purpose		
Comments: The release management implementation should be directly linked with the ticket system and version control domains.		

ID: 2.4.9		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Comprehensive development process		
Name: Automatic release deployment roll back		
Revision: 1	**Date:** 30.10.2020	**Status:** postponed
Description: In case of a necessary roll back, the system shall provide an automated solution.		
Comments: Until integration of the automated system, the release deployment roll back shall be accomplished manually.		

ID: 2.5.1		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Detection and logging of parameter changes		
Name: Parameter change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any changes of parameter values. Changes shall only trigger once each change.		
Comments: The implementation using a state-of-the-art hashing algorithm with proper hemming distance is sufficient for detection reliability.		

ID: 2.5.2		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Detection and logging of parameter changes		
Name: Parameter query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The Parameter values shall be queried at least 4 times per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 2.5.3		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Detection and logging of parameter changes		
Name: Logging of Parameter value adjustments		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each parameter adjustment shall trigger the system to automatically record the event persistently.		
Comments: Using a suitable ticket system could be used for logging.		

ID: 2.5.4		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Detection and logging of parameter changes		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 2.5.5		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Detection and logging of parameter changes		
Name: Notification in case of parameter adjustment events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected paremeter adjustment event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 2.5.6		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Detection and logging of parameter changes		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 1500 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 2.5.7		
Subject: Siemens Simatic S7 ET200SP CPU1515-SP		
Mitigation: Detection and logging of parameter changes		
Name: Parameter changes differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Parameter changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 3.1.1		
Subject: IO Link ET200SP IM155-6PN-HF		
Mitigation: Automatic firmware version logging		
Name: FW version change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any artifact version change. Events shall only be triggered once each change.		
Comments: The artifact version could be acquired from the target device using appropriate communication protocols.		

ID: 3.1.2		
Subject: IO Link ET200SP IM155-6PN-HF		
Mitigation: Automatic firmware version logging		
Name: FW version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The FW version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 3.1.3		
Subject: IO Link ET200SP IM155-6PN-HF		
Mitigation: Automatic firmware version logging		
Name: Store records only on firmware change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall only contain records triggered by a change of the firmware version compared the current stored record.		
Comments: Avoid unnecessary redundant records.		

ID: 3.1.4		
Subject: IO Link ET200SP IM155-6PN-HF		
Mitigation: Automatic firmware version logging		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 3.1.5		
Subject: IO Link ET200SP IM155-6PN-HF		
Mitigation: Automatic firmware version logging		
Name: Notification in case of firmware version change event		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected firmware version change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 3.1.6		
Subject: IO Link ET200SP IM155-6PN-HF		
Mitigation: Automatic firmware version logging		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 100 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 3.2.1		
Subject: IO Link ET200SP IM155-6PN-HF		
Mitigation: Detection and logging of configuration changes		
Name: Configuration change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any changes of configuration values. Changes shall only trigger once each change.		
Comments: The implementation using a state-of-the-art hashing algorithm with proper hemming distance is sufficient for detection reliability.		

ID: 3.2.2		
Subject: IO Link ET200SP IM155-6PN-HF		
Mitigation: Detection and logging of configuration changes		
Name: Logging of configuration value adjustments		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each configuration adjustment shall trigger the system to automatically record the event persistently.		
Comments: Using a suitable ticket system could be used for logging.		

ID: 3.2.3		
Subject: IO Link ET200SP IM155-6PN-HF		
Mitigation: Detection and logging of configuration changes		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 3.2.4		
Subject: IO Link ET200SP IM155-6PN-HF		
Mitigation: Detection and logging of configuration changes		
Name: Notification in case of configuration adjustment events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected configuration adjustment event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 3.2.5		
Subject: IO Link ET200SP IM155-6PN-HF		
Mitigation: Detection and logging of configuration changes		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 50 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 3.2.6		
Subject: IO Link ET200SP IM155-6PN-HF		
Mitigation: Detection and logging of configuration changes		
Name: Configuration changes differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Configuration changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 4.1.1		
Subject: HMI TP700		
Mitigation: Automatic firmware version logging		
Name: FW version change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any artifact version change. Events shall only be triggered once each change.		
Comments: The artifact version could be acquired from the target device using appropriate communication protocols.		

ID: 4.1.2		
Subject: HMI TP700		
Mitigation: Automatic firmware version logging		
Name: FW version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The FW version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 4.1.3		
Subject: HMI TP700		
Mitigation: Automatic firmware version logging		
Name: Store records only on firmware change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall only contain records triggered by a change of the firmware version compared the current stored record.		
Comments: Avoid unnecessary redundant records.		

ID: 4.1.4		
Subject: HMI TP700		
Mitigation: Automatic firmware version logging		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 4.1.5		
Subject: HMI TP700		
Mitigation: Automatic firmware version logging		
Name: Notification in case of firmware version change event		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected firmware version change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 4.1.6		
Subject: HMI TP700		
Mitigation: Automatic firmware version logging		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 100 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 4.2.1		
Subject: HMI TP700		
Mitigation: Automatic application version logging		
Name: Application version change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any artifact version change. Events shall only be triggered once each change.		
Comments: The artifact version could be acquired from the target device using appropriate communication protocols.		

ID: 4.2.2		
Subject: HMI TP700		
Mitigation: Automatic application version logging		
Name: Application version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The Application version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 4.2.3		
Subject: HMI TP700		
Mitigation: Automatic application version logging		
Name: Store records only on application change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall only contain records triggered by a change of the application version compared the current stored record.		
Comments: Avoid unnecessary redundant records.		

ID: 4.2.4		
Subject: HMI TP700		
Mitigation: Automatic application version logging		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 4.2.5		
Subject: HMI TP700		
Mitigation: Automatic application version logging		
Name: Notification in case of application version change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected application version change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 4.2.6		
Subject: HMI TP700		
Mitigation: Automatic application version logging		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 150 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 4.3.1		
Subject: HMI TP700		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Artifact binary integrity surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any changes within the artifact section. Changes shall only trigger once each change.		
Comments: The implementation using a state-of-the-art hashing algorithm with proper hemming distance is sufficient for detection reliability.		

ID: 4.3.2		
Subject: HMI TP700		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Artifact version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The application version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 4.3.3		
Subject: HMI TP700		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Logging of binary change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each binary integrity modification event shall trigger the system to automatically record the event persistently.		
Comments: Using a suitable ticket system could be used for logging.		

ID: 4.3.4		
Subject: HMI TP700		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 4.3.5		
Subject: HMI TP700		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Notification in case of binary change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected binary change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 4.3.6		
Subject: HMI TP700		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 50 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 4.4.1		
Subject: HMI TP700		
Mitigation: Comprehensive development process		
Name: Ticket system		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall implement a ticket based project management system in order to provide a guided development life cycle.		
Comments: The ticket system implementation should be directly linked with the version control and release management domains.		

ID: 4.4.2		
Subject: HMI TP700		
Mitigation: Comprehensive development process		
Name: Ticket system - Unique identifier		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each ticket shall be represented by a unique identifier.		
Comments: N/A		

ID: 4.4.3		
Subject: HMI TP700		
Mitigation: Comprehensive development process		
Name: Ticket system - User and role management		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The ticket system shall support a user and role oriented management. Users shall be linked to roles, the roles shall be configurable regarding authorization levels.		
Comments: N/A		

ID: 4.4.4		
Subject: HMI TP700		
Mitigation: Comprehensive development process		
Name: Ticket system - Process oriented ticket management		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The ticket system shall implement the well established enterprise development process. Following attributes are mandatory: • Lifecycle state • Owner • Severity/Priority • Type/Class • Target release		
Comments: N/A		

ID: 4.4.5		
Subject: HMI TP700		
Mitigation: Comprehensive development process		
Name: Transparent version control		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall implement a version control framework to trace detailed changes of the application content.		
Comments: The version control implementation should be directly linked with the ticket system and release management domains.		

ID: 4.4.6		
Subject: HMI TP700		
Mitigation: Comprehensive development process		
Name: Transparent version control - Longevity support		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Selection of version control system shall be conducted to ensure longevity support of at least 10 years.		
Comments: If applicable, internal maintenance of a open source solution should be preferred.		

ID: 4.4.7		
Subject: HMI TP700		
Mitigation: Comprehensive development process		
Name: Transparent version control - Artifact content differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Content changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 4.4.8		
Subject: HMI TP700		
Mitigation: Comprehensive development process		
Name: Roll out history of application releases		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall record any deployment of releases to the MES4 central SCADA system. Each record shall contain the following data: • Date • Version • Purpose		
Comments: The release management implementation should be directly linked with the ticket system and version control domains.		

ID: 4.4.9		
Subject: HMI TP700		
Mitigation: Comprehensive development process		
Name: Automatic release deployment roll back		
Revision: 1	**Date:** 30.10.2020	**Status:** postponed
Description: In case of a necessary roll back, the system shall provide an automated solution.		
Comments: Until integration of the automated system, the release deployment roll back shall be accomplished manually.		

ID: 4.5.1		
Subject: HMI TP700		
Mitigation: User management		
Name: User management modification surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any artifact version change. Events shall only be triggered once each change.		
Comments: The artifact version could be acquired from the target device using appropriate communication protocols.		

ID: 4.5.2		
Subject: HMI TP700		
Mitigation: User management		
Name: User management modification query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The user management information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 4.5.3		
Subject: HMI TP700		
Mitigation: User management		
Name: Store records only on user management modification events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall only contain records triggered by a change of the user management compared the current stored record.		
Comments: Avoid unnecessary redundant records.		

ID: 4.5.4		
Subject: HMI TP700		
Mitigation: User management		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 4.5.5		
Subject: HMI TP700		
Mitigation: User management		
Name: Notification in case of application user management modification events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected user management modification event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 4.5.6		
Subject: HMI TP700		
Mitigation: User management		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 150 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 4.5.7		
Subject: HMI TP700		
Mitigation: User management		
Name: User management differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Content changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 5.1.1		
Subject: D-Link DAP-1665		
Mitigation: Automatic firmware version logging		
Name: FW version change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any artifact version change. Events shall only be triggered once each change.		
Comments: The artifact version could be acquired from the target device using appropriate communication protocols.		

ID: 5.1.2		
Subject: D-Link DAP-1665		
Mitigation: Automatic firmware version logging		
Name: FW version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The FW version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 5.1.3		
Subject: D-Link DAP-1665		
Mitigation: Automatic firmware version logging		
Name: Store records only on firmware change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall only contain records triggered by a change of the firmware version compared the current stored record.		
Comments: Avoid unnecessary redundant records.		

ID: 5.1.4		
Subject: D-Link DAP-1665		
Mitigation: Automatic firmware version logging		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 5.1.5		
Subject: D-Link DAP-1665		
Mitigation: Automatic firmware version logging		
Name: Notification in case of firmware version change event		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected firmware version change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 5.1.6		
Subject: D-Link DAP-1665		
Mitigation: Automatic firmware version logging		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 100 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 5.2.1		
Subject: D-Link DAP-1665		
Mitigation: Detection and logging of configuration changes		
Name: Configuration change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any changes of configuration values. Changes shall only trigger once each change.		
Comments: The implementation using a state-of-the-art hashing algorithm with proper hemming distance is sufficient for detection reliability.		

ID: 5.2.2		
Subject: D-Link DAP-1665		
Mitigation: Detection and logging of configuration changes		
Name: Logging of configuration value adjustments		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each configuration adjustment shall trigger the system to automatically record the event persistently.		
Comments: Using a suitable ticket system could be used for logging.		

ID: 5.2.3		
Subject: D-Link DAP-1665		
Mitigation: Detection and logging of configuration changes		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 5.2.4		
Subject: D-Link DAP-1665		
Mitigation: Detection and logging of configuration changes		
Name: Notification in case of configuration adjustment events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected configuration adjustment event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 5.2.5		
Subject: D-Link DAP-1665		
Mitigation: Detection and logging of configuration changes		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 50 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 5.2.6		
Subject: D-Link DAP-1665		
Mitigation: Detection and logging of configuration changes		
Name: Configuration changes differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Configuration changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 6.1.1		
Subject: Fibreoptic SOE4-FO-L-HF2		
Mitigation: Detection and logging of configuration changes		
Name: Configuration change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any changes of configuration values. Changes shall only trigger once each change.		
Comments: The implementation using a state-of-the-art hashing algorithm with proper hemming distance is sufficient for detection reliability.		

ID: 6.1.2		
Subject: Fibreoptic SOE4-FO-L-HF2		
Mitigation: Detection and logging of configuration changes		
Name: Logging of configuration value adjustments		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each configuration adjustment shall trigger the system to automatically record the event persistently.		
Comments: Using a suitable ticket system could be used for logging.		

ID: 6.1.3		
Subject: Fibreoptic SOE4-FO-L-HF2		
Mitigation: Detection and logging of configuration changes		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 6.1.4		
Subject: Fibreoptic SOE4-FO-L-HF2		
Mitigation: Detection and logging of configuration changes		
Name: Notification in case of configuration adjustment events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected configuration adjustment event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 6.1.5		
Subject: Fibreoptic SOE4-FO-L-HF2		
Mitigation: Detection and logging of configuration changes		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 50 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 6.1.6		
Subject: Fibreoptic SOE4-FO-L-HF2		
Mitigation: Detection and logging of configuration changes		
Name: Configuration changes differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Configuration changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 7.1.1		
Subject: DriveMotor Control M-MZ-4-30		
Mitigation: Detection and logging of configuration changes		
Name: Configuration change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any changes of configuration values. Changes shall only trigger once each change.		
Comments: The implementation using a state-of-the-art hashing algorithm with proper hemming distance is sufficient for detection reliability.		

ID: 7.1.2		
Subject: DriveMotor Control M-MZ-4-30		
Mitigation: Detection and logging of configuration changes		
Name: Logging of configuration value adjustments		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each configuration adjustment shall trigger the system to automatically record the event persistently.		
Comments: Using a suitable ticket system could be used for logging.		

ID: 7.1.3		
Subject: DriveMotor Control M-MZ-4-30		
Mitigation: Detection and logging of configuration changes		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 7.1.4		
Subject: DriveMotor Control M-MZ-4-30		
Mitigation: Detection and logging of configuration changes		
Name: Notification in case of configuration adjustment events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected configuration adjustment event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 7.1.5		
Subject: DriveMotor Control M-MZ-4-30		
Mitigation: Detection and logging of configuration changes		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 50 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 7.1.6		
Subject: DriveMotor Control M-MZ-4-30		
Mitigation: Detection and logging of configuration changes		
Name: Configuration changes differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Configuration changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 8.1.1		
Subject: MURR Mico 2.6		
Mitigation: Detection and logging of configuration changes		
Name: Configuration change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any changes of configuration values. Changes shall only trigger once each change.		
Comments: The implementation using a state-of-the-art hashing algorithm with proper hemming distance is sufficient for detection reliability.		

ID: 8.1.2		
Subject: MURR Mico 2.6		
Mitigation: Detection and logging of configuration changes		
Name: Logging of configuration value adjustments		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each configuration adjustment shall trigger the system to automatically record the event persistently.		
Comments: Using a suitable ticket system could be used for logging.		

ID: 8.1.3		
Subject: MURR Mico 2.6		
Mitigation: Detection and logging of configuration changes		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 8.1.4		
Subject: MURR Mico 2.6		
Mitigation: Detection and logging of configuration changes		
Name: Notification in case of configuration adjustment events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected configuration adjustment event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 8.1.5		
Subject: MURR Mico 2.6		
Mitigation: Detection and logging of configuration changes		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 50 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 8.1.6		
Subject: MURR Mico 2.6		
Mitigation: Detection and logging of configuration changes		
Name: Configuration changes differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Configuration changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 9.1.1		
Subject: TURCK RFID Hub TBEN-S2-2RFID-4DXP		
Mitigation: Automatic firmware version logging		
Name: FW version change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any artifact version change. Events shall only be triggered once each change.		
Comments: The artifact version could be acquired from the target device using appropriate communication protocols.		

ID: 9.1.2		
Subject: TURCK RFID Hub TBEN-S2-2RFID-4DXP		
Mitigation: Automatic firmware version logging		
Name: FW version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The FW version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 9.1.3		
Subject: TURCK RFID Hub TBEN-S2-2RFID-4DXP		
Mitigation: Automatic firmware version logging		
Name: Store records only on firmware change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall only contain records triggered by a change of the firmware version compared the current stored record.		
Comments: Avoid unnecessary redundant records.		

ID: 9.1.4		
Subject: TURCK RFID Hub TBEN-S2-2RFID-4DXP		
Mitigation: Automatic firmware version logging		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 9.1.5		
Subject: TURCK RFID Hub TBEN-S2-2RFID-4DXP		
Mitigation: Automatic firmware version logging		
Name: Notification in case of firmware version change event		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected firmware version change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 9.1.6		
Subject: TURCK RFID Hub TBEN-S2-2RFID-4DXP		
Mitigation: Automatic firmware version logging		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 100 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 9.2.1		
Subject: TURCK RFID Hub TBEN-S2-2RFID-4DXP		
Mitigation: Detection and logging of parameter changes		
Name: Parameter change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any changes of parameter values. Changes shall only trigger once each change.		
Comments: The implementation using a state-of-the-art hashing algorithm with proper hemming distance is sufficient for detection reliability.		

ID: 9.2.2		
Subject: TURCK RFID Hub TBEN-S2-2RFID-4DXP		
Mitigation: Detection and logging of parameter changes		
Name: Parameter query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The Parameter values shall be queried at least 4 times per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 9.2.3		
Subject: TURCK RFID Hub TBEN-S2-2RFID-4DXP		
Mitigation: Detection and logging of parameter changes		
Name: Logging of Parameter value adjustments		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each parameter adjustment shall trigger the system to automatically record the event persistently.		
Comments: Using a suitable ticket system could be used for logging.		

ID: 9.2.4		
Subject: TURCK RFID Hub TBEN-S2-2RFID-4DXP		
Mitigation: Detection and logging of parameter changes		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 9.2.5		
Subject: TURCK RFID Hub TBEN-S2-2RFID-4DXP		
Mitigation: Detection and logging of parameter changes		
Name: Notification in case of parameter adjustment events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected paremeter adjustment event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 9.2.6		
Subject: TURCK RFID Hub TBEN-S2-2RFID-4DXP		
Mitigation: Detection and logging of parameter changes		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 1500 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 9.2.7		
Subject: TURCK RFID Hub TBEN-S2-2RFID-4DXP		
Mitigation: Detection and logging of parameter changes		
Name: Parameter changes differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Parameter changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 10.1.1		
Subject: Festo CMMP-AS-C2-3A-M3		
Mitigation: Automatic firmware version logging		
Name: FW version change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any artifact version change. Events shall only be triggered once each change.		
Comments: The artifact version could be acquired from the target device using appropriate communication protocols.		

ID: 10.1.2		
Subject: Festo CMMP-AS-C2-3A-M3		
Mitigation: Automatic firmware version logging		
Name: FW version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The FW version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 10.1.3		
Subject: Festo CMMP-AS-C2-3A-M3		
Mitigation: Automatic firmware version logging		
Name: Store records only on firmware change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall only contain records triggered by a change of the firmware version compared the current stored record.		
Comments: Avoid unnecessary redundant records.		

ID: 10.1.4		
Subject: Festo CMMP-AS-C2-3A-M3		
Mitigation: Automatic firmware version logging		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 10.1.5		
Subject: Festo CMMP-AS-C2-3A-M3		
Mitigation: Automatic firmware version logging		
Name: Notification in case of firmware version change event		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected firmware version change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 10.1.6		
Subject: Festo CMMP-AS-C2-3A-M3		
Mitigation: Automatic firmware version logging		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 100 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 10.2.1		
Subject: Festo CMMP-AS-C2-3A-M3		
Mitigation: Detection and logging of configuration changes		
Name: Configuration change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any changes of configuration values. Changes shall only trigger once each change.		
Comments: The implementation using a state-of-the-art hashing algorithm with proper hemming distance is sufficient for detection reliability.		

ID: 10.2.2		
Subject: Festo CMMP-AS-C2-3A-M3		
Mitigation: Detection and logging of configuration changes		
Name: Logging of configuration value adjustments		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each configuration adjustment shall trigger the system to automatically record the event persistently.		
Comments: Using a suitable ticket system could be used for logging.		

ID: 10.2.3		
Subject: Festo CMMP-AS-C2-3A-M3		
Mitigation: Detection and logging of configuration changes		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 10.2.4		
Subject: Festo CMMP-AS-C2-3A-M3		
Mitigation: Detection and logging of configuration changes		
Name: Notification in case of configuration adjustment events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected configuration adjustment event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 10.2.5		
Subject: Festo CMMP-AS-C2-3A-M3		
Mitigation: Detection and logging of configuration changes		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 50 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 10.2.6		
Subject: Festo CMMP-AS-C2-3A-M3		
Mitigation: Detection and logging of configuration changes		
Name: Configuration changes differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Configuration changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 10.3.1		
Subject: Festo CMMP-AS-C2-3A-M3		
Mitigation: Detection and logging of configuration changes (HW)		
Name: Configuration change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any changes of configuration values. Changes shall only trigger once each change.		
Comments: The implementation using a state-of-the-art hashing algorithm with proper hemming distance is sufficient for detection reliability.		

ID: 10.3.2		
Subject: Festo CMMP-AS-C2-3A-M3		
Mitigation: Detection and logging of configuration changes (HW)		
Name: Logging of configuration value adjustments		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each configuration adjustment shall trigger the system to automatically record the event persistently.		
Comments: Using a suitable ticket system could be used for logging.		

ID: 10.3.3		
Subject: Festo CMMP-AS-C2-3A-M3		
Mitigation: Detection and logging of configuration changes (HW)		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 10.3.4		
Subject: Festo CMMP-AS-C2-3A-M3		
Mitigation: Detection and logging of configuration changes (HW)		
Name: Notification in case of configuration adjustment events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected configuration adjustment event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 10.3.5		
Subject: Festo CMMP-AS-C2-3A-M3		
Mitigation: Detection and logging of configuration changes (HW)		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 50 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 10.3.6		
Subject: Festo CMMP-AS-C2-3A-M3		
Mitigation: Detection and logging of configuration changes (HW)		
Name: Configuration changes differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Configuration changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 11.1.1		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Automatic firmware version logging		
Name: FW version change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any artifact version change. Events shall only be triggered once each change.		
Comments: The artifact version could be acquired from the target device using appropriate communication protocols.		

ID: 11.1.2		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Automatic firmware version logging		
Name: FW version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The FW version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 11.1.3		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Automatic firmware version logging		
Name: Store records only on firmware change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall only contain records triggered by a change of the firmware version compared the current stored record.		
Comments: Avoid unnecessary redundant records.		

ID: 11.1.4		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Automatic firmware version logging		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 11.1.5		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Automatic firmware version logging		
Name: Notification in case of firmware version change event		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected firmware version change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 11.1.6		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Automatic firmware version logging		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 100 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 11.2.1		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Automatic SPS application version logging		
Name: Application version change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any artifact version change. Events shall only be triggered once each change.		
Comments: The artifact version could be acquired from the target device using appropriate communication protocols.		

ID: 11.2.2		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Automatic SPS application version logging		
Name: Application version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The Application version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 11.2.3		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Automatic SPS application version logging		
Name: Store records only on application change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall only contain records triggered by a change of the application version compared the current stored record.		
Comments: Avoid unnecessary redundant records.		

ID: 11.2.4		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Automatic SPS application version logging		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 11.2.5		
Subject: Festo MES4 System		
Mitigation: Automatic SPS application version logging		
Name: Notification in case of SPS application version change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected SPS application version change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 11.2.6		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Automatic SPS application version logging		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 200 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 11.3.1		
Subject: Festo SBOC-Q-R3C Camera App SPS		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Artifact binary integrity surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any changes within the artifact section. Changes shall only trigger once each change.		
Comments: The implementation using a state-of-the-art hashing algorithm with proper hemming distance is sufficient for detection reliability.		

ID: 11.3.2		
Subject: Festo SBOC-Q-R3C Camera App SPS		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Artifact version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The application version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 11.3.3		
Subject: Festo SBOC-Q-R3C Camera App SPS		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Logging of binary change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each binary integrity modification event shall trigger the system to automatically record the event persistently.		
Comments: Using a suitable ticket system could be used for logging.		

ID: 11.3.4		
Subject: Festo SBOC-Q-R3C Camera App SPS		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 11.3.5		
Subject: Festo SBOC-Q-R3C Camera App SPS		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Notification in case of binary change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected binary change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 11.3.6		
Subject: Festo SBOC-Q-R3C Camera App SPS		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 50 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 11.4.1		
Subject: Festo SBOC-Q-R3C Camera App SPS		
Mitigation: Comprehensive development process		
Name: Ticket system		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall implement a ticket based project management system in order to provide a guided development life cycle.		
Comments: The ticket system implementation should be directly linked with the version control and release management domains.		

ID: 11.4.2		
Subject: Festo SBOC-Q-R3C Camera App SPS		
Mitigation: Comprehensive development process		
Name: Ticket system - Unique identifier		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each ticket shall be represented by a unique identifier.		
Comments: N/A		

ID: 11.4.3		
Subject: Festo SBOC-Q-R3C Camera App SPS		
Mitigation: Comprehensive development process		
Name: Ticket system - User and role management		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The ticket system shall support a user and role oriented management. Users shall be linked to roles, the roles shall be configurable regarding authorization levels.		
Comments: N/A		

ID: 11.4.4		
Subject: Festo SBOC-Q-R3C Camera App SPS		
Mitigation: Comprehensive development process		
Name: Ticket system - Process oriented ticket management		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The ticket system shall implement the well established enterprise development process. Following attributes are mandatory: • Lifecycle state • Owner • Severity/Priority • Type/Class • Target release		
Comments: N/A		

ID: 11.4.5		
Subject: Festo SBOC-Q-R3C Camera App SPS		
Mitigation: Comprehensive development process		
Name: Transparent version control		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall implement a version control framework to trace detailed changes of the application content.		
Comments: The version control implementation should be directly linked with the ticket system and release management domains.		

ID: 11.4.6		
Subject: Festo SBOC-Q-R3C Camera App SPS		
Mitigation: Comprehensive development process		
Name: Transparent version control - Longevity support		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Selection of version control system shall be conducted to ensure longevity support of at least 10 years.		
Comments: If applicable, internal maintenance of a open source solution should be preferred.		

ID: 11.4.7		
Subject: Festo SBOC-Q-R3C Camera App SPS		
Mitigation: Comprehensive development process		
Name: Transparent version control - Artifact content differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Content changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 11.4.8		
Subject: Festo SBOC-Q-R3C Camera App SPS		
Mitigation: Comprehensive development process		
Name: Roll out history of application releases		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall record any deployment of releases to the MES4 central SCADA system. Each record shall contain the following data: • Date • Version • Purpose		
Comments: The release management implementation should be directly linked with the ticket system and version control domains.		

ID: 11.4.9		
Subject: Festo SBOC-Q-R3C Camera App SPS		
Mitigation: Comprehensive development process		
Name: Automatic release deployment roll back		
Revision: 1	**Date:** 30.10.2020	**Status:** postponed
Description: In case of a necessary roll back, the system shall provide an automated solution.		
Comments: Until integration of the automated system, the release deployment roll back shall be accomplished manually.		

ID: 11.5.1		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Automatic optical verification program application version logging		
Name: Application version change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any artifact version change. Events shall only be triggered once each change.		
Comments: The artifact version could be acquired from the target device using appropriate communication protocols.		

ID: 11.5.2		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Automatic optical verification program application version logging		
Name: Application version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The Application version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 11.5.3		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Automatic optical verification program application version logging		
Name: Store records only on application change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall only contain records triggered by a change of the application version compared the current stored record.		
Comments: Avoid unnecessary redundant records.		

ID: 11.5.4		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Automatic optical verification program application version logging		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 11.5.5		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Automatic optical verification program application version logging		
Name: Notification in case of application version change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected application version change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 11.5.6		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Automatic optical verification program application version logging		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 1000 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 11.6.1		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Artifact binary integrity surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any changes within the artifact section. Changes shall only trigger once each change.		
Comments: The implementation using a state-of-the-art hashing algorithm with proper hemming distance is sufficient for detection reliability.		

ID: 11.6.2		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Artifact version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The application version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 11.6.3		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Logging of binary change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each binary integrity modification event shall trigger the system to automatically record the event persistently.		
Comments: Using a suitable ticket system could be used for logging.		

ID: 11.6.4		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 11.6.5		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Notification in case of binary change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected binary change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 11.6.6		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 50 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 11.7.1		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Comprehensive development process		
Name: Ticket system		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall implement a ticket based project management system in order to provide a guided development life cycle.		
Comments: The ticket system implementation should be directly linked with the version control and release management domains.		

ID: 11.7.2		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Comprehensive development process		
Name: Ticket system - Unique identifier		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each ticket shall be represented by a unique identifier.		
Comments: N/A		

ID: 11.7.3		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Comprehensive development process		
Name: Ticket system - User and role management		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The ticket system shall support a user and role oriented management. Users shall be linked to roles, the roles shall be configurable regarding authorization levels.		
Comments: N/A		

ID: 11.7.4		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Comprehensive development process		
Name: Ticket system - Process oriented ticket management		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The ticket system shall implement the well established enterprise development process. Following attributes are mandatory: • Lifecycle state • Owner • Severity/Priority • Type/Class • Target release		
Comments: N/A		

ID: 11.7.5		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Comprehensive development process		
Name: Transparent version control		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall implement a version control framework to trace detailed changes of the application content.		
Comments: The version control implementation should be directly linked with the ticket system and release management domains.		

ID: 11.7.6		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Comprehensive development process		
Name: Transparent version control - Longevity support		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Selection of version control system shall be conducted to ensure longevity support of at least 10 years.		
Comments: If applicable, internal maintenance of a open source solution should be preferred.		

ID: 11.7.7		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Comprehensive development process		
Name: Transparent version control - Artifact content differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Content changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 11.7.8		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Comprehensive development process		
Name: Roll out history of application releases		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall record any deployment of releases to the MES4 central SCADA system. Each record shall contain the following data: • Date • Version • Purpose		
Comments: The release management implementation should be directly linked with the ticket system and version control domains.		

ID: 11.7.9		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Comprehensive development process		
Name: Automatic release deployment roll back		
Revision: 1	**Date:** 30.10.2020	**Status:** postponed
Description: In case of a necessary roll back, the system shall provide an automated solution.		
Comments: Until integration of the automated system, the release deployment roll back shall be accomplished manually.		

ID: 11.8.1		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Detection and logging of parameter changes		
Name: Parameter change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any changes of parameter values. Changes shall only trigger once each change.		
Comments: The implementation using a state-of-the-art hashing algorithm with proper hemming distance is sufficient for detection reliability.		

ID: 11.8.2		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Detection and logging of parameter changes		
Name: Parameter query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The Parameter values shall be queried at least 4 times per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 11.8.3		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Detection and logging of parameter changes		
Name: Logging of Parameter value adjustments		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each parameter adjustment shall trigger the system to automatically record the event persistently.		
Comments: Using a suitable ticket system could be used for logging.		

ID: 11.8.4		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Detection and logging of parameter changes		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 11.8.5		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Detection and logging of parameter changes		
Name: Notification in case of parameter adjustment events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected paremeter adjustment event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 11.8.6		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Detection and logging of parameter changes		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 1500 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 11.8.7		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Detection and logging of parameter changes		
Name: Parameter changes differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Parameter changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 11.9.1		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Detection and logging of configuration changes		
Name: Configuration change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any changes of configuration values. Changes shall only trigger once each change.		
Comments: The implementation using a state-of-the-art hashing algorithm with proper hemming distance is sufficient for detection reliability.		

ID: 11.9.2		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Detection and logging of configuration changes		
Name: Logging of configuration value adjustments		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each configuration adjustment shall trigger the system to automatically record the event persistently.		
Comments: Using a suitable ticket system could be used for logging.		

ID: 11.9.3		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Detection and logging of configuration changes		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 11.9.4		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Detection and logging of configuration changes		
Name: Notification in case of configuration adjustment events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected configuration adjustment event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 11.9.5		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Detection and logging of configuration changes		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 50 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 11.9.6		
Subject: Festo SBOC-Q-R3C Camera		
Mitigation: Detection and logging of configuration changes		
Name: Configuration changes differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Configuration changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 12.1.1		
Subject: KUKA KRC4 Robot Control		
Mitigation: Automatic firmware version logging		
Name: FW version change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any artifact version change. Events shall only be triggered once each change.		
Comments: The artifact version could be acquired from the target device using appropriate communication protocols.		

ID: 12.1.2		
Subject: KUKA KRC4 Robot Control		
Mitigation: Automatic firmware version logging		
Name: FW version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The FW version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 12.1.3		
Subject: KUKA KRC4 Robot Control		
Mitigation: Automatic firmware version logging		
Name: Store records only on firmware change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall only contain records triggered by a change of the firmware version compared the current stored record.		
Comments: Avoid unnecessary redundant records.		

ID: 12.1.4		
Subject: KUKA KRC4 Robot Control		
Mitigation: Automatic firmware version logging		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 12.1.5		
Subject: KUKA KRC4 Robot Control		
Mitigation: Automatic firmware version logging		
Name: Notification in case of firmware version change event		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected firmware version change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 12.1.6		
Subject: KUKA KRC4 Robot Control		
Mitigation: Automatic firmware version logging		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 100 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 12.2.1		
Subject: KUKA KRC4 Robot Control		
Mitigation: Automatic KUKA Robotic Language (KRL) application version logging		
Name: Application version change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any artifact version change. Events shall only be triggered once each change.		
Comments: The artifact version could be acquired from the target device using appropriate communication protocols.		

ID: 12.2.2		
Subject: KUKA KRC4 Robot Control		
Mitigation: Automatic KUKA Robotic Language (KRL) application version logging		
Name: Application version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The Application version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 12.2.3		
Subject: KUKA KRC4 Robot Control		
Mitigation: Automatic KUKA Robotic Language (KRL) application version logging		
Name: Store records only on application change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall only contain records triggered by a change of the application version compared the current stored record.		
Comments: Avoid unnecessary redundant records.		

ID: 12.2.4		
Subject: KUKA KRC4 Robot Control		
Mitigation: Automatic KUKA Robotic Language (KRL) application version logging		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 12.2.5		
Subject: KUKA KRC4 Robot Control		
Mitigation: Automatic KUKA Robotic Language (KRL) application version logging		
Name: Notification in case of application version change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected application version change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 12.2.6		
Subject: KUKA KRC4 Robot Control		
Mitigation: Automatic KUKA Robotic Language (KRL) application version logging		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 500 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 12.3.1		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Artifact binary integrity surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any changes within the artifact section. Changes shall only trigger once each change.		
Comments: The implementation using a state-of-the-art hashing algorithm with proper hemming distance is sufficient for detection reliability.		

ID: 12.3.2		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Artifact version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The application version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 12.3.3		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Logging of binary change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each binary integrity modification event shall trigger the system to automatically record the event persistently.		
Comments: Using a suitable ticket system could be used for logging.		

ID: 12.3.4		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 12.3.5		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Notification in case of binary change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected binary change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 12.3.6		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 50 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 12.4.1		
Subject: KUKA KRC4 Robot Control		
Mitigation: Comprehensive development process		
Name: Ticket system		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall implement a ticket based project management system in order to provide a guided development life cycle.		
Comments: The ticket system implementation should be directly linked with the version control and release management domains.		

ID: 12.4.2		
Subject: KUKA KRC4 Robot Control		
Mitigation: Comprehensive development process		
Name: Ticket system - Unique identifier		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each ticket shall be represented by a unique identifier.		
Comments: N/A		

ID: 12.4.3		
Subject: KUKA KRC4 Robot Control		
Mitigation: Comprehensive development process		
Name: Ticket system - User and role management		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The ticket system shall support a user and role oriented management. Users shall be linked to roles, the roles shall be configurable regarding authorization levels.		
Comments: N/A		

ID: 12.4.4		
Subject: KUKA KRC4 Robot Control		
Mitigation: Comprehensive development process		
Name: Ticket system - Process oriented ticket management		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The ticket system shall implement the well established enterprise development process. Following attributes are mandatory: • Lifecycle state • Owner • Severity/Priority • Type/Class • Target release		
Comments: N/A		

ID: 12.4.5		
Subject: KUKA KRC4 Robot Control		
Mitigation: Comprehensive development process		
Name: Transparent version control		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall implement a version control framework to trace detailed changes of the application content.		
Comments: The version control implementation should be directly linked with the ticket system and release management domains.		

ID: 12.4.6		
Subject: KUKA KRC4 Robot Control		
Mitigation: Comprehensive development process		
Name: Transparent version control - Longevity support		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Selection of version control system shall be conducted to ensure longevity support of at least 10 years.		
Comments: If applicable, internal maintenance of a open source solution should be preferred.		

ID: 12.4.7		
Subject: KUKA KRC4 Robot Control		
Mitigation: Comprehensive development process		
Name: Transparent version control - Artifact content differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Content changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 12.4.8		
Subject: KUKA KRC4 Robot Control		
Mitigation: Comprehensive development process		
Name: Roll out history of application releases		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall record any deployment of releases to the MES4 central SCADA system. Each record shall contain the following data: • Date • Version • Purpose		
Comments: The release management implementation should be directly linked with the ticket system and version control domains.		

ID: 12.4.9		
Subject: KUKA KRC4 Robot Control		
Mitigation: Comprehensive development process		
Name: Automatic release deployment roll back		
Revision: 1	**Date:** 30.10.2020	**Status:** postponed
Description: In case of a necessary roll back, the system shall provide an automated solution.		
Comments: Until integration of the automated system, the release deployment roll back shall be accomplished manually.		

ID: 12.5.1		
Subject: KUKA KRC4 Robot Control		
Mitigation: Automatic SPS application version logging		
Name: Application version change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any artifact version change. Events shall only be triggered once each change.		
Comments: The artifact version could be acquired from the target device using appropriate communication protocols.		

ID: 12.5.2		
Subject: KUKA KRC4 Robot Control		
Mitigation: Automatic SPS application version logging		
Name: Application version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The Application version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 12.5.3		
Subject: KUKA KRC4 Robot Control		
Mitigation: Automatic SPS application version logging		
Name: Store records only on application change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall only contain records triggered by a change of the application version compared the current stored record.		
Comments: Avoid unnecessary redundant records.		

ID: 12.5.4		
Subject: KUKA KRC4 Robot Control		
Mitigation: Automatic SPS application version logging		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 12.5.5		
Subject: KUKA KRC4 Robot Control		
Mitigation: Automatic SPS application version logging		
Name: Notification in case of SPS application version change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected SPS application version change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 12.5.6		
Subject: KUKA KRC4 Robot Control		
Mitigation: Automatic SPS application version logging		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 300 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 12.6.1		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Artifact binary integrity surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any changes within the artifact section. Changes shall only trigger once each change.		
Comments: The implementation using a state-of-the-art hashing algorithm with proper hemming distance is sufficient for detection reliability.		

ID: 12.6.2		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Artifact version query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The application version information shall be queried at least once per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 12.6.3		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Logging of binary change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each binary integrity modification event shall trigger the system to automatically record the event persistently.		
Comments: Using a suitable ticket system could be used for logging.		

ID: 12.6.4		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 12.6.5		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Notification in case of binary change events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected binary change event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 12.6.6		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of unauthorized artifact changes		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 50 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 12.7.1		
Subject: KUKA KRC4 Robot Control		
Mitigation: Comprehensive development process		
Name: Ticket system		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall implement a ticket based project management system in order to provide a guided development life cycle.		
Comments: The ticket system implementation should be directly linked with the version control and release management domains.		

ID: 12.7.2		
Subject: KUKA KRC4 Robot Control		
Mitigation: Comprehensive development process		
Name: Ticket system - Unique identifier		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each ticket shall be represented by a unique identifier.		
Comments: N/A		

ID: 12.7.3		
Subject: KUKA KRC4 Robot Control		
Mitigation: Comprehensive development process		
Name: Ticket system - User and role management		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The ticket system shall support a user and role oriented management. Users shall be linked to roles, the roles shall be configurable regarding authorization levels.		
Comments: N/A		

ID: 12.7.4		
Subject: KUKA KRC4 Robot Control		
Mitigation: Comprehensive development process		
Name: Ticket system - Process oriented ticket management		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The ticket system shall implement the well established enterprise development process. Following attributes are mandatory: • Lifecycle state • Owner • Severity/Priority • Type/Class • Target release		
Comments: N/A		

ID: 12.7.5		
Subject: KUKA KRC4 Robot Control		
Mitigation: Comprehensive development process		
Name: Transparent version control		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall implement a version control framework to trace detailed changes of the application content.		
Comments: The version control implementation should be directly linked with the ticket system and release management domains.		

ID: 12.7.6		
Subject: KUKA KRC4 Robot Control		
Mitigation: Comprehensive development process		
Name: Transparent version control - Longevity support		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Selection of version control system shall be conducted to ensure longevity support of at least 10 years.		
Comments: If applicable, internal maintenance of a open source solution should be preferred.		

ID: 12.7.7		
Subject: KUKA KRC4 Robot Control		
Mitigation: Comprehensive development process		
Name: Transparent version control - Artifact content differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Content changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 12.7.8		
Subject: KUKA KRC4 Robot Control		
Mitigation: Comprehensive development process		
Name: Roll out history of application releases		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall record any deployment of releases to the MES4 central SCADA system. Each record shall contain the following data: • Date • Version • Purpose		
Comments: The release management implementation should be directly linked with the ticket system and version control domains.		

ID: 12.7.9		
Subject: KUKA KRC4 Robot Control		
Mitigation: Comprehensive development process		
Name: Automatic release deployment roll back		
Revision: 1	**Date:** 30.10.2020	**Status:** postponed
Description: In case of a necessary roll back, the system shall provide an automated solution.		
Comments: Until integration of the automated system, the release deployment roll back shall be accomplished manually.		

ID: 12.8.1		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of parameter changes		
Name: Parameter change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any changes of parameter values. Changes shall only trigger once each change.		
Comments: The implementation using a state-of-the-art hashing algorithm with proper hemming distance is sufficient for detection reliability.		

ID: 12.8.2		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of parameter changes		
Name: Parameter query interval		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The Parameter values shall be queried at least 4 times per hour.		
Comments: Consider resulting possible high network load caused by high polling rate.		

ID: 12.8.3		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of parameter changes		
Name: Logging of Parameter value adjustments		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each parameter adjustment shall trigger the system to automatically record the event persistently.		
Comments: Using a suitable ticket system could be used for logging.		

ID: 12.8.4		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of parameter changes		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 12.8.5		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of parameter changes		
Name: Notification in case of parameter adjustment events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected paremeter adjustment event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 12.8.6		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of parameter changes		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 1500 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 12.8.7		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of parameter changes		
Name: Parameter changes differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Parameter changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 12.9.1		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of configuration changes		
Name: Configuration change surveillance		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: A surveillance service shall monitor and detect any changes of configuration values. Changes shall only trigger once each change.		
Comments: The implementation using a state-of-the-art hashing algorithm with proper hemming distance is sufficient for detection reliability.		

ID: 12.9.2		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of configuration changes		
Name: Logging of configuration value adjustments		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each configuration adjustment shall trigger the system to automatically record the event persistently.		
Comments: Using a suitable ticket system could be used for logging.		

ID: 12.9.3		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of configuration changes		
Name: Storage format		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The data records shall be stored in a format to fulfill machine to machine interaction.		
Comments: Data accessible for further automated analysis.		

ID: 12.9.4		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of configuration changes		
Name: Notification in case of configuration adjustment events		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: In case of a detected configuration adjustment event, the system shall send a notification to a defined recipient.		
Comments: Using a suitable ticket system could be used for notification.		

ID: 12.9.5		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of configuration changes		
Name: Data history minimum record quantity		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The database shall be capable to store at least 50 records.		
Comments: Additional logging should be evaluated regarding data protection regulation (data minimizing).		

ID: 12.9.6		
Subject: KUKA KRC4 Robot Control		
Mitigation: Detection and logging of configuration changes		
Name: Configuration changes differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Configuration changes shall be comparable to previous versions in a detailed manner.		
Comments: A code diff should be supported using a graphical user interface.		

ID: 13.1.1		
Subject: Hardware infrastructure		
Mitigation: Hardware infrastructure design modification tracking		
Name: Ticket system		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall implement a ticket based project management system in order to provide a guided development life cycle.		
Comments: The ticket system implementation should be directly linked with the version control and release management domains.		

ID: 13.1.2		
Subject: Hardware infrastructure		
Mitigation: Hardware infrastructure design modification tracking		
Name: Ticket system - Unique identifier		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each ticket shall be represented by a unique identifier.		
Comments: N/A		

ID: 13.1.3		
Subject: Hardware infrastructure		
Mitigation: Hardware infrastructure design modification tracking		
Name: Ticket system - User and role management		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The ticket system shall support a user and role oriented management. Users shall be linked to roles, the roles shall be configurable regarding authorization levels.		
Comments: N/A		

ID: 13.1.4		
Subject: Hardware infrastructure		
Mitigation: Hardware infrastructure design modification tracking		
Name: Ticket system - Process oriented ticket management		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The ticket system shall implement the well established enterprise development process. Following attributes are mandatory: • Lifecycle state • Owner • Severity/Priority • Type/Class • Target release		
Comments: N/A		

ID: 13.1.5		
Subject: Hardware infrastructure		
Mitigation: Hardware infrastructure design modification tracking		
Name: Transparent design release control		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall implement a design release control framework to trace detailed changes of the content.		
Comments: The design release control implementation should be directly linked with the ticket system and release management domains.		

ID: 13.1.6		
Subject: Hardware infrastructure		
Mitigation: Hardware infrastructure design modification tracking		
Name: Design release control - Longevity support		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Selection of version control system shall be conducted to ensure longevity support of at least 10 years.		
Comments: If applicable, internal maintenance of a open source solution should be preferred.		

ID: 13.1.7		
Subject: Hardware infrastructure		
Mitigation: Hardware infrastructure design modification tracking		
Name: Design release control - Artifact content differences comparison		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Content changes shall be comparable to previous versions in a detailed manner.		
Comments: A content diff should be supported using a graphical user interface.		

ID: 13.1.8		
Subject: Hardware infrastructure		
Mitigation: Hardware infrastructure design modification tracking		
Name: Design modification application		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The system shall provide all data required for the physical application of the new design state.		
Comments: N/A		

ID: 13.1.9		
Subject: Hardware infrastructure		
Mitigation: Hardware infrastructure design modification tracking		
Name: Design modification differences		
Revision: 1	**Date:** 30.10.2020	**Status:** postponed
Description: The content of the releases shall be comparable regarding changes between design releases.		
Comments: A ticket system could be used.		

ID: 13.2.1		
Subject: Hardware infrastructure		
Mitigation: Implementation of computer aided spare part replacement process		
Name: Ticket system		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall implement a ticket based project management system in order to provide a guided development life cycle.		
Comments: The ticket system implementation should be directly linked with the version control and release management domains.		

ID: 13.2.2		
Subject: Hardware infrastructure		
Mitigation: Implementation of computer aided spare part replacement process		
Name: Ticket system - Unique identifier		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each ticket shall be represented by a unique identifier.		
Comments: N/A		

ID: 13.2.3		
Subject: Hardware infrastructure		
Mitigation: Implementation of computer aided spare part replacement process		
Name: Ticket system - User and role management		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The ticket system shall support a user and role oriented management. Users shall be linked to roles, the roles shall be configurable regarding authorization levels.		
Comments: N/A		

ID: 13.2.4		
Subject: Hardware infrastructure		
Mitigation: Implementation of computer aided spare part replacement process		
Name: Ticket system - Process oriented ticket management		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The ticket system shall implement the well established enterprise development process. Following attributes are mandatory: • Lifecycle state • Owner • Severity/Priority • Type/Class • Target release		
Comments: N/A		

ID: 13.2.5		
Subject: Hardware infrastructure		
Mitigation: Implementation of computer aided spare part replacement process		
Name: Transparent spare part replacement tracking		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The configuration management shall implement a transparent spare part replacement tracking framework to trace detailed changes of the content.		
Comments: The design release control implementation should be directly linked with the ticket system and release management domains.		

ID: 13.2.6		
Subject: Hardware infrastructure		
Mitigation: Implementation of computer aided spare part replacement process		
Name: Transparent spare part replacement tracking - automatic information aquisition		
Revision: 1	**Date:** 30.10.2020	**Status:** postponed
Description: The system shall provide automatic information aquisition of the replaced components, if applicable.		
Comments: N/A		

ID: 13.2.7		
Subject: Hardware infrastructure		
Mitigation: Implementation of computer aided spare part replacement process		
Name: Transparent spare part replacement tracking - unique spare part information		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: Each record shall contain the following data: • Equipment identification marking • Serial number (removed part) • Serial number (installed part) • Revision status		
Comments: N/A		

ID: 13.2.8		
Subject: Hardware infrastructure		
Mitigation: Implementation of computer aided spare part replacement process		
Name: Spare part replacement application		
Revision: 1	**Date:** 30.10.2020	**Status:** approved
Description: The system shall provide a replacement instruction for application.		
Comments: N/A		

C. Appendix: Integration Test

Hochschule
Augsburg University of
Applied Sciences

Major Project 2020

Master Industrial Security
University of Applied Sciences Augsburg

Test Report
Integration Test

Table 1: Student authors

Student	Student ID
Alexander Holzmann	2081881
Markus Kamm	2060929
Michael Wager	2081894

1 Test Report

Report ID: CM_TR_RC1_integration_test_iteration1
Date: 02.12.2020
Tester: Team CM
Result: passed
Object under Test: CM server infrastructure - Release Candidate 1
Test Environment: • Lenovo IdeaPad 510s • Windows 10 Professional • CPU Intel Core i7 • 8GB RAM • WLAN network interface • Wired LAN interface • Oracle VM Virtual Box Version 6.1.8 r137981 • Webbrowser: Mozilla Firefox v83.0 (64bit)

Test result summary:	
Step	**result**
1. Login to GitLab web frontend	passed
2. Create GitLab project	passed
3. Add member to project group	passed
4. Create new milestone	passed
5. Create new issue	passed
6. Reassign issue	passed
7. Assign issue to milestone	passed
8. Create new merge request and new branch	passed
9. Modify content of repository	passed
10. Approve and close merge request	passed
11. Close milestone	passed
12. Create new release (release freeze)	passed

2 Test Steps

1. Login to GitLab web frontend	
1.1 Access web frontend via URL "https://yeskia-hs.com" Expected result: GitLab login website visible	passed
1.2 User login on web frontend (user: root, password: Passw0rd!) Expected result: GitLab user dashboard visible	passed

2. Create GitLab project	
2.1 Create new blank project (button "new project") Expected result: Project start page of created project	passed

3. Add member to project group	
3.1 Open project member view (menu: project - members) Expected result: formular for assigning members visible	passed
3.2 Assign member to project (search user and assign with desired role description) and confirm Expected result: new member invited via email to project	passed

4. Create new milestone	
4.1 Create new milestone (menu: issue - milestones - new) Expected result: formular for entering milestone information visible	passed
4.2 Enter milestone information data (description, ...) and confirm Expected result: new milestone available	passed

5. Create new issue	
5.1 Create new issue (menu: issue - new) Expected result: formular for entering issue information visible	passed
5.2 Enter issue information data (description, ...) and confirm Expected result: new issue available	passed

6. Reassign issue	
6.1 Open existing issue (menu: issue) Expected result: existing issue information visible	passed
6.2 Change assignee to other user (edit assignee) and confirm Expected result: new assigned user to issue	passed

7. Assign issue to milestone	
7.1 Assign existing issue to milestone (menu: issue - milestones) Expected result: issue assigned to milestone	passed

8. Create new merge request and new branch	
8.1 Create new merge request (menu: merge request - new) Expected result: formular for entering merge request information visible	passed
8.2 Enter merge request information data (description, ...) and confirm Expected result: new merge request available	passed
8.3 Assign merge request to maintainer (edit dropdown) Expected result: maintainer assigned to merge request	passed
8.4 Create associated branch for git repository (implicitly created with merge request) Expected result: associated branch available	passed

9. Modify content of repository	
9.1 Clone repository content to local machine (`git clone <repository URL>`) Expected result: repository content available on local drive	passed
9.2 Checkout branch associated with merge request (`git checkout <branchname>`) Expected result: repository branch changed to `<branchname>`	passed
9.3 Modify content of the repository on local drive (change any file content) Expected result: file content differs to origin state	passed
9.4 Stage and commit modified files (`git commit -a -m <commit message>`) Expected result: commit successful	passed
9.5 Push commit to server (`git push origin HEAD`) Expected result: commit successfully pushed to server	passed

10. Approve and close merge request	
10.1 Close issue (menu: issue - close) Expected result: issue in state `closed`	passed
10.2 Prepare merge request for closing (remove draft prefix from name) Expected result: merge request name without the prefix "draft"	passed
10.3 Approve merge request (button approve) Expected result: merge request marked as `approved`	passed
10.4 Merge changes to master branch (button merge) Expected result: repository content changes available at master branch, merge request in state `merged`	passed

11. Close milestone	
11.1 Close milestone (menu: milestone - close) Expected result: milestone in state `closed`	passed

12. Create new release (release freeze)	
12.1 Create new release (menu: release - new) Expected result: formular for entering release information visible	passed
12.2 Assign existing milestone to release (dropdown milestones) Expected result: milestone assigned to release	passed
12.3 Enter release information data (description, ...) and confirm closing Expected result: new release available and closed	passed

z

D. Appendix: Customer Integration Test

Hochschule
Augsburg University of
Applied Sciences

Major Project 2020

Master Industrial Security
University of Applied Sciences Augsburg

Test Report
Customer Integration Test

Table 1: Student authors

Student	Student ID
Alexander Holzmann	2081881
Markus Kamm	2060929
Michael Wager	2081894

1 Test Report - Customer Integration 1

Report ID: CM_TR_RC1_customer_integration_test_iteration1
Date: 03.12.2020
Tester: Team CM
Result: failed
Object under Test: CM server infrastructure - Release Candidate 1
Test Environment: • DELL OptiPlex 5050 • Windows 10 Professional • CPU Intel Core i5 • 8GB RAM • Wired LAN interface (2x; to Local Network, deactivated) • Oracle VM Virtual Box Version 6.1.16 r140961 • Webbrowser: Mozilla Firefox v83.0 (64bit)

Test result summary:	
Step	**result**
1. Login to GitLab web frontend	failed
2. Create GitLab project	NA
3. Add member to project group	NA
4. Create new milestone	NA
5. Create new issue	NA
6. Reassign issue	NA
7. Assign issue to milestone	NA
8. Create new merge request and new branch	NA
9. Modify content of repository	NA
10. Approve and close merge request	NA
11. Close milestone	NA
12. Create new release (release freeze)	NA

2 Test Steps - Customer Integration 1

1. Login to GitLab web frontend	
1.1 Access web frontend via URL "https://yeskia-hs.com" Expected result: GitLab login website visible	failed
1.2 User login on web frontend (user: root, password: Passw0rd!) Expected result: GitLab user dashboard visible	NA

2. Create GitLab project	
2.1 Create new blank project (button "new project") Expected result: Project start page of created project	NA

3. Add member to project group	
3.1 Open project member view (menu: project - members) Expected result: formular for assigning members visible	NA
3.2 Assign member to project (search user and assign with desired role description) and confirm Expected result: new member invited via email to project	NA

4. Create new milestone	
4.1 Create new milestone (menu: issue - milestones - new) Expected result: formular for entering milestone information visible	NA
4.2 Enter milestone information data (description, ...) and confirm Expected result: new milestone available	NA

5. Create new issue	
5.1 Create new issue (menu: issue - new) Expected result: formular for entering issue information visible	NA
5.2 Enter issue information data (description, ...) and confirm Expected result: new issue available	NA

6. Reassign issue	
6.1 Open existing issue (menu: issue) Expected result: existing issue information visible	NA
6.2 Change assignee to other user (edit assignee) and confirm Expected result: new assigned user to issue	NA

7. Assign issue to milestone	
7.1 Assign existing issue to milestone (menu: issue - milestones) Expected result: issue assigned to milestone	NA

8. Create new merge request and new branch	
8.1 Create new merge request (menu: merge request - new) Expected result: formular for entering merge request information visible	NA
8.2 Enter merge request information data (description, ...) and confirm Expected result: new merge request available	NA
8.3 Assign merge request to maintainer (edit dropdown) Expected result: maintainer assigned to merge request	NA
8.4 Create associated branch for git repository (implicitly created with merge request) Expected result: associated branch available	NA

9. Modify content of repository	
9.1 Clone repository content to local machine (`git clone <repository URL>`) Expected result: repository content available on local drive	NA
9.2 Checkout branch associated with merge request (`git checkout <branchname>`) Expected result: repository branch changed to `<branchname>`	NA
9.3 Modify content of the repository on local drive (change any file content) Expected result: file content differs to origin state	NA
9.4 Stage and commit modified files (`git commit -a -m <commit message>`) Expected result: commit successful	NA
9.5 Push commit to server (`git push origin HEAD`) Expected result: commit successfully pushed to server	NA

10. Approve and close merge request	
10.1 Close issue (menu: issue - close) Expected result: issue in state `closed`	NA
10.2 Prepare merge request for closing (remove draft prefix from name) Expected result: merge request name without the prefix "draft"	NA
10.3 Approve merge request (button approve) Expected result: merge request marked as `approved`	NA
10.4 Merge changes to master branch (button merge) Expected result: repository content changes available at master branch, merge request in state `merged`	NA

11. Close milestone	
11.1 Close milestone (menu: milestone - close) Expected result: milestone in state `closed`	NA

12. Create new release (release freeze)	
12.1 Create new release (menu: release - new) Expected result: formular for entering release information visible	NA
12.2 Assign existing milestone to release (dropdown milestones) Expected result: milestone assigned to release	NA
12.3 Enter release information data (description, ...) and confirm closing Expected result: new release available and closed	NA

3 Test Report - Customer Integration 2

Report ID: CM_TR_RC1_customer_integration_test_iteration2
Date: 03.12.2020
Tester: Team CM
Result: failed
Object under Test: CM server infrastructure - Release Candidate 1
Test Environment: • DELL OptiPlex 5050 • Windows 10 Professional • CPU Intel Core i5 • 8GB RAM • Wired LAN interface (2x; to Local Network, deactivated) • Oracle VM Virtual Box Version 6.1.16 r140961 • Webbrowser: Mozilla Firefox v83.0 (64bit)

Test result summary:	
Step	**result**
1. Login to GitLab web frontend	passed
2. Create GitLab project	passed
3. Add member to project group	passed
4. Create new milestone	passed
5. Create new issue	passed
6. Reassign issue	passed
7. Assign issue to milestone	passed
8. Create new merge request and new branch	failed
9. Modify content of repository	NA
10. Approve and close merge request	NA
11. Close milestone	NA
12. Create new release (release freeze)	NA

4 Test Steps - Customer Integration 2

1. Login to GitLab web frontend	
1.1 Access web frontend via URL "https://yeskia-hs.com" Expected result: GitLab login website visible	passed
1.2 User login on web frontend (user: root, password: Passw0rd!) Expected result: GitLab user dashboard visible	passed

2. Create GitLab project	
2.1 Create new blank project (button "new project") Expected result: Project start page of created project	passed

3. Add member to project group	
3.1 Open project member view (menu: project - members) Expected result: formular for assigning members visible	passed
3.2 Assign member to project (search user and assign with desired role description) and confirm Expected result: new member invited via email to project	passed

4. Create new milestone	
4.1 Create new milestone (menu: issue - milestones - new) Expected result: formular for entering milestone information visible	passed
4.2 Enter milestone information data (description, ...) and confirm Expected result: new milestone available	passed

5. Create new issue	
5.1 Create new issue (menu: issue - new) Expected result: formular for entering issue information visible	passed
5.2 Enter issue information data (description, ...) and confirm Expected result: new issue available	passed

6. Reassign issue	
6.1 Open existing issue (menu: issue) Expected result: existing issue information visible	passed
6.2 Change assignee to other user (edit assignee) and confirm Expected result: new assigned user to issue	passed

7. Assign issue to milestone	
7.1 Assign existing issue to milestone (menu: issue - milestones) Expected result: issue assigned to milestone	passed

8. Create new merge request and new branch	
8.1 Create new merge request (menu: merge request - new) Expected result: formular for entering merge request information visible	failed
8.2 Enter merge request information data (description, ...) and confirm Expected result: new merge request available	NA
8.3 Assign merge request to maintainer (edit dropdown) Expected result: maintainer assigned to merge request	NA
8.4 Create associated branch for git repository (implicitly created with merge request) Expected result: associated branch available	NA

9. Modify content of repository	
9.1 Clone repository content to local machine (`git clone <repository URL>`) Expected result: repository content available on local drive	NA
9.2 Checkout branch associated with merge request (`git checkout <branchname>`) Expected result: repository branch changed to `<branchname>`	NA
9.3 Modify content of the repository on local drive (change any file content) Expected result: file content differs to origin state	NA
9.4 Stage and commit modified files (`git commit -a -m <commit message>`) Expected result: commit successful	NA
9.5 Push commit to server (`git push origin HEAD`) Expected result: commit successfully pushed to server	NA

10. Approve and close merge request	
10.1 Close issue (menu: issue - close) Expected result: issue in state `closed`	NA
10.2 Prepare merge request for closing (remove draft prefix from name) Expected result: merge request name without the prefix "draft"	NA
10.3 Approve merge request (button approve) Expected result: merge request marked as `approved`	NA
10.4 Merge changes to master branch (button merge) Expected result: repository content changes available at master branch, merge request in state `merged`	NA

11. Close milestone	
11.1 Close milestone (menu: milestone - close) Expected result: milestone in state `closed`	NA

12. Create new release (release freeze)	
12.1 Create new release (menu: release - new) Expected result: formular for entering release information visible	NA
12.2 Assign existing milestone to release (dropdown milestones) Expected result: milestone assigned to release	NA
12.3 Enter release information data (description, ...) and confirm closing Expected result: new release available and closed	NA

5 Test Report - Customer Integration 3

Report ID: CM_TR_RC1_customer_integration_test_iteration3
Date: 03.12.2020
Tester: Team CM
Result: passed
Object under Test: CM server infrastructure - Release Candidate 1
Test Environment: • DELL OptiPlex 5050 • Windows 10 Professional • CPU Intel Core i5 • 8GB RAM • Wired LAN interface (2x; to Local Network, deactivated) • Oracle VM Virtual Box Version 6.1.16 r140961 • Webbrowser: Mozilla Firefox v83.0 (64bit)

Test result summary:	
Step	**result**
1. Login to GitLab web frontend	passed
2. Create GitLab project	passed
3. Add member to project group	passed
4. Create new milestone	passed
5. Create new issue	passed
6. Reassign issue	passed
7. Assign issue to milestone	passed
8. Create new merge request and new branch	passed
9. Modify content of repository	passed
10. Approve and close merge request	passed
11. Close milestone	passed
12. Create new release (release freeze)	passed

6 Test Steps - Customer Integration 3

1. Login to GitLab web frontend	
1.1 Access web frontend via URL "https://yeskia-hs.com" Expected result: GitLab login website visible	passed
1.2 User login on web frontend (user: root, password: Passw0rd!) Expected result: GitLab user dashboard visible	passed

2. Create GitLab project	
2.1 Create new blank project (button "new project") Expected result: Project start page of created project	passed

3. Add member to project group	
3.1 Open project member view (menu: project - members) Expected result: formular for assigning members visible	passed
3.2 Assign member to project (search user and assign with desired role description) and confirm Expected result: new member invited via email to project	passed

4. Create new milestone	
4.1 Create new milestone (menu: issue - milestones - new) Expected result: formular for entering milestone information visible	passed
4.2 Enter milestone information data (description, ...) and confirm Expected result: new milestone available	passed

5. Create new issue	
5.1 Create new issue (menu: issue - new) Expected result: formular for entering issue information visible	passed
5.2 Enter issue information data (description, ...) and confirm Expected result: new issue available	passed

6. Reassign issue	
6.1 Open existing issue (menu: issue) Expected result: existing issue information visible	passed
6.2 Change assignee to other user (edit assignee) and confirm Expected result: new assigned user to issue	passed

7. Assign issue to milestone	
7.1 Assign existing issue to milestone (menu: issue - milestones) Expected result: issue assigned to milestone	passed

8. Create new merge request and new branch	
8.1 Create new merge request (menu: merge request - new) Expected result: formular for entering merge request information visible	passed
8.2 Enter merge request information data (description, ...) and confirm Expected result: new merge request available	passed
8.3 Assign merge request to maintainer (edit dropdown) Expected result: maintainer assigned to merge request	passed
8.4 Create associated branch for git repository (implicitly created with merge request) Expected result: associated branch available	passed

9. Modify content of repository	
9.1 Clone repository content to local machine (`git clone <repository URL>`) Expected result: repository content available on local drive	passed
9.2 Checkout branch associated with merge request (`git checkout <branchpassedme>`) Expected result: repository branch changed to `<branchpassedme>`	passed
9.3 Modify content of the repository on local drive (change any file content) Expected result: file content differs to origin state	passed
9.4 Stage and commit modified files (`git commit -a -m <commit message>`) Expected result: commit successful	passed
9.5 Push commit to server (`git push origin HEAD`) Expected result: commit successfully pushed to server	passed

10. Approve and close merge request	
10.1 Close issue (menu: issue - close) Expected result: issue in state `closed`	passed
10.2 Prepare merge request for closing (remove draft prefix from passedme) Expected result: merge request passedme without the prefix "draft"	passed
10.3 Approve merge request (button approve) Expected result: merge request marked as `approved`	passed
10.4 Merge changes to master branch (button merge) Expected result: repository content changes available at master branch, merge request in state `merged`	passed

11. Close milestone	
11.1 Close milestone (menu: milestone - close) Expected result: milestone in state `closed`	passed

12. Create new release (release freeze)	
12.1 Create new release (menu: release - new) Expected result: formular for entering release information visible	passed
12.2 Assign existing milestone to release (dropdown milestones) Expected result: milestone assigned to release	passed
12.3 Enter release information data (description, ...) and confirm closing Expected result: new release available and closed	passed

E. Appendix: Integration Test RC2

Hochschule
Augsburg University of
Applied Sciences

Major Project 2020

Master Industrial Security
University of Applied Sciences Augsburg

Test Report
Integration Test RC2

Table 1: Student authors

Student	Student ID
Alexander Holzmann	2081881
Markus Kamm	2060929
Michael Wager	2081894

1 Test Report - Binary Integrity Surveillance

Report ID: CM_TR_RC2_integration_test_iteration1
Date: 15.12.2020
Tester: Team CM
Result: passed
Object under Test: Binary integrity surveillance - Release Candidate 2
Test Environment: • Lenovo IdeaPad 510s • Windows 10 Professional • CPU Intel Core i7 • 8GB RAM • WLAN network interface • Wired LAN interface • Oracle VM Virtual Box Version 6.1.8 r137981 • Webbrowser: Mozilla Firefox v83.0 (64bit)

Test result summary:	
Step	**result**
1. HW component not available	passed
2. No checksum available	passed
3. Checksum value change	passed
4. Checksum no value change	passed

2 Test Steps - Binary Integrity Surveillance

1. HW component not available	
1.1 configure comm adapter stub to represent production cell state "off" Expected result: comm adapter configuration state "off"	passed
1.2 start-up surveillance application Expected result: surveillance application running w/o error	passed
1.3 wait for check interval attempt Expected result: exp. log output "Facility seems to be down..." Expected result: no issue created, no e-mail creation	passed

2. No checksum available	
2.1 configure comm adapter stub to represent production cell state "no checksum" Expected result: comm adapter configuration state "no checksum"	passed
2.2 start-up surveillance application Expected result: surveillance application running w/o error	passed
2.3 wait for check interval attempt Expected result: log output "Indicator for compromised application" Expected result: issue created, e-mail created	passed

3. Checksum value change	
3.1 configure comm adapter stub to represent production cell state "new checksum" Expected result: comm adapter configuration state "new checksum"	passed
3.2 start-up surveillance application Expected result: surveillance application running w/o error	passed
3.3 wait for check interval attempt Expected result: log output "Change detected" Expected result: issue created, e-mail created	passed

4. Checksum no value change	
4.1 configure comm adapter stub to represent production cell state "no change" Expected result: comm adapter configuration state "no change"	passed
4.2 start-up surveillance application Expected result: surveillance application running w/o error	passed
4.4 wait for check interval attempt Expected result: log output "No changes detected" Expected result: no issue created, no e-mail created	passed

3 Test Report - Spare Part Replacement Process

Report ID: CM_TR_RC2_integration_test_iteration2
Date: 16.12.2020
Tester: Team CM
Result: passed
Object under Test: Spare part replacement process - Release Candidate 2
Test Environment: • Lenovo IdeaPad 510s • Windows 10 Professional • CPU Intel Core i7 • 8GB RAM • WLAN network interface • Wired LAN interface • Oracle VM Virtual Box Version 6.1.8 r137981 • Webbrowser: Mozilla Firefox v83.0 (64bit)

Test result summary:	
Step	**result**
1. Spare part replacement - default case	passed
2. Spare part replacement - date selection in future	passed
3. Spare part replacement - mandatory serial field empty	passed

4 Test Steps - Spare Part Replacement Process

1. Spare part replacement - default case	
1.1 Access webserver requesting frontend start page (`https://yeskia-hs.com:30443`) Expected result: View of start page with process submission form	passed
1.2 Select `Show BOM invetory list` Expected result: Page listing the content of the inventory database with correct content	passed
1.3 Check the value in field `Date of replacement` Expected result: Field contains the current calendar date	passed
1.4 Open dropdown list `User` Expected result: Dropdown list showing all defined replacement users.	passed
1.5 Open dropdown list `Material equipment identifier` Expected result: Field contains all available HW components of the facility	passed
1.6 Fill all fields with proper values for the selected part to replace Expected result: All fields contain proper values, form ready to submit	passed
1.7 Submit the replacement form (button `Submit`) Expected result: Modalwindow pop-up with text: *Data successfully saved*	passed
1.8 Check `BOM invetory list` for updated values Expected result: The selected part is updated with the passed values	passed
1.9 Verify for successful issue creation and mail notification Expected result: The GitLab issue is created with the desired content, a notification mail is sent to the submitter	passed

2. Spare part replacement - date selection in future	
2.1 Access webserver requesting frontend start page (`https://yeskia-hs.com:30443`) Expected result: View of start page with process submission form	passed
2.2 Select a date value pointing to the future in the field `Date of replacement` Expected result: desired date value displayed	passed
2.3 Fill all remaining fields with proper values for the selected part to replace Expected result: All fields contain proper values, form ready to submit	passed
2.4 Submit the replacement form (button `Submit`) Expected result: Modalwindow pop-up with text: *Date must not be in future*	passed

3. Spare part replacement - mandatory serial field empty	
3.1 Access webserver requesting frontend start page (`https://yeskia-hs.com:30443`) Expected result: View of start page with process submission form	passed
2.3 Fill all fields with proper values for the selected part to replace, but leave the field `Serial number - installed part` empty Expected result: All fields, excepting the serial number, contain proper values, form ready to submit	passed
2.4 Submit the replacement form (button `Submit`) Expected result: Modalwindow pop-up with text: *Please verify serial number, must not be empty for component: BMK*	passed

F. Appendix: System Integration Test RC2

Hochschule
Augsburg University of
Applied Sciences

Major Project 2020

Master Industrial Security

University of Applied Sciences Augsburg

Test Report
System Integration Test RC2

Table 1: Student authors

Student	Student ID
Alexander Holzmann	2081881
Markus Kamm	2060929
Michael Wager	2081894

1 Test Report - Binary Integrity Surveillance

Report ID: CM_TR_RC2_system_integration_test_iteration1
Date: 15.12.2020
Tester: Team CM
Result: passed
Object under Test: Binary integrity surveillance - Release Candidate 2
Test Environment: • DELL OptiPlex 5050 • Windows 10 Professional • CPU Intel Core i5 • 8GB RAM • Wired LAN interface (2x; to Local Network, deactivated) • Oracle VM Virtual Box Version 6.1.16 r140961 • Webbrowser: Mozilla Firefox v83.0 (64bit) • Festo CP Factory RASS-KUKA • Siemens Simatic S7 CPU1515-SP (plcRASS)

Test result summary:	
Step	**result**
1. HW component not available	passed
2. No checksum available	passed
3. Checksum value change	passed
4. Checksum no value change	passed

2 Test Steps - Binary Integrity Surveillance

1. HW component not available	
1.1 power off production cell Expected result: power indication lights off	passed
1.2 start-up surveillance application Expected result: surveillance application running w/o error	passed
1.3 wait for check interval attempt Expected result: exp. log output "Facility seems to be down..." Expected result: no issue created, no e-mail creation	passed

2. No checksum available	
2.1 power on production cell Expected result: power indication lights onr	passed
2.2 deploy application w/o "getChecksum" function Expected result: TIA indicating successful deployment	passed
2.3 start-up surveillance application Expected result: surveillance application running w/o error	passed
2.4 wait for check interval attempt Expected result: log output "Indicator for compromised application" Expected result: issue created, e-mail created	passed

3. Checksum value change	
3.1 power on production cell Expected result: power indication lights onr	passed
3.2 deploy application with "getChecksum" function Expected result: TIA indicating successful deployment	passed
3.3 start-up surveillance application Expected result: surveillance application running w/o error	passed
3.4 wait for check interval attempt Expected result: any log output	passed
3.5 re-deploy application with arbitrary change Expected result: TIA indicating successful deployment	passed
3.6 wait for check interval attempt Expected result: log output "Change detected" Expected result: issue created, e-mail created	passed

4. Checksum no value change	
4.1 power on production cell Expected result: power indication lights onr	passed
4.2 verify valid application deployed (refer test stage 3) Expected result: TIA indicating no checksum value change	passed
4.3 start-up surveillance application Expected result: surveillance application running w/o error	passed
4.4 wait for check interval attempt Expected result: log output "No change detected" Expected result: no issue created, no e-mail created	passed

3 Test Report - Spare Part Replacement Process

Report ID: CM_TR_RC2_system_integration_test_iteration2
Date: 15.12.2020
Tester: Team CM
Result: passed
Object under Test: Spare part replacement process - Release Candidate 2
Test Environment: • DELL OptiPlex 5050 • Windows 10 Professional • CPU Intel Core i5 • 8GB RAM • Wired LAN interface (2x; to Local Network, deactivated) • Oracle VM Virtual Box Version 6.1.16 r140961 • Webbrowser: Mozilla Firefox v83.0 (64bit) • Festo CP Factory RASS-KUKA • Siemens Simatic S7 CPU1515-SP (plcRASS)

Test result summary:	
Step	**result**
1. Spare part replacement - default case	passed
2. Spare part replacement - date selection in future	passed
3. Spare part replacement - mandatory serial field empty	passed

4 Test Steps - Spare Part Replacement Process

1. HW component not available	
1.1 power off production cell Expected result: power indication lights off	passed
1.2 start-up surveillance application Expected result: surveillance application running w/o error	passed
1.3 wait for check interval attempt Expected result: exp. log output "Facility seems to be down..." Expected result: no issue created, no e-mail creation	passed

2. No checksum available	
2.1 power on production cell Expected result: power indication lights onr	passed
2.2 deploy application w/o "getChecksum" function Expected result: TIA indicating successful deployment	passed
2.3 start-up surveillance application Expected result: surveillance application running w/o error	passed
2.4 wait for check interval attempt Expected result: log output "Indicator for compromised application" Expected result: issue created, e-mail created	passed

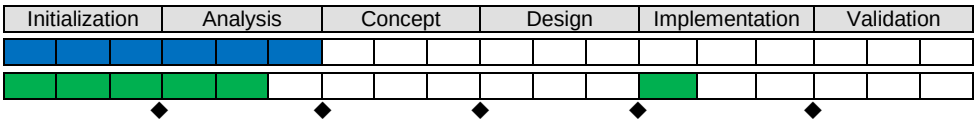
3. Checksum value change	
3.1 power on production cell Expected result: power indication lights onr	passed
3.2 deploy application with "getChecksum" function Expected result: TIA indicating successful deployment	passed
3.3 start-up surveillance application Expected result: surveillance application running w/o error	passed
3.4 wait for check interval attempt Expected result: any log output	passed
3.5 re-deploy application with arbitrary change Expected result: TIA indicating successful deployment	passed
3.6 wait for check interval attempt Expected result: log output "Change detected" Expected result: issue created, e-mail created	passed

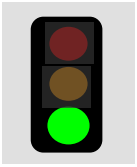
4. Checksum no value change	
4.1 power on production cell Expected result: power indication lights onr	passed
4.2 verify valid application deployed (refer test stage 3) Expected result: TIA indicating no checksum value change	passed
4.3 start-up surveillance application Expected result: surveillance application running w/o error	passed
4.4 wait for check interval attempt Expected result: log output "No change detected" Expected result: no issue created, no e-mail created	passed

G. Appendix: Project Status Reports

Project **MIS 2020 – Major Project** **Configuration Management**	**STATUS REPORT NO. 1**	**02.11.2020**
Project Start: 06.10.2020 Planned End: 22.12.2020 Project Lead: A. Holzmann	Reporting Period: 06.10.2020 – 30.10.2020	Reason: Milestone – Requirements Freeze

Current Status (target vs. actual)





Expected Results within the Reporting Period	achieved	not achieved
○ Project setup defined and established	☒	☐
○ Project definition and infrastructure installed (incl. project plan)	☒	☐
○ Scenario definition developed	☒	☐
○ Risk assessment evaluated and documented	☒	☐
○ Requirements specification created	☒	☐
○ Plant hardware infrastructure analysis finished	☒	☐
○ Plant software infrastructure analysis finished	☐	☒

Reasons for Deviation
○ Rescheduled the installation of the change management software framework (ticket system and version control management software) with the objective of early collecting modifications by other groups regarding the development path.

Further procedure	Responsible
○ Finalization of the software infrastructure analysis has to be postponed to the next phase	Team
○	
○	
○	

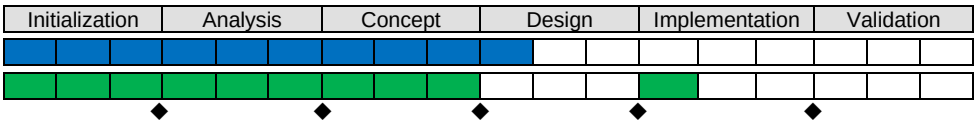
Project costs	Budget	Effort reporting period	Effort total	Remaining budget
External	0€	0€	0€	0€
Internal	79,200€	28,800€	28,800€	50,400€
Total	79,200€	28,800€	28,800€	50,400€

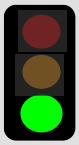
Notes
Budget calculation: hourly rate per person: 120€ 20h per week Team size: 3 developers Project duration: 11 weeks

Report author:	Receiver:	Date of next report:
Team config management	Prof. Dr. P. Richard	13.11.2020

Project **MIS 2020 – Major Project** **Configuration Management**	**STATUS REPORT NO. 2**	**17.11.2020**
Project Start: 06.10.2020 Planned End: 22.12.2020 Project Lead: A. Holzmann	Reporting Period: 01.11.2020 – 16.11.2020	Reason: Milestone – Concept Freeze

Current Status (target vs. actual)





Expected Results within the Reporting Period	achieved	not achieved
○ Plant software infrastructure analysis finished	☒	☐
○ Concept evaluation finished	☒	☐
○ Concept architecture finished	☒	☐
○ Concept functional specifications finished	☒	☐

Reasons for Deviation
○ N/A

Further procedure	Responsible
○ Further design and implementation of CM System components will focus on:	
○ Detection and Logging of unauthorized artifact changes	Team
○ Comprehensive Development Process	Team
○ Computer aided spare part replacement process	Team

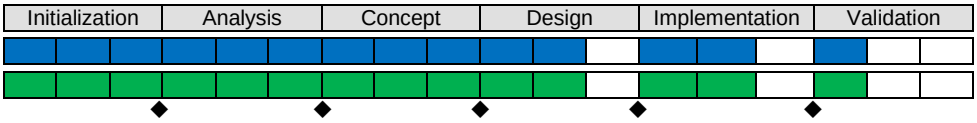
Project costs	Budget	Effort reporting period	Effort total	Remaining budget
External	0€	0€	0€	0€
Internal	79,200€	14,400€	43,200€	36,000€
Total	79,200€	14,400€	43,200€	36,000€

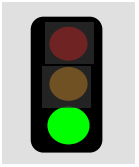
Notes
Budget calculation: hourly rate per person: 120€ additional 5h per week Team size: 3 developers Project duration: 5 weeks Proposed necessary budged increase: 9,000€

Report author:	Receiver:	Date of next report:
Team config management	Prof. Dr. P. Richard	04.12.2020

Project **MIS 2020 – Major Project** **Configuration Management**	**STATUS REPORT NO. 3**	**04.12.2020**
Project Start: 06.10.2020 Planned End: 22.12.2020 Project Lead: A. Holzmann	Reporting Period: 17.11.2020 – 04.12.2020	Reason: Milestone – Release Candidate 1

Current Status (target vs. actual)





Expected Results within the Reporting Period	achieved	not achieved
○ Design Phase RC1 (Comprehensive Development Process) finished	☒	☐
○ Implementation Phase RC1 finished	☒	☐
○ Validation Phase RC1 finished	☒	☐
○ Training course material creation started	☒	☐

Reasons for Deviation
○ N/A

Further procedure	Responsible
○ Design, Implementation & Validation RC2 (Detection and Logging of unauthorized artifact changes)	Team
○ Design, Implementation & Validation RC2 (Computer aided spare part replacement process)	Team
○ Training course material creation	Team
○	

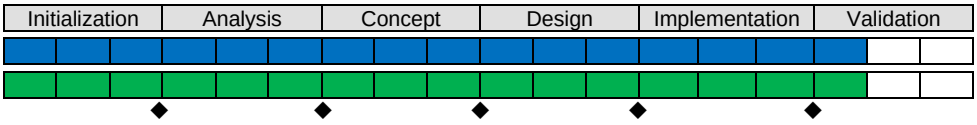
Project costs	Budget	Effort reporting period	Effort total	Remaining budget
External	0€	0€	0€	0€
Internal	79,200€	18,000€	61,200€	18,000€
Total	79,200€	18,000€	61,200€	18,000€

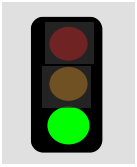
Notes

Report author:	Receiver:	Date of next report:
Team config management	Prof. Dr. P. Richard	22.12.2020

Project **MIS 2020 – Major Project** **Configuration Management**	**STATUS REPORT NO. 4**	**14.12.2020**
Project Start: 06.10.2020 Planned End: 22.12.2020 Project Lead: A. Holzmann	Reporting Period: 04.12.2020 – 14.12.2020	Reason: Milestone – Release Candidate 2

Current Status (target vs. actual)





Expected Results within the Reporting Period	achieved	not achieved
○ Design Phase RC2 finished	☒	☐
○ Implementation Phase RC2 finished	☒	☐
○	☐	☐
○	☐	☐

Reasons for Deviation
○ N/A

Further procedure	Responsible
○ Validation RC2 (Detection and Logging of unauthorized artifact changes)	Team
○ Validation RC2 (Computer aided spare part replacement process)	Team
○ Training course material creation	Team
○	

Project costs	Budget	Effort reporting period	Effort total	Remaining budget
External	0€	0€	0€	0€
Internal	79,200€	10,800€	61,200€	7,200€
Total	79,200€	10,800€	61,200€	7,200€

Notes

Report author:	Receiver:	Date of next report:
Team config management	Prof. Dr. P. Richard	22.12.2020