

Johannes Aas  
Sondre Jørgensen  
Sang Ngoc Nguyen

# Monitoring of Telephony Infrastructure for Receiving of Emergency Calls

Bachelor's thesis in Digital Infrastructure and Cyber Security  
Supervisor: Ernst Gunnar Gran

May 2023



Norwegian University of  
Science and Technology



Johannes Aas  
Sondre Jørgensen  
Sang Ngoc Nguyen

# **Monitoring of Telephony Infrastructure for Receiving of Emergency Calls**

Bachelor's thesis in Digital Infrastructure and Cyber Security  
Supervisor: Ernst Gunnar Gran  
May 2023

Norwegian University of Science and Technology  
Faculty of Information Technology and Electrical Engineering  
Dept. of Information Security and Communication Technology





# Monitoring of Telephony Infrastructure for Receival of Emergency Calls

Johannes Aas, Sondre Jørgensen and Sang Ngoc Nguyen

2023/05/21



# Abstract

*Helsetjenestens driftsorganisasjon for nødnett HF (HDO)* provides services that realize the Norwegian Emergency Reporting Service called "Nødnett". Monitoring traffic and performance is a crucial part of ensuring the critical infrastructure operation. The Norwegian emergency call network consists of a number of different components. *HDO* wanted to expand their ability to efficiently monitor a particular piece of hardware component that exists within their network called SBC EDGE. This thesis will cover the development of a so-called Prometheus exporter, a tool that collects data and helps convert them into metrics that *HDO* can use in their emergency call network monitoring system. This will contribute to monitoring trends and performance and provide information that will help secure the robustness of this infrastructure.

The thesis starts with an introduction to the project and then the background information necessary to understand it. Afterwards, the requirements of the developed product, methodology used for developing it, its design choices, implementation, its collected data, and usage will be presented. Finally, the decisions made throughout the project and security aspects related to the product will be discussed, followed by a conclusion of the project.





# Sammendrag

*Helsetjenestens driftsorganisasjon for nødnett HF (HDO) leverer tjenester som realiserer den nasjonale medisinske nødmeldetjenesten som heter "Nødnett". Monitorering av nettverkstrafikk og ytelse er en viktig del av prosessen for å drive denne kritiske infrastrukturen. Nødnettet består av flere komponenter og HDO ønsket å utvide deres muligheter til å monitorere et spesifikt komponent av deres nettverk kalt SBC EDGE. Denne rapport vil ta for seg utviklingen av en såkalt Prometheus exporter, et verktøy som henter data og konverterer disse til metrikker som HDO kan integrere som en del av deres monitoreringssystemer. Å monitorere trender, ytelse og informasjon vil bidra med å sikre robustheten av deres infrastruktur.*

Rapporten starter med en introduksjon av prosjektet og bakgrunnsinformasjonen som er nødvendig for å forstå den. Etterpå vil kravspesifikasjon av produktet, metoden brukt for å utvikle den, dens designvalg, implementasjon, data som blir samlet og bruksanvisning bli diskutert. Til slutt diskuteres beslutninger tatt underveis i prosjektet og sikkerhetsutfordringer knyttet til produktet som har blitt utviklet, etterfulgt av en konklusjon av prosjektet.



# Contents

<b>Abstract</b> . . . . .	<b>iii</b>
<b>Sammendrag</b> . . . . .	<b>v</b>
<b>Contents</b> . . . . .	<b>vii</b>
<b>Figures</b> . . . . .	<b>xi</b>
<b>Tables</b> . . . . .	<b>xiii</b>
<b>Code Listings</b> . . . . .	<b>xv</b>
<b>Acronyms</b> . . . . .	<b>xvii</b>
<b>Glossary</b> . . . . .	<b>xix</b>
<b>1 Introduction</b> . . . . .	<b>1</b>
1.1 Project Background . . . . .	1
1.2 Task Description . . . . .	2
1.3 Project Goals . . . . .	3
1.4 Constraints . . . . .	3
1.5 Project Scope . . . . .	4
1.6 Target Audience . . . . .	4
1.7 Delimitations . . . . .	5
1.8 Organization . . . . .	5
1.8.1 The Project Group . . . . .	5
1.8.2 Roles and Responsibilities . . . . .	5
1.9 Structure of the Report . . . . .	6
<b>2 Background</b> . . . . .	<b>9</b>
2.1 Theory and Technologies . . . . .	9
2.1.1 Event Monitoring . . . . .	9
2.1.2 Time Series Database (TSDB) . . . . .	11
2.1.3 Containers . . . . .	11
2.1.4 REST API . . . . .	13
2.2 Software . . . . .	15
2.2.1 Prometheus . . . . .	15
2.2.2 Grafana . . . . .	18
2.2.3 Docker . . . . .	20
2.2.4 The Go Programming Language . . . . .	21
2.2.5 The API of the SBCs . . . . .	23
<b>3 Requirements</b> . . . . .	<b>25</b>
3.1 Non-Functional Requirements . . . . .	25

3.2	Functional Requirements	26
<b>4</b>	<b>Methodology</b>	<b>29</b>
4.1	Choice of Programming Language	29
4.2	Tools Used for Development	30
4.3	Usage of a Development Model	30
4.4	Development and Testing	31
<b>5</b>	<b>Design</b>	<b>33</b>
5.1	Overview	33
5.2	Where the Exporter Fits Inside HDOs Infrastructure	35
5.2.1	Data Groupings and Collectors	36
5.2.2	Docker	36
5.2.3	Temporary Storage of Certain Data	36
5.3	Design of the Exporter	37
5.3.1	Collector Design	38
<b>6</b>	<b>Implementation</b>	<b>41</b>
6.1	Terms and Standards	41
6.1.1	Terms	41
6.2	Usage of Pointers	42
6.2.1	Error Handling	42
6.3	The Go Prometheus Package	43
6.3.1	The Describe Interface	43
6.3.2	The Collect Interface	43
6.4	Implementation of Code	44
6.4.1	The HTTP Package	45
6.4.2	Config Package	47
6.4.3	Database Package	49
6.4.4	Utils Package	52
6.4.5	Main Package	53
6.5	Collector Package	54
6.5.1	HTTP Handler and Probe Interface	54
6.5.2	Collectors	55
6.5.3	System Collector	59
6.5.4	Routingentry Collector	59
6.5.5	Comprehensive Changes to the Implementation	60
<b>7</b>	<b>Collected Data</b>	<b>63</b>
7.1	Availability	63
7.2	Why is the Data Collected?	63
7.3	Data Groups Used by the Exporter	64
<b>8</b>	<b>Usage</b>	<b>67</b>
8.1	Configuration of the Exporter	67
8.2	Installation and Deployment	68
8.2.1	Deployment of the Exporter as a Docker Image	68
8.2.2	Installation and Deployment Without Docker	69
8.3	Monitoring Metrics Produced by the Exporter	70

<b>9 Discussion</b>	<b>71</b>
9.1 Technical Discussion	71
9.1.1 Preexisting Security Measures	71
9.1.2 API Authentication	72
9.1.3 Security of the Code	73
9.1.4 Using the Exporter as Open Source	73
9.1.5 Issues That Were Revealed During Testing	74
9.2 Project Execution	75
9.2.1 Communication	75
9.2.2 Working Process	75
9.2.3 Planning	75
9.2.4 Meetings	75
9.2.5 Time Tracking	76
9.2.6 Report	77
9.2.7 Development Model	77
<b>10 Conclusion</b>	<b>79</b>
10.1 What Has the Group Achieved?	79
10.2 Further Work	79
10.2.1 Potential Improvements	79
10.2.2 Ways to Improve the Implementation	80
<b>Bibliography</b>	<b>81</b>
<b>A Data Collected</b>	<b>87</b>
<b>B Metrics Output</b>	<b>93</b>
<b>C Standard Agreement and Confidentiality Agreement</b>	<b>121</b>
<b>D The Project Plan</b>	<b>131</b>
<b>E Task Description</b>	<b>149</b>
<b>F Repository</b>	<b>155</b>
<b>G Minutes of Meeting</b>	<b>161</b>
<b>H Time Tracking</b>	<b>205</b>



# Figures

1.1	Data Flow and Exporter Function Inside of <i>HDO</i> 's Infrastructure . . .	2
2.1	The Process of Event Monitoring . . . . .	10
2.2	Container Example . . . . .	12
2.3	Container Orchestration . . . . .	13
2.4	REST API Example . . . . .	14
2.5	Multi Target Exporter . . . . .	16
2.6	Interaction Between a Prometheus Server and Exporter . . . . .	18
2.7	Grafana Dashboards . . . . .	19
2.8	SBC API . . . . .	23
5.1	Exporter Tasks . . . . .	34
5.2	Data Flow and Exporter Function in <i>HDO</i> 's Infrastructure . . . . .	35
5.3	Design of the Exporter . . . . .	37
5.4	Collector Design . . . . .	39
6.1	The Processes of the Collectors . . . . .	44
6.2	The Authentication and Data Collection Process . . . . .	46
6.3	Expiration of Data . . . . .	53
6.4	Implementation Overview . . . . .	56
6.5	The First Version of the Implementation . . . . .	60
6.6	The Current Implementation Version . . . . .	61
7.1	<i>HDO</i> 's Two Test SBCs Inside of Prometheus . . . . .	64
7.2	CPU Usage Inside of Prometheus . . . . .	64





# Tables

6.1	The Structure of the Table Storing Routing Data . . . . .	50
-----	---	----



# Code Listings

2.1	An example of a function in Go. . . . .	22
6.1	The describe interface in the Prometheus Go package. . . . .	43
6.2	The collect interface in the Prometheus Go package. . . . .	44
6.3	The function for connecting to an SBC. . . . .	45
6.4	The function for retrieving a session cookie before expiration. . . . .	45
6.5	The function for retrieving API data. . . . .	47
6.6	Example of a struct used by the collectors. . . . .	47
6.7	Describing the configuration file. . . . .	47
6.8	Reading the configuration file. . . . .	48
6.9	Returning SBC hosts' configuration used by the collectors. . . . .	48
6.10	Retrieving session cookie from database. . . . .	49
6.11	Inserting session cookies into the database. . . . .	49
6.12	Storing routingtables and entries into the database. . . . .	50
6.13	Retrieving routing data from the database. . . . .	51
6.14	Fetching chassis information from either an SBC or database. . . . .	52
6.15	The function Expired(). . . . .	53
6.16	The main function. . . . .	54
6.17	The function Probehandler(). . . . .	54
6.18	Interface Probe and Prometheus interfaces. . . . .	55
6.19	The Linecard collector. . . . .	57
8.1	Layout of the configuration file. . . . .	68



# Acronyms

**HDO** *Helsetjenestens driftsorganisasjon for nødnett HF.* iii, v, xi, 1–6, 10, 25–27, 30–33, 35, 36, 51, 57, 59, 60, 63, 64, 70–74, 79, 80, 149

**NTNU** *Norwegian University of Science and Technology.* 4–6

**API** Application Programming Interface. xix, 5, 13, 23, 27, 31, 33, 36–38, 42, 47, 49, 51, 56, 57, 59, 64, 70, 72–74, 87

**CPU** Central Processing Unit. 64, 73, 74, 80

**DIGSEC** Digital Infrastructure and Cyber Security. 5

**DoS** Denial-of-Service. 72

**HTTP** Hypertext Transfer Protocol. xix, 13–15, 18, 23, 31, 36–38, 42, 44, 45, 47, 49, 52–55, 67

**IP** Internet Protocol. 2, 26, 35, 45, 49, 50, 52, 59, 73

**ISDN** Integrated Services Digital Network. 2, 35

**JSON** JavaScript Object Notation. 14

**PHP** PHP: Hypertext Preprocessor. xix, 27, 36–38, 45, 52, 60

**REST API** Representational State Transfer Application Programming Interface. xix, 1–3, 5, 13, 14, 23, 25, 31, 33, 36, 63, 72, 73

**SBC** Session Border Controller. iii, v, xix, 1–5, 17, 23, 25, 26, 31–33, 35–38, 42, 45, 49, 51, 52, 54, 56, 57, 59–61, 63, 64, 66, 69, 72–74, 79, 80

**SIP** Session Initiation Protocol. 2, 35

**SQL** Structured Query Language. 49, 72

**TLS** Transport Layer Security. 73

**TSDB** Time Series Database. 11, 15

**URI** Uniform Resource Identifier. 14

**URL** Uniform Resource Locator. 22, 23, 36, 42, 43

**VPN** Virtual Private Network. 71

**XML** Extensible Markup Language. 23, 33, 38, 47, 56, 57

**ZTA** Zero Trust Architecture. 71

# Glossary

**PHP Session Cookie** When connecting to an SBCs REST API through HTTP, it returns a PHP session cookie that is required when fetching data from the API. 27, 36–38, 45, 52, 60

**Prometheus Collector** A Prometheus collector, also called a probe, is in this project intended to describe each component within the exporter responsible for collecting a specific set of data and format them into metrics. In the case of this bachelor's thesis one collector is for instance responsible for collecting system resource usage data from all the SBCs. 26, 27, 36, 42, 43, 59, 64

**Prometheus Exporter** A Prometheus exporter is a scripted program that functions as a custom data collector, collecting data from a specific source and formatting it into metrics readable by Prometheus. 25–27, 43

**Prometheus Server** Used in this project to refer to the server or servers that collect and store data from one or more Prometheus exporters. 25, 43

**Ribbon Communications** The manufacturer [1] of the SBCs, its REST API and associated documentation. This documentation is not accessible without being a customer of Ribbon Communications, but it contains more than 1000 pages of information about data that can be collected and how to use the API. 3, 25, 45

**Scrape** A scrape in Prometheus terms, is each time a Prometheus server or other processes sends a HTTP-request to an exporter to collect data from this exporter. 25, 35, 42, 43, 59





# Chapter 1

## Introduction

This chapter will introduce the background of the project, the task at hand, the goals of the project, constraints, scope, delimitations, organization, and thesis structure.

### 1.1 Project Background

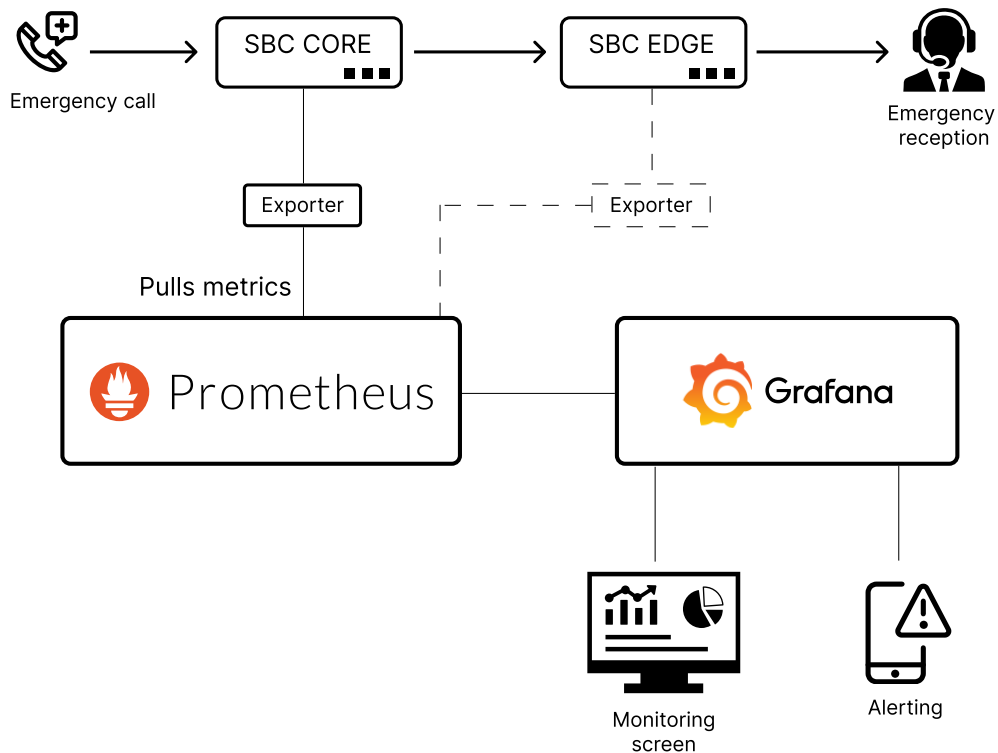
*Helsetjenestens driftsorganisasjon for nødnett HF (HDO)* [2] is a Norwegian organization responsible for delivering various services that realize the Norwegian Emergency Reporting Service called "Nødnett" [3]. *HDO's* goal is to deliver secure, reliable, and user-friendly services for emergency networks in all municipalities of Norway.

In recent years, *HDO* has rebuilt its infrastructure to accommodate for the Norwegian emergency calls and created solutions to analyze and uncover errors within the network in real time. This network consists of a number of components and monitoring solutions which are a crucial part of operating this infrastructure. The focus of this project is on a particular hardware component, the so-called Session Border Controller (SBC) [4]. SBCs are an essential part of *HDO's* infrastructure for handling emergency calls. A SBC is essentially a router that protects communication between two parties over networks [5]. *HDO* wish to use an existing Representational State Transfer Application Programming Interface (REST API) that can communicate with these SBCs. This REST API can be utilized to retrieve various useful data regarding the traffic that this component handles, as well as performance data. Collecting and monitoring these data can prove useful for monitoring trends, more efficiently uncovering potential errors, and preventing errors or performance problems in advance. They wish to do so by collecting data from the SBCs with this REST API, by using a combination of different technologies such as container technology such as Docker [6] and open-source tools used for monitoring and storing data such as Grafana [7], Prometheus [8] and Loki [9].

As of January 2023, *HDO* has not utilized this REST API, which is available on the SBCs. Therefore, *HDO* wish to make use of this REST API by collecting data to be used to monitor both the trends in emergency calls and the performance trends of these SBCs.

## 1.2 Task Description

The task of this project is to integrate an additional solution to *HDO*'s infrastructure called an exporter. *HDO* already have such an exporter for their SBC CORE network, but not their SBC EDGE network as shown in figure 1.1. Furthermore, the figure illustrates *HDO*'s infrastructure for receiving emergency calls. The SBC CORE is the main network used for handling emergency calls and uses Internet Protocol (IP)/Session Initiation Protocol (SIP) for communication. The SBC EDGE converts from IP/SIP to Integrated Services Digital Network (ISDN), which is what emergency receptions use to communicate.



**Figure 1.1:** This figure shows the flow of data and the exporters function within *HDO*'s infrastructure. The dotted lines illustrates the exporter *HDO* wish to be built.

They want to use the exporter to collect data from the SBCs about its call data and performance data. They wish to do so using the existing REST API provided by Ribbon Communications, the manufacturer of the SBCs, and feed these data to Prometheus. These data have to be stored in such a way that allows them to be used by employees at *HDO* to monitor and alert trends in a monitoring application called Grafana. The solution must be able to be deployed on a virtual machine inside a Docker container. In addition, the solution must be able to collect data from multiple SBCs at the same time and be configurable in a way that allows the user to specify which SBCs it will run on, and more.

## 1.3 Project Goals

This section will outline the group goals for this project in terms of effect, result, and learning goals.

### Effect Goals

By developing the exporter, the goal in terms of effect is to utilize the existing REST API to make emergency call data and SBC performance data more accessible. This will make monitoring trends in the emergency call network easier and more efficient.

### Result Goals

The exporter will assist the *HDO* employees in identifying potential faults in the SBCs, so they can address the issues proactively and prevent potential downtime or quality loss. Being able to uncover faults or errors in the network using the exporter will allow employees at *HDO* to respond to issues sooner, which will facilitate results in improved reliability and quality of emergency calls.

### Learning Goals

After working on this project, the group's learning goal is to gain knowledge of how to secure and operate social critical infrastructure and how to contribute to Norway's emergency preparedness by making emergency call data more accessible and monitorable. The group also wishes to acquire more knowledge of tools used to develop, operate, and uncover trends in applications and infrastructure.

## 1.4 Constraints

### Time Constraints

The time constraints for the project range from the first school day after the Christmas holidays of 2022 (11th of January 2023) and last until the projects deadline

(22nd of May 2023). The project report has to be delivered before the 22nd of May 2023.

### **Technological Constraints**

The product that is going to be developed has several technological constraints. First, the solution should be able to run inside a Docker container. Additionally, the solution should be configurable to allow users to choose which SBCs to collect data from. Data collection from multiple SBCs at the same time is also required.

## **1.5 Project Scope**

This project covers multiple subject areas; such as technological infrastructure, which includes programming and using technologies such as Prometheus and Docker, but also understanding the complex network that handles emergency calls.

## **1.6 Target Audience**

This project consists of both a report and a product that the employer can use as part of their own infrastructure. However, this project could be of interest to other people as well.

### ***Norwegian University of Science and Technology (NTNU)***

The report for the project will be of interest to other students and academic purposes.

### **Employer**

This project report will be of interest to our employer *HDO*, not to mention their particular interest in the product that the group has developed for them, as it improves their ability to monitor their infrastructure.

### **Third Parties**

These SBCs are widely used in leading cloud-based services. Services such as Microsoft Teams, Zoom Phone, Ring Central, Cisco, Genesys, Five9, Nice CXone, Talkdesk, Mida, AnyWhere365, and others are known to use these SBCs [4]. Therefore, service providers such as these might find the product both interesting and useful.

## 1.7 Delimitations

The project will be limited to the creation of a product to monitor data collected from the SBC REST API. This product creates a system for collecting data from emergency call data and SBC performance data and storing the data in Prometheus to monitor future potential events related to errors or weaknesses within HDO's infrastructure regarding the SBCs. The group is explicitly going to work with collecting data from the SBC REST API. The solution could be developed to collect data from other APIs as well, but the decision not to was made because it is not an integral part of making the solution work for the API specified in the task description (see Appendix E).

## 1.8 Organization

This section describes the project group and other parties involved in this project.

### 1.8.1 The Project Group

The project group consists of three students from *NTNU* in Gjøvik who are following the Bachelor's degree program Digital Infrastructure and Cyber Security (DIGSEC) [10]. The group has a broad coverage of knowledge from all mandatory courses the group has taken as part of the study plan. The group's knowledge is not specialized in any way; the group only has general knowledge of different technological concepts with limited experience working on larger projects such as this one. However, the most relevant knowledge for this specific project would be container technology using Docker and general basic programming.

### 1.8.2 Roles and Responsibilities

During the making of the project plan (see Appendix D) in the early stages of January 2023, the group chose to delegate every group member their own respective roles and responsibilities. Every group member is responsible for developing the product and writing the project report. In addition, we have a few special roles that we delegated to the members of the group.

#### Group leader

The group leader for the project was decided by the group members to be Johannes Aas. He is responsible for delegating tasks to all members and for making sure the project is moving forward and progressing.

#### Contact person

The contact person is responsible for communication between the group, the supervisor and the employer. For this project that person is Sondre Jørgensen.

### **Secretary**

The secretary for the project is Sang Nguyen, he is responsible for making the minutes of the meetings and preparing questions and topics of discussion for meetings with the supervisor and employer.

### **Supervisor**

The supervisor for the group project is Ernst Gunnar Gran, who is an associate professor working in the Department of Information Security and Communication Technology [11] at *NTNU*. His responsibility is to provide guidance and feedback throughout the project period.

### **Employers**

The employer contact person for the project is Stig Atle Haugen, who works as Operations Engineer ICT at *HDO*. His responsibility is to decide the requirements for the product that the group will develop. Additionally, Stig Atle Haugen gave the group advice and feedback on the development of the product. Haugen should be contacted for assistance in case of any technological issues with respect to the servers or virtual machines provided by *Helsetjenestens driftsorganisasjon for n dnett HF (HDO)*.

## **1.9 Structure of the Report**

The project report consists of 10 chapters in total. The chapters should be read successively, starting from Chapter 1 and ending in Chapter 10. In addition to these chapters, the report also includes a list of figures, tables, glossaries, and acronyms prior to chapter one, and references and appendixes after chapter 10. The following sections give a brief description of each chapter.

### **Chapter 1 - Introduction**

The introduction provides a brief description of the background of the project. It also includes the task description, project goals, organization, constraints, delimitations, target audience, and thesis structure.

### **Chapter 2 - Background**

The background section has been divided into two sections, Theory and Technologies and Software. The purpose of this chapter is to provide the background knowledge necessary to understand the content presented in the subsequent chapters.

### **Chapter 3 - Requirements**

This chapter will describe the requirements that were established for the product. The chapter consists of two sections; non-functional and functional requirements.

### **Chapter 4 - Methodology**

This chapter will discuss the programming language chosen to develop the exporter, the tools used to do so, and how development and testing were carried out.

### **Chapter 5 - Design**

The design chapter describes the overall design of the product, including its main components and the inner workings of the architecture.

### **Chapter 6 - Implementation**

This chapter will discuss the code for implementation, the choice of features, and the details of the architecture that has been developed. Additionally, we will discuss why it was decided to change the implementation to an overall better and more scalable version.

### **Chapter 7 - Collected Data**

This chapter will discuss the availability of the data that are collected, why they are collected, and what kind of value they create.

### **Chapter 8 - Usage**

This chapter explains how to install, configure, and run the exporter with or without Docker.

### **Chapter 9 - Discussion**

This chapter will discuss the security aspects related to the product that has been developed and some technical difficulties discovered during testing and how they were addressed. Additionally, different aspects of the project related to the process of creating the product and thesis will be discussed.

### **Chapter 10 - Conclusion**

This chapter concludes the report and discusses what the group has been able to achieve. Furthermore, it will discuss further work related to the final result and the possible improvements that can be made to the product.





## Chapter 2

# Background

The background section has been divided into two sections, Theory and Technologies, and Software. The purpose of this chapter is to provide the background knowledge necessary to understand the content presented in the subsequent chapters.

### 2.1 Theory and Technologies

The first section of the background will explain different concepts related to both the product developed by the group and the bachelor thesis you are currently reading.

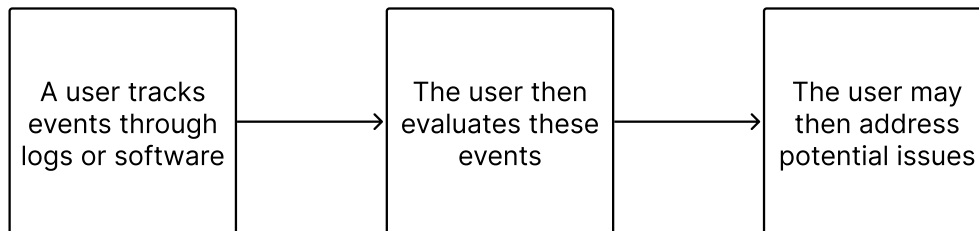
#### 2.1.1 Event Monitoring

To better understand the report and the product, it is essential to understand the concept of event monitoring. Event monitoring in computer science involves the tracking and evaluation of events that occur within a system [12]. Event monitoring is frequently used in computer networks to identify and address security concerns, performance difficulties, and other potential issues. By doing this, the system administrator can address problems before they cause problems for the entire system. Events such as these can occur when certain values exceed or fall below a certain threshold.

One way of conducting event monitoring is to use log files to record events as they occur in real time. There are different software and hardware parts of a system that can be used to produce log files for event monitoring, such as operating systems, network devices, and applications. After these logs have been produced, they may be examined by a user (often described as a subscriber) to look for trends and abnormalities that may point to potential issues or concerns.

Another way of monitoring events is by using specialized software tools created especially for this purpose. An example of such a tool is Prometheus [8]. We will learn more about this tool later in the technologies section of the background chapter. These specialized tools are able to perform tasks such as real-time monitoring of network traffic, system performance, and other important indicators.

Figure 2.1 shows the basic process of practicing event monitoring. It starts with a user tracking various events through the two aforementioned methods of logs or special software. Afterwards, the user reviews and evaluates these events to find potential issues. The final step of the process is to address the potential issues gathered from the previous evaluation step.



**Figure 2.1:** This figure illustrates the basic process of event monitoring.

In addition, these tools can also send warnings to the user when certain thresholds are reached or suspicious activity is discovered. An example of this is when the temperature of a CPU exceeds a certain threshold. Lastly, these warnings may be utilized to start automated responses called alerts to notify operators so that they can address the issue manually.

In addition to detecting and responding to problems, event monitoring can also help users improve system performance and optimize resource usage. This can be achieved by collecting and tracking key values such as CPU usage, network bandwidth, and disk I/O. By doing this, system administrators can identify bottlenecks and other issues that may impact performance and take the necessary steps to address them.

In summary, event monitoring is the act of tracking and evaluating event occurrences within a system. It may be used to improve the security, performance, and reliability of computer systems. The two main ways of doing this are by using specialized tools such as Prometheus or by using log files to record events in real time. Therefore, it is a very important concept for people operating critical systems such as our employers *HDO*, because it allows them to ensure that their systems are running optimally by monitoring the system and addressing potential issues.

### 2.1.2 Time Series Database (TSDB)

To understand how event monitoring is achieved, first we must talk about an essential concept that is used for this. The concept is called a Time Series Database (TSDB) and it is a specialized type of database that is designed both to store and to handle timestamped data [13].

Timestamped data are types of data that have a time and a date attached that indicate when that data was generated. The TSDBs are commonly used to store data that change over time, such as performance data and financial data. For this reason, TSDBs are frequently a part of applications or systems where data needs to be analyzed in real time, such as in a monitoring system. Examples of applications that use TSDBs are Prometheus [8], MongoDB [14], Graphite [15] and InfluxDB [16], among others.

One of the key characteristics of a TSDB is the ability to handle large volumes of data efficiently. This means that they are well suited for tasks such as when one needs to monitor large amounts of data from a large system. Compared to a traditional database, TSDBs are much better suited for handling and storing large amounts of data [17]. This is because time series databases are optimized for this task and have been purpose-made for storing and querying time-stamped data in real time.

Another key feature of a TSDB is the ability to handle data with high velocity [18]. High-velocity data is data that arrives at a high frequency, such as sensor data, performance data, log data, or financial market data. TSDBs typically use specialized algorithms to handle the high volume and velocity of data.

In addition to efficient data storage and retrieval, time series databases also offer advanced and efficient querying and analysis capabilities. Some of the commonly used query operations in time series databases include aggregation, filtering, and grouping. The queries can be used to extract valuable insights from the data for analysis in real time.

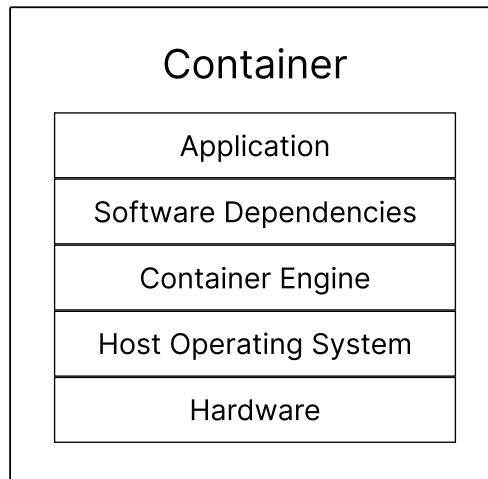
In conclusion, TSDBs are specialized databases that are designed to store, handle, and retrieve time-stamped data. They are able to handle large volumes of high-velocity data efficiently. Lastly, they are well suited for tasks such as monitoring and analysis, thanks to the advanced and efficient querying operations available when working with them.

### 2.1.3 Containers

Containerization is a technology that has been developed to make the deployment and management of software applications easier and more efficient [19]. A container is known as a lightweight and portable package that consists of an

application and its dependencies that have been encapsulated together into one unit that runs on a container engine on the host operating system.

Figure 2.2 shows the basic components of a containerized application. A container encapsulates applications and software dependencies. These two components then run on top of a container engine that runs on top of a host operating system that finally runs on some hardware.



**Figure 2.2:** This figure shows an example of a container.

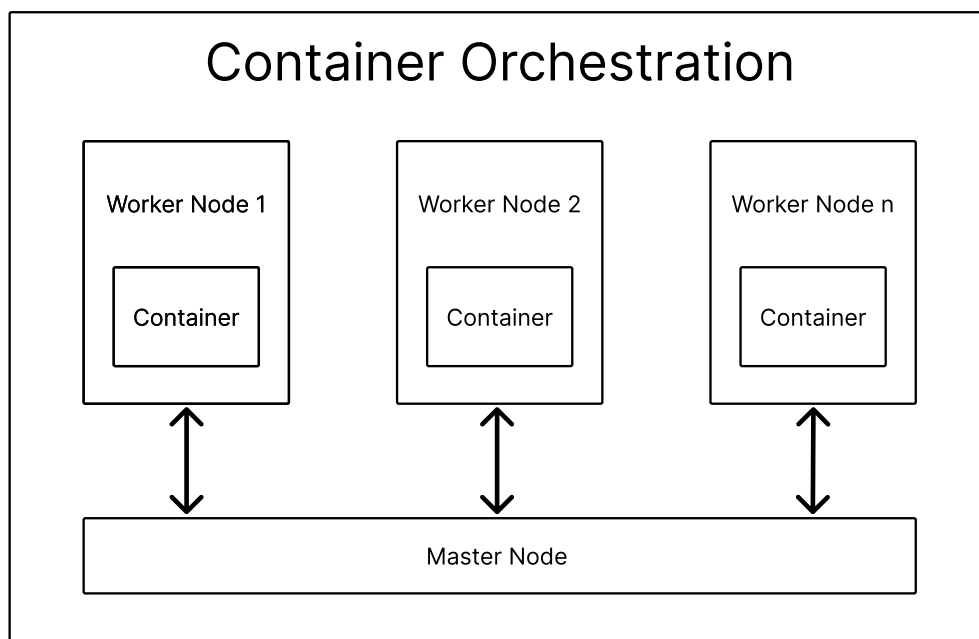
The use of containers has several benefits, including increased efficiency, scalability, and portability [20]. These containers can be easily deployed and run on any machine that supports containerization technology, such as Docker [21]. By encapsulating the application and its dependencies into one unit, containers can provide a consistent and reproducible environment for running the application regardless of the underlying operating system or hardware. This makes it much easier to deploy applications across different environments, such as development, testing, and production.

Another advantage of containers is that they are lightweight and consume minimal resources compared to virtual machines [22]. This allows multiple containers to be run on a single machine without causing significant performance overhead. This enables applications to be scaled up or down quickly and easily to meet changing demands.

Yet another advantage of containerization is the ability to isolate applications from each other. Each container runs in its own isolated environment, with its own resources and network connections, reducing the risk of conflicts between different applications. This also reduces the risks of one application causing damage to another if one of them gets infected with some kind of virus.

Containerization can be used together with container orchestration tools [23], such as Kubernetes [24], to manage the deployment, scaling, and monitoring of containerized applications. These tools provide a way to automate many of the tasks associated with managing containers, making it easier to deploy and scale applications across large, distributed environments.

Figure 2.3 illustrates the orchestration of multiple containers. Multiple containers may run on a physical or virtual machine, called a worker node. All of the worker nodes part of a container orchestration are connected to a node that manages these worker nodes, this node is called the master node.



**Figure 2.3:** Example of orchestration of multiple containers.

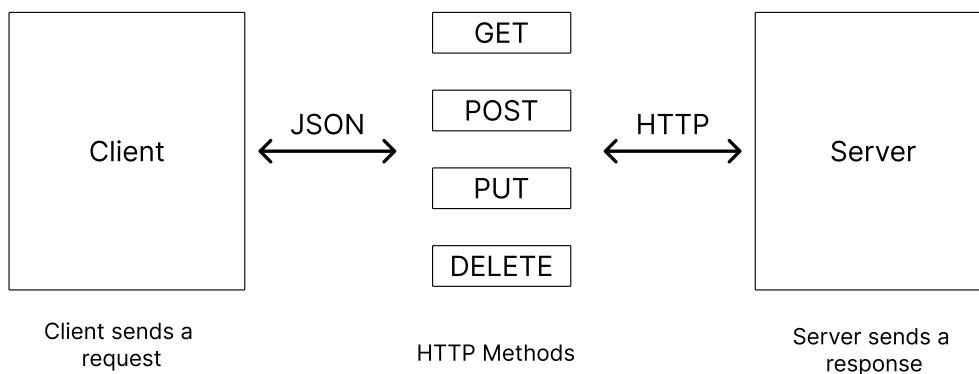
In conclusion, containerization is a powerful technology that provides a lightweight and efficient way to deploy and manage software applications. Its ability to provide portability, scalability, and isolation makes it an essential tool for software development and deployment.

#### 2.1.4 REST API

Representational State Transfer Application Programming Interface (REST API) is a widely used software architectural style to design web-based software systems [25]. It is based on the HTTP protocol and works with the interaction between the client and the server applications. An API that uses the REST architecture is often described as a RESTful API. RESTful APIs provide a standardized way of accessing and manipulating resources over the Internet. For this reason, they are

often used to build modular and scalable web applications.

Figure 2.4 shows a client and a server communicating using a REST API. In this figure, a client sends different requests to the server using the different HTTP methods that the REST API provides to the server. The information sent to the server will then perform different tasks, depending on the type of HTTP method used. For example, if a user uses the GET method, then they are requesting to receive some type of data from the server. The client uses the JavaScript Object Notation (JSON) format to use the HTTP methods in the REST API. Communication between the server and REST API uses HTTP.



**Figure 2.4:** This figure illustrates an example of a system with an integrated REST API solution for manipulating and accessing resources.

The main design principles of the REST architecture are statelessness, uniform interface, caching, layered system, and client-server communication [25]. A stateless system means that each request from the client must contain all the necessary information to complete the request, without relying on any previous state or context from the server. This allows for scalability, because it means that the server can process requests independently and without any dependencies on previous requests.

Uniform interface is a second key principle of RESTful architectures. It defines a consistent way to access and manipulate resources. This includes using HTTP operations such as GET, POST, PUT, and DELETE, among others, to represent different operations on resources. RESTful architectures also use Uniform Resource Identifier (URI) to identify resources [25]. This provides a standardized and predictable way for clients to interact with the API, making it easier to understand and use.

Caching is another important principle of RESTful architectures. It allows the client to store and reuse the responses. This can improve performance and reduce the load on the server, since the client can retrieve cached responses instead of requesting them from the server every time.

The layered system architecture is another key principle of REST. It allows the separation of responsibilities between different components of the system. This can improve scalability, as different layers can be added or removed without affecting other layers. Additionally, this can also improve security by adding layers of protection between the client and the server.

Lastly, the client-server architecture is a very important principle of RESTful architectures. It separates the responsibilities of the client and the server applications. This allows for decoupling of the presentation and logic layers, making it easier to maintain and update the system over time.

In summary, a REST API is a software architectural style for building web-based software systems. Its key design principles include statelessness, uniform interface, caching, layered system, and client-server architecture. This architectural style of software provides a standardized and scalable way of accessing and manipulating resources on the Web.

## 2.2 Software

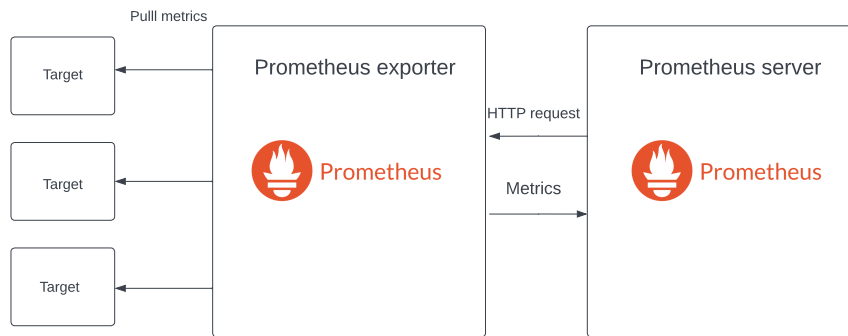
This chapter will go into detail about the specific software that has been used to develop the product.

### 2.2.1 Prometheus

Prometheus is an open source monitoring system that was developed by SoundCloud in 2012 [8]. It is a system that is used to track and collect information from different applications and systems. By doing this, Prometheus is able to provide users with valuable information about the functionality and status of a system that a user is currently monitoring.

Prometheus functions as a TSDB that collects and stores metric data from various targets [26]. It uses a pull-based model, where clients, called exporters, expose metrics over Hypertext Transfer Protocol (HTTP) [8] and are periodically scraped by Prometheus. The targets of Prometheus may be any application or service that exposes different values in a compatible format.

Figure 2.5 shows an example of a Prometheus exporter that pulls metrics from various targets and sends them to a Prometheus server where they are stored. A more detailed explanation of how this works will be given later in this subsection.



**Figure 2.5:** This figure shows an example of a Prometheus exporter that collects metric data from various targets.

Additionally, Prometheus supports a flexible query language called PromQL [8], which allows users to write queries to extract and analyze data from the time-series database. PromQL supports a wide range of operations, including aggregation, arithmetic, and filtering. This makes it a powerful tool that can be used to analyze data.

Prometheus also includes a number of other features, such as alerting, graphing, and dashboarding. The alerting system allows users to define rules that trigger alerts when certain conditions are met, for example, when a metric exceeds a threshold or when a service goes down. The graphing and dashboarding features enable users to visualize and explore the data in real-time. This helps to identify trends, patterns, and anomalies.

### Metrics in Prometheus

A core component of Prometheus is metrics. Examples of metrics may include values like CPU use, memory consumption, request delay, error rate, etc. They are numerical values that describe some element of a system or application [27]. In most cases, metrics are gathered periodically as part of a so called scrape and kept as time-series data. They are saved as a data point that consists of a timestamp and a value. Counters, gauges, histograms, and summaries are the four types of metrics that Prometheus provides. Different aspects of a system's behavior are captured by each kind of measure. Each metric will be described in the following paragraphs.



The first metric that Prometheus provides is called a counter. A counter is a metric that represents a monotonically increasing value. A monotonically increasing value is a value that either stays at a constant value or increases; they may not decrease. Examples of counters may include the number of requests processed or the number of errors encountered. For this reason, counters are typically used to measure the rate of events over time. Counters may only go up, however, they may decrease to a value of zero if the Prometheus server restarts.

The second metric that Prometheus provides is called a gauge. A gauge is a metric that represents a value that can increase or decrease. An example of a gauge value is the CPU utilization of a system or the number of active connections at any one time on a SBC. The gauges do not depend on each other, and this means that their values can change independently of each other. Gauges are therefore frequently used to assess a system's or application's current status.

The third metric that Prometheus provides is called a histogram. A histogram is a metric that represents the distribution of multiple values over time. These can, for example, be request latency or response size. They are typically used to measure the distribution of events over time. Histograms are divided into sections called buckets, which represent ranges of values. The different buckets have their own counts of data points that fall within its range.

The fourth and final metric provided by Prometheus is called a summary. Summaries calculate percentiles based on a sliding time window. They are commonly used to measure the distribution of events over time similarly to histograms. However, a key difference between a summary and a histogram is that a summary provides a more accurate representation of the distribution of values. Another difference is that summaries are more computationally expensive compared to histograms, they are more accurate, as they have to calculate percentiles based on a sliding time window.

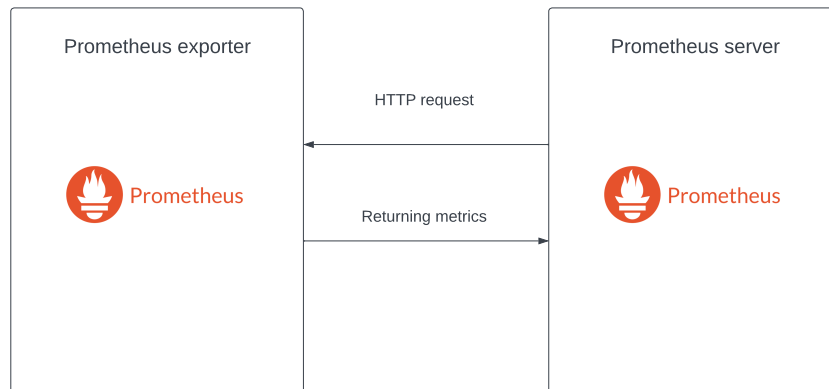
### **Prometheus Exporter**

A Prometheus exporter, the main subject of this Bachelor's thesis, is a custom made application that fetches and defines data from one or more sources and provides them to a Prometheus server, where they can be stored and used for monitoring [28]. There is a wide range of open-source Prometheus exporters available that can be used to monitor specific sources of data. A common example is a node exporter, which collects system resource data for a host.

Prometheus exporters are applications that can serve its data through either a push mechanism to the Prometheus server, or through a pull mechanism. Exporters that work with pull mechanisms is the most common way for exporters to

serve Prometheus metrics. It works in such a way that whenever a HTTP request is made to the exporter, the exporter runs its collectors and, as a result, returns metrics.

The Prometheus server illustrated in Figure 2.6 is responsible for making these HTTP requests, which are called scrapes. The time between the scrape intervals is defined in the Prometheus server configuration file. Whenever a scrape is executed, the exporter runs its collect routines, collecting new data, and provides them to the Prometheus server through HTTP. A Prometheus server can request metrics from multiple exporters simultaneously, but each exporter instance should only serve data to one server. This is to avoid data corruption caused by more than one scrape occurring at the same time.



**Figure 2.6:** This figure shows an example of a Prometheus server making a HTTP request to a Prometheus exporter, that returns metrics to the Prometheus server.

### 2.2.2 Grafana

Grafana is an open source data visualization and monitoring tool that enables users to query, visualize, alert, and understand metrics [29]. It provides a user-friendly interface to visualize time series data from multiple data sources such as MySQL [30], Graphite [15], InfluxDB [16], Prometheus [8], Elasticsearch [31], and more. Grafana provides a simple and intuitive interface that allows users to create custom dashboards and panels that display data in a variety of formats, including graphs, tables, and gauges. A Grafana set-up with different dashboards that display various data in various formats is shown in Figure 2.7.



**Figure 2.7:** This figure shows an example of Grafana with different dashboards.

One of the key features of Grafana is the support for multiple sources of data, such as popular databases like MySQL, PostgreSQL, InfluxDB, and Prometheus. This allows users to collect data from multiple sources and put it into a single dashboard, providing a full view of their systems and applications. Grafana also supports a variety of data collection methods, like pull-based metrics collection through the use of plugins.

Another key benefit of Grafana is its ease of use. The platform provides a simple and intuitive interface that allows users to quickly create custom dashboards and panels. This makes it a great tool for both technical and non-technical users who need to monitor and analyze data. Additionally, Grafana's support for a wide range of data sources and collection methods makes it a flexible and versatile platform that can be used in a variety of environments.

Grafana's architecture is designed to be highly modular and extensible. The platform consists of a core engine that provides basic functionality such as data collection and visualization. Additionally, it provides a plugin system that allows users to extend the functionality of the platform with custom plugins.

In summary, Grafana is a powerful and flexible platform for monitoring and data visualization. It supports a wide range of data sources and collection methods, as well as its ease of use and extensibility. It can be used in a small-scale development environment or a large-scale production environment because of its great scalability. Grafana also provides a powerful and intuitive solution for monitoring and analyzing data, allowing users to create and customize dashboards and panels.

### 2.2.3 Docker

Docker is a popular platform used for containerization [32]. As described earlier, containerization is the process of creating self-contained, isolated environments that can run applications with all their dependencies. Docker allows developers to package their applications and dependencies into a single unit, called a container. Then, these containers can easily be shared and deployed across different environments.

This technology has become very popular due to its ability to simplify application development, deployment, and scaling. It is frequently used to build and deploy microservice-based architectures [33], as it enables developers to break down complex applications into smaller, more manageable components. By using containers, developers can isolate each microservice and deploy it independently, making it easier to scale and maintain the overall application.

Docker provides a standardized way to package, distribute, and run applications. This helps to eliminate compatibility issues and reduces the time and complexity required to deploy and manage applications. Docker containers are designed to be self-contained and portable. This means that they can run on any infrastructure that supports Docker, for example, on a developer's laptop, a virtual machine, or a cloud-based server.

To help us understand the Docker platform better, there are a few components that we need to mention; the Docker engine, Docker Hub, and Docker Compose. The Docker Engine is the core component that creates, runs, and manages Docker containers. The Docker Hub is a cloud-based registry that allows developers to store, share, and manage Docker images, while Docker Compose is a tool that allows developers to define and run multi-container Docker applications.

A huge benefit of using Docker is that it helps developers achieve consistency in their development, testing, and production environments. Docker containers provide a consistent runtime environment, which means that applications will run in the same way on any machine that supports Docker. This consistency eliminates the need for developers to worry about the compatibility issues that are caused by different operating systems, libraries, software versions, and such.

Another benefit of Docker is its scalability. Docker containers are lightweight, portable, and allow for clustering of multiple Docker containers called swarms. That means that applications may easily be scaled up or down as needed depending on the workload required. This means that applications can be deployed quickly and efficiently, without the need for additional hardware or infrastructure.

A third benefit of Docker is that it also helps improve the security of applications. Containers are isolated between applications and the host system. This reduces the chance of any vulnerabilities in one container affecting other containers or the host system [34]. Furthermore, Docker provides several security features, such as image signing, container access control, and secure image transfer, which can help protect applications from potential security threats.

In summary, Docker is a powerful platform that is used for containerization. Docker is often used to build and create microservice-based architectures because of its ability to break large components into smaller pieces and thus reduce complexity. The three main components of Docker are called Docker Hub, Docker Compose, and the Docker Engine. There are many benefits associated with Docker, including consistency, scalability, and security.

#### **2.2.4 The Go Programming Language**

The Go programming language is often referred to as both Go and Golang [35]. It is a programming language that was created in order to improve older programming languages such as C++ and Java. Golang maintains many useful features of the other older programming languages, such as high performance, readability, static typing, runtime, and usability [36].

One feature of the Go programming language is its concurrency support. The Go programming language was designed in such a way that it could take full advantage of modern hardware. The two essential parts of the concurrency support in Go are called goroutines and channels.

Goroutines can be viewed as lightweight threads for execution of code in Go. The goroutines make it possible to execute concurrent programming by allowing functions to execute code concurrently within the same shared address space. Compared to traditional threads, goroutines are more cheap when it comes to the amount of system memory required to create them. A program can therefore consist of hundreds, if not more goroutines. These goroutines can communicate with each other using something called a channel, which is another essential part of the concurrency support in Go.

A channel in Go is a means by which data can be sent concurrently from one goroutine to another, making it an efficient way of handling data. A channel differs from the traditional array data structure because an array is normally filled with all the necessary data before it is retrieved. Arrays in most languages have custom ways in which you can iterate through them to find a desired data member or a number of data members. Using for loops is also a way of iterating which is efficient and often the usual way of iterating in many languages. Channels, however, are not meant to be used in this manner. As soon as a channel in one routine receives a data member, it is immediately received by the other routine with which it interacts.

Another feature of the Go programming language is the addition of a garbage collection system. It uses a system that manages allocation of deacollection of memory allocations, this is also known as collecting garbage. This garbage collection system runs concurrently with the program, meaning that garbage will be collected while the program is running to make sure it runs as smooth as possible in terms of memory usage.

In the Go language, packages are both a way of organizing code and a way of making them available as open source for other projects. They contain exported functions, which means functions that can be used inside other projects. As mentioned, they can also be used by entirely other Go projects if these packages are open source and are defined as a web URL to its git repository. An example of this is the Go `sqlite3` package; this package's project name is "`github.com/mattn/go-sqlite3`". In order to use this package, one only has to import this URL into their Go file, and one's code is then able to download it during run time.

Another function of Go is that it is possible for a function to return multiple variables. As shown in Code Listing 2.1, an example from our solution, the variables "ipaddress" and "phpsessid" are parameters, meaning they are input variables for the code within the function. The return variables in this example are "chassisType", "serialNumber", and "err".

**Code listing 2.1:** An example of a function in Go.

---

```
1 func GetChassisLabels(ipaddress string, phpsessid string) (chassisType string,  
2 serialNumber string, err error) {}
```

---

The return variables can either be named or unnamed, meaning you can choose to give them names or only use its datatype.

In summary, Go is a programming language that was developed by Google to take advantage of the concurrency capabilities of newer processor architectures. It has two essential parts of its concurrency support, called goroutines and channels. A goroutine is a lightweight thread that is used for execution of code. These

goroutines are able to communicate with each other through channels. As soon as a channel in one routine receives data, that routine may send those data immediately to another routine. It uses a system for managing memory allocation and deallocation, called a garbage collector. Finally, it uses packages, which is a way of organizing code and sharing it with others to use in their projects.

### 2.2.5 The API of the SBCs

The SBCs comes with a REST API that our group was to use when developing the product. In essence, it has the functionality of providing its operators with a wide range of data ranging from call statistics, system resource usage, and data related to networking. These data include both historical statistics as well as real time statistics. The API is issued through HTTP which means that any technology that supports this protocol can request data from the API. The URL of the API determines what data types are received. An example of API usage is seen in Figure 2.8. The figure shows how a process requests system data from the API and receives them as XML data [37]. Further examples of API data are attached to this thesis as Appendix A.

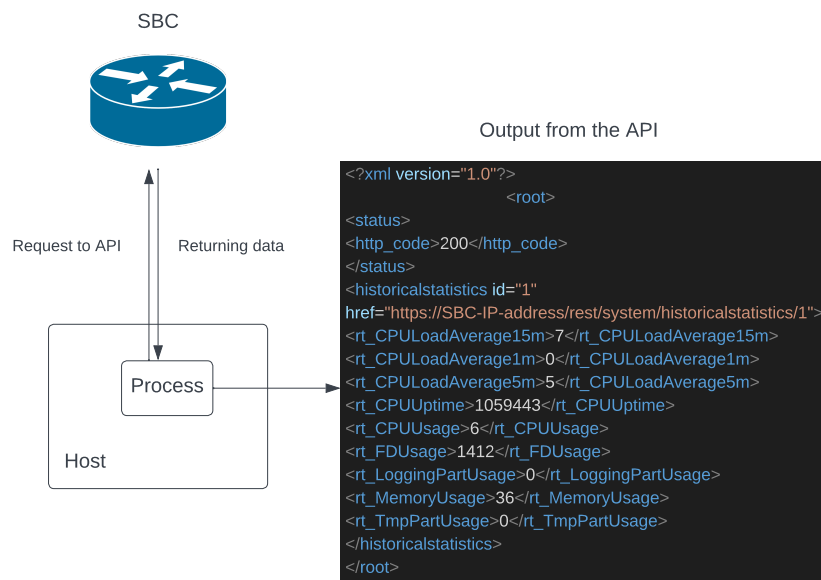


Figure 2.8: The API of the SBCs.





## Chapter 3

# Requirements

The requirements for the Prometheus Exporter is defined according to the employer at *HDO*. This laid the foundation for how the group decided to implement the exporter and which metrics to include in it. There have been small modifications and additional features to the exporter agreed upon by the employer, as all solutions and requirements were not entirely clear from the beginning of this project. The requirements are divided into Non-functional and Functional requirements, the former serving as a description of the overall purpose of the system laid out by *HDO*, and the latter being more technical in nature.

### 3.1 Non-Functional Requirements

*HDO* already had a pre-existing Prometheus Server and a variety of Prometheus Exporters in use that were monitoring most areas of their infrastructure. However, they had not implemented an Prometheus Exporter for monitoring their SBC Edge components which plays a crucial role in handling incoming emergency calls. They wanted this Prometheus Exporter partly because they experienced issues with regard to the quality and stability of many of these calls. These routers consist of a total of 36 units and provide the aforementioned REST API, which provides a vast amount of data necessary to view metrics regarding everything from system workload to phone call quality and more. However, it is nearly impossible to interpret and monitor all these data, as SBCs from Ribbon Communications did not provide a proficient monitoring solution. Therefore, *HDO* needed an additional Prometheus Exporter to collect and provide data to their Prometheus Server, monitoring the SBCs, as well as using their preexisting Prometheus Server system to alert during certain events. The exporter needed to be reliable in such a way that it could potentially run for months before restarting it. It should also be maintainable and portable, utilizing a Docker container that could be quickly shipped and deployed on various hosts. *HDO* also needed the product to be efficient so that the Scrape interval could be set to 15 seconds.

## 3.2 Functional Requirements

The functional requirements are shown as a list of technical details *HDO* wanted implemented in the exporter. Defining these requirements was an ongoing process through communication with *HDO*.

- **Configuration file**

*HDO* wanted a configuration file written in the YAML file format, in order for *HDO* to define the Prometheus Exporter as declarative code. This file should define the attributes of all hosts from which the data were collected. The file should contain a list of hosts, values should contain name, IP address, username, password and a list of Prometheus Collectors that are to be excluded for a host.
- **Included Data Groups** The Prometheus Exporter should consist of groups of metrics that belong together as the same metric affiliation. These data groups are discussed in Section 7.3. The data groups *HDO* wanted to include were the following:
  - **Disk Partition**

This data group contains statistics of the resource usage of each disk partition
  - **Routing entry**

This data group contains information about routing table statistics.
  - **Linecard**

This data group contains information about each line card.
  - **System**

This data group contains information about the utilization of the system resource by the SBC.
  - **System Call**

This data group gives, among other information, information about the success rate of all calls.
  - **Ethernet port**

This data group contains ethernet port statistics which are specific to this technology
- **Labels**

Each metric contains labels defining them in an accurate manner. *HDO* wanted each metric to contain at least the following labels: "hostip" (IP address), "hostname", description of each metric, SBC type and serial number.

- **Repeated and Complex API Call Routines** Some Prometheus Collectors required multiple API calls for collecting data to the same hosts before the final data could be collected. "Routingentry" is an example of such a collector, where it is necessary to fetch the number of routing tables existing for the host, following the number of routing entries for each table, and then retrieve metrics associated with each entry. *HDO* wanted all metrics associated with each routing entry.
- **Docker** The Prometheus Exporter should reside within a Docker container for easier maintainability, shipment, and deployment. The Prometheus Exporter should work so that it can be possible to deploy multiple containers containing multiple hosts as their collect targets.
- **Error Handling** The employer at *HDO* wanted to use a form of error handling called Prometheus up, which is specific to Prometheus.
- **Temporary Storage of Certain Data** Certain data were to be temporarily stored to reuse them. This included PHP Session Cookies, which expire after 10 minutes of authentication to a router with a username and password. It also includes data related to the routing entry collector. These data were to be stored for 24 hours because the frequency with which these data can change may be several weeks. The group was free to choose the storage method.



## Chapter 4

# Methodology

This chapter will discuss the programming language chosen to develop the exporter, the tools that were used to do so, and how development and testing were carried out.

### 4.1 Choice of Programming Language

For the writing of the Prometheus exporter, the group had to decide on a programming language to use for its development. After doing some research and discussion, the group decided to use the Go programming language, also referred to as Golang.

The first reason for choosing the Go programming language is its performance [38]. The product the group has developed handles a large volume of data. As described in Chapter 2, Golang excels in concurrency support with the use of channels and goroutines. This gives it an advantage when working with a lot of data.

Another reason for choosing Golang is because it is a well-documented programming language with many useful resources [39]. One of the packages that Golang supports is the Prometheus package, which is useful for programming applications for Prometheus, such as an Prometheus exporter [40].

A third reason for choosing Golang is that its syntax is relatively simple to learn because of its similarity to other programming languages such as PHP [41]. This also makes it easier for other people who have experience working with similar programming languages to understand the code written in it.

It has a large community of developers who work to develop the programming language and help other users [42]. One may look for help in the various available media, such as GitHub [43], Stack Overflow [44], forums, documentation, etc.

In conclusion, one of the reasons why the group chose the Go programming language for exporter development is because of its performance. Additionally, the group chose it because it is well documented and because it provides many useful resources, such as a helpful community for issues one may face. Lastly, the group also chose it because of its relatively simple syntax similar to other programming languages.

## 4.2 Tools Used for Development

It was decided to use git [45] for its common development features such as version control, the ability to revert back to earlier versions of the code when difficult problems arise, and the ability to create new branches when it is necessary to keep different versions.

After studying a number of open source exporters and online tutorials for creating exporters, it was decided to use CURL [46] to view the output produced by the exporter. This output should be analyzed to see whether correct data had been produced.

## 4.3 Usage of a Development Model

When working with the project plan before development had started, it was decided to use Scrum [47] as our development model. Each iteration (sprint) of planning and development was to last one week. The beginning of each week should have a meeting to discuss what had been done during the previous sprint and what would be developed during the next.

However, this plan was not fully followed because some aspects of development were more time consuming, whilst others were faster to complete. Instead, we had regular meetings with our employer at *HDO* discussing both current status, changes to be made, and what additional features our employer wished to be developed. The time between each meeting with the employer was about two to three weeks.

## 4.4 Development and Testing

There were several difficulties in developing and testing the application due to several factors. The SBC test boxes that *HDO* provided were only available from within their server infrastructure, which meant that they had to be tested on the Ubuntu server [48] virtual machines, which was also provided. Developing on virtual machines was out of the question because Ubuntu servers do not have desktop functionality and applications such as Virtual Studio Code and its utilities. Developing the exporter in this way would have considerably lowered productivity.

Therefore, the code was developed on local computers, which meant that git had to be used in an unusual and impractical manner. Whenever new code was to be tested, we uploaded it from our local computer (git push), and downloaded it from the server (git pull). The first tests were rarely correct, and as you can imagine, the number of git commits was tremendous, eventually exceeding 1000 commits for the whole project. To preserve basic git functionality, such as being able to revert to previous versions, it was decided to include the word "test" for all the commits related to testing.

In order to start developing the exporter, it was found necessary to start off with developing small starting points such as a main method with smaller functions that would give us a proof of concept of certain tasks. The first functions were decided to only validate concepts that revealed whether it was possible to produce a simple output from the REST API. These first functions were tested with different API endpoints before any Prometheus logic was included in the code.

In the beginning of the development process some inspiration was taken from the open source git repositories "Fortigate exporter" [49] and "prometheus-nginx-exporter" [50]. The official Go documentation [39] was used to learn the Go language and all the packages needed. The code was written in Virtual Studio Code due to its features such as displaying errors, listing possible methods for packages, and due to previous experiences using it.

Test functions were used to a great extent during development in order to test the proficiency of components in isolation before integrating them with the overall code. The review of text strings or other data was used extensively to analyze the validity of its output. For example, the development of certain database functions was integrated with code that compares timestamps with the current time. This comparison should determine whether data should be retrieved from the database or whether to retrieve new data over HTTP (discussed in Section 5.4). In these cases, strings were used to reveal whether, in fact, the data had been retrieved over HTTP or from the database.

Testing during development was found to be convenient and more efficient without the need for *HDO*'s Prometheus systems, because metrics could be viewed using CURL [46]. Two SBC hosts were provided for testing that were not in use by *HDO*. The output revealed whether the exporters produced the correct labels and data. For example, labels such as chassis type, serial number and disk names were good indicators that data had been successfully retrieved by the SBCs. During the development of the most difficult collector, the "routingentry" collector, it was necessary to insert text strings into the code to reveal which routing tables and routing entries that had been retrieved from the SBCs. These were then compared with the labels from the metrics output to see whether all of this data had been joined.

This test process worked well in general, except for the aspect of not having *HDO* thoroughly testing our product on their Prometheus systems. This happened at a very late stage of the development process. All tests that were made up to this point were only done internally by the group without access to *HDO*'s Prometheus systems or any other Prometheus servers. This resulted in *HDO*'s tests revealing significant improvement potential (discussed in Section 6.5.5, although they also revealed that the product was proficient and viable in most ways.



# Chapter 5

## Design

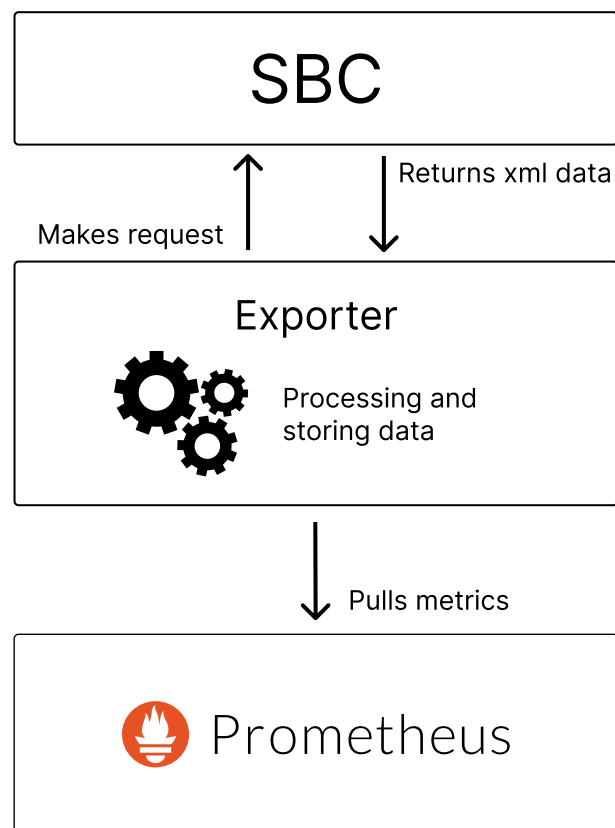
The design chapter describes the overall design of the exporter, including its main components and the inner workings of the architecture.

### 5.1 Overview

The purpose of the exporter is to collect and provide Prometheus metrics from the SBC Edge components that handle emergency calls. To understand the overall design, we must first understand exactly where the exporter fits into the infrastructure, between the hardware it collects data from, to a readable dashboards for employees at *HDO*.

Whenever Prometheus pulls metrics, it triggers the exporter to run its collector routines. It makes Application Programming Interface (API) requests to the SBCs using its already existing REST API as shown in Figure 5.1. The REST API will then return the requested data in a Extensible Markup Language (XML) [37] format. The exporter then processes and stores these received data and provides them to Prometheus. Prometheus can then provide Grafana metrics that can be rendered into visualizations such as graphs and dashboards. This is a continuous process, as Prometheus routinely requests data.

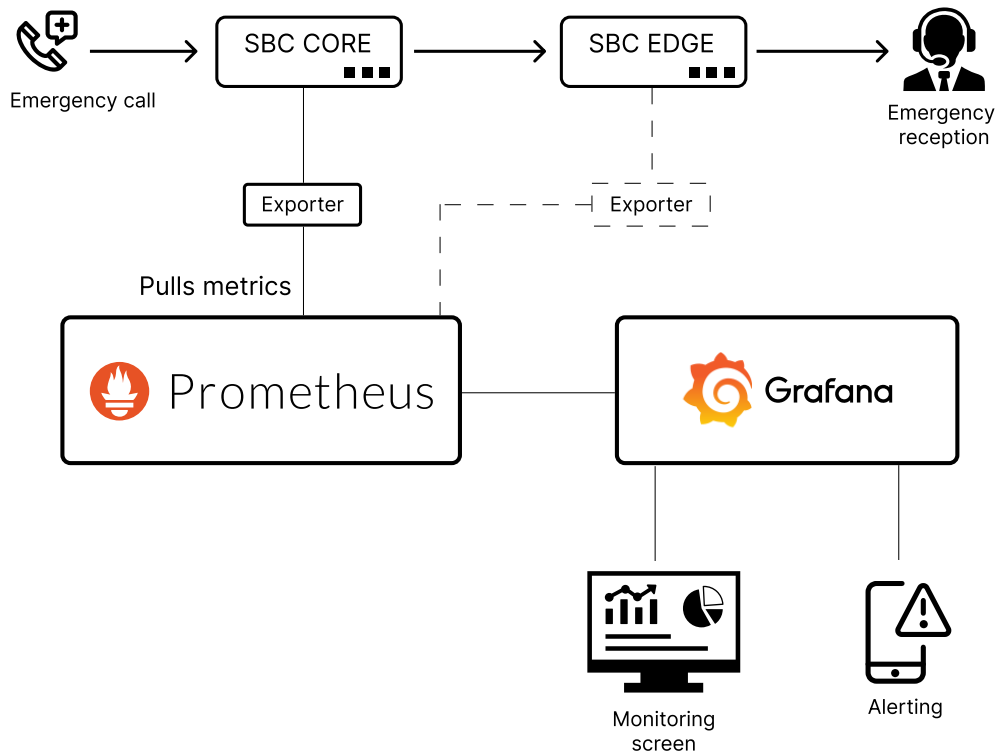
We can divide the exporter's job into two main tasks. First, it needs to request and receive data using the REST API. The exporter will do so by running its routines to collect data. Furthermore, it needs to transform metrics from retrieved data into a format that can be ingested by Prometheus, including categorizing these metrics with accurate labels to make the data readable and useful for employees at *HDO*.



**Figure 5.1:** The main tasks of the exporter simplified.

## 5.2 Where the Exporter Fits Inside HDOs Infrastructure

Figure 5.2 illustrates *HDO*'s infrastructure for receiving emergency calls. SBC CORE is the main network used for handling emergency calls and uses Internet Protocol (IP)/Session Initiation Protocol (SIP) for communication, the SBC EDGE converts from IP/SIP to Integrated Services Digital Network (ISDN), which is what emergency receptions use for communication. *HDO* also has preexisting Prometheus systems that they already use to monitor the SBC CORE network as well as other parts of their infrastructure. Their Prometheus systems work in interaction with several exporters that are responsible for collecting its own set of metrics. The exporter discussed in this thesis is responsible for collecting data from the SBC EDGE routers as an addition to their preexisting Prometheus systems. On the Prometheus server's side, the configuration will be set to determine how often a Scrape is executed. The exporter in question only starts collecting data on each scrape request it receives from the server and returns the finalized metrics to the Prometheus server.



**Figure 5.2:** This figure shows the flow of data and the exporters function within *HDO*'s infrastructure. The dotted lines illustrates where the developed exporter fits inside of *HDO*'s infrastructure.

### 5.2.1 Data Groupings and Collectors

*HDO* wanted an option to exclude certain data groups. Therefore, it was natural to design the exporter in such a way that data types belonging together inside the API, also were defined together in each Prometheus Collector of the exporter. This project refers to a Prometheus Collector as the code that collects metrics from a specific data group on a host and defines them as Prometheus metrics. This way of grouping metrics within the exporter is convenient because it allows *HDO* to exclude their own defined data affiliations for specific SBC hosts, if needed. These metric groups are also defined by the SBC documentation [51]. For example, one data group contains data concerning system call statistics, having metrics such as 'rt\_NumCallAttempts', 'rt\_NumCallSucceeded' and 'rt\_NumCallFailed'. This particular group of data is collected using HTTP requests to the API's URL; 'https://ipaddress/rest/systemcallstats'.

### 5.2.2 Docker

The original concept was to use a small number of Docker containers containing multiple SBC targets in its configuration. However, as issues arose concerning performance in speed for scenarios where more than two hosts were configured, the option of splitting the deployment of the solution into a greater number of containers, having only one or two SBC hosts was later considered. This possible scenario did not affect the development of the exporter in any way, because one container does not affect another.

### 5.2.3 Temporary Storage of Certain Data

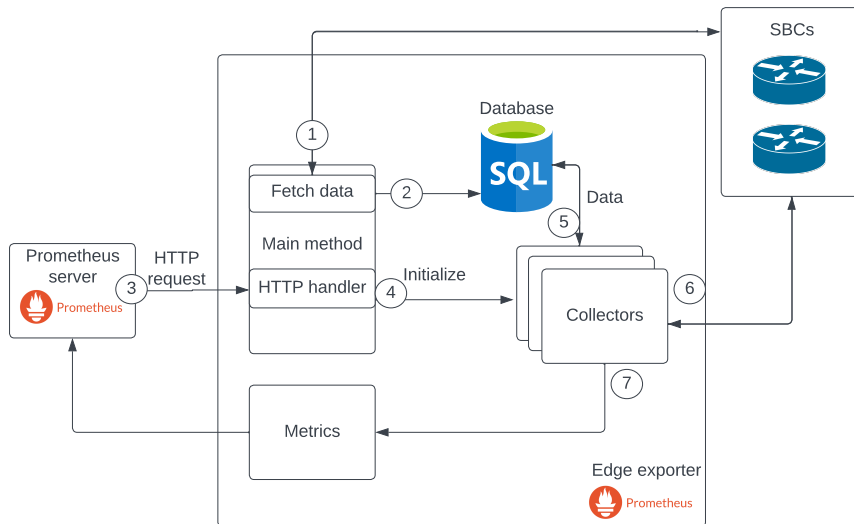
During the development of the exporter, issues arose about the efficiency of each scrape. Many of the performance issues were associated with the large number of REST API calls that were being made to each SBCs. To eliminate some of the traffic to each SBC, it was decided to use Sqlite3 [52] to store the required authentication identifier, which is a PHP Session Cookie for each host.

More technically, to get data from a host, you would have to send a HTTP POST request containing a username and password, and as a result retrieve a session cookie to retrieve the intended data. As this session only lasted for 10 minutes, it was decided to store each session cookie for 9 minutes within the database so that it could be used multiple times instead of requesting a new cookie for each API call. This resulted in a scrape being shortened from 13 seconds to 10 seconds when using six collectors and two hosts.

To further decrease the time spent for each scrape, other data that had to be fetched were also saved, this will be discussed in our database implementation. The implementation of the database is discussed in Section 6.4.3.

### 5.3 Design of the Exporter

Each process that occurs in the exporter shown in Figure 5.3, will be discussed in this section. The numbers in parentheses refer to the numbers in the figure. The figure shows three main components, which are the exporter, a Prometheus server 2.2.1 that requests data, and the SBCs they are collecting data from.



**Figure 5.3:** Design of the exporter.

- A PHP Session Cookie is required to fetch data from the API of the SBCs as described in Section 2.2.5. To obtain one, one would need to make a HTTP POST request with a username and password attached. Afterward, a PHP session cookie can be retrieved from the response header. When starting the exporter, the main method gathers all SBC configuration and fetches PHP Session Cookies and chassis information from the SBC hosts (1), which is then inserted into the database (2).
- The exporter is then in the mode where no operation is executed, it is only waiting for incoming HTTP-requests from any host. Each time the HTTP handler receives a scrape (3), it initializes the Collectors to start collecting data (4). A HTTP handler is a process that responds and connects to a HTTP request. The response in this case is to initialize the collectors and return their metrics on completion.

- After initialization, each collector then queries the database for reusable data, such as PHP Session Cookie and chassis information, that it needs to collect data from the SBCs (5). Section 5.3.1 discusses when data from the database are being used.
- The collectors start collecting data from the SBCs (6) and sometimes updates the database with certain data.
- The collectors then convert the data into metrics and sends it to the Prometheus server (7) via the HTTP handler. Section 5.3.1 explains in more detail what processes occur in each collector, and Section 6.5 explains the implementation of its code.

### 5.3.1 Collector Design

Figure 5.4 gives an overview of the processes that occur for each of the collectors implemented. Numbers are used in parentheses to refer to each process in the figure. Chapter 6 discusses each step in the collector more thoroughly and how each of the components is implemented. Most components of the collectors are implemented as packages, which describes how code has been structured. We will go through the steps of each event in the collector in the same order as provided. The numbers in parentheses are references to the processes shown in Figure 5.4.

- (1) The collector starts by fetching the configuration of each SBC target defined in the `config.yml` file.
- Each collector looks for PHP Session Cookies in the database and compares its timestamp with the current time and uses these session cookies to fetch data from the SBCs if the timestamps have not expired (2).
- Whenever a session cookie has expired, the HTTP-package is used to authenticate to the SBCs, fetching new PHP Session Cookies (3). (4) The database is then updated with recently recovered PHP Session Cookies.

In the case of three of the collectors, other reusable data are also queried and stored in the database, see Section 6.4.3. Some of these data expire after a duration, which the user can configure in the product's configuration file.

- (5) The API data containing its original XML format are filtered and converted to variables that will be used as metrics.
- (6) These variables are then formatted as Prometheus metrics or as their labels. Labels such as "hostip", "hostname", and additional labels are then added. In Chapter 6, further details of how these collectors register each

metric will be discussed.

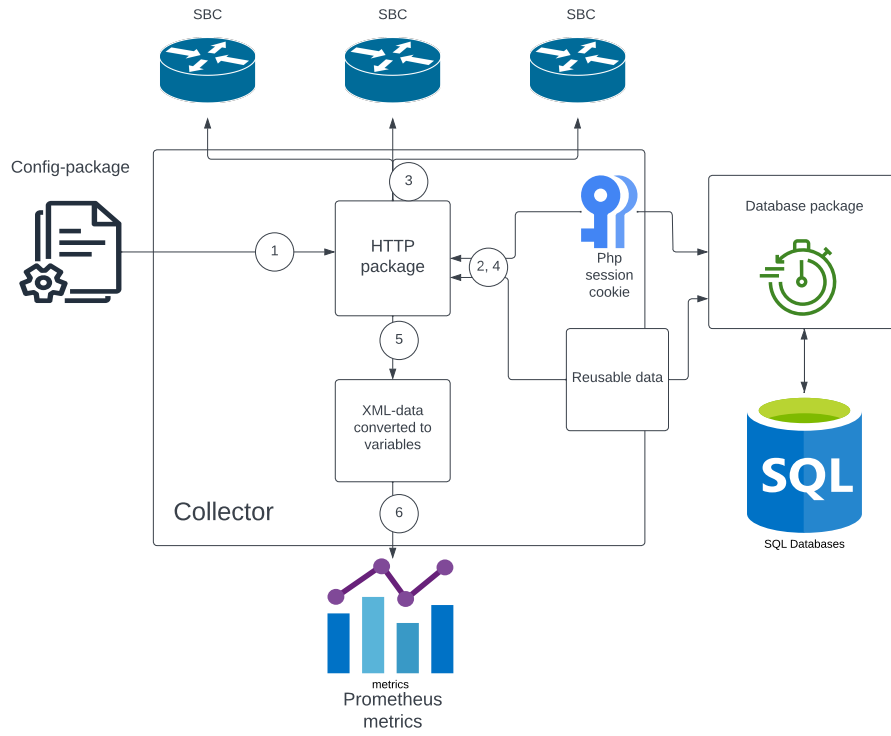


Figure 5.4: Each collector consist of these processes.





## Chapter 6

# Implementation

This chapter will discuss the code for the implementation, the choice of features, and the details of the architecture that has been developed. Additionally, we will discuss the decision to change the implementation to an overall improved and more scalable version.

### 6.1 Terms and Standards

This section describes some standards and terms used to discuss the implementation of code.

Firstly, when discussing a function that is not listed as a code listing, only the function name is included with parenthesis at the end, indicating that it is a function. An example of such a function is `GetChassisLabels()`. Many of these functions are not listed for various reasons. The main reason being that they are database related functions with functionality that are similar across data types, such as inserting and retrieving data, checking if tables exist and so forth.

Sometimes, all the code contained within a function is not included. The reason behind this is that it would be unproductive to discuss every detail of our code, as sometimes the purpose is to see the overall picture. In these cases, we instead use the same standards as in the Go documentation [39] which is to include the name of the function and its parameters and return values.

#### 6.1.1 Terms

This subsection will cover some of the terms used when discussing code in this chapter.

### Session Cookies

In order to fetch data from the API of the SBCs as described in Section 2.2.5, one would need to perform a HTTP POST request with a username and password attached. Afterwards, a PHP session cookie would be received from the response header which is required to collect data from the API.

### Collector

In our implementation, a Prometheus Collector contains the code that collects metrics from a specific data grouping on each host and defines them as Prometheus metrics.

### HTTP Handler

A HTTP handler [53] is a process that responds to a HTTP request. In this implementation, the response is to initialize the collectors and return metrics upon completion.

### Interface

The Go programming language provides something called interfaces [54]. An interface is a set of methods that can be applied to data, without knowing the underlying data structure. The Prometheus Go package described in Section 6.3 provides mandatory interfaces.

## 6.2 Usage of Pointers

For the code developed for the product, pointers were not implemented in all cases where it was possible. The reason behind this is due to time constraints [55]. The main drawback of omitting the usage of pointers in Go is the efficiency of running the code, because the data retrieved multiple times are then copied to new locations in memory during run time [56].

### 6.2.1 Error Handling

For error handling, the plan was to use a type of error handling specific to Prometheus called Prometheus up [57]. However, this was difficult to implement in integration with the entire system that had been developed up to that point. The reason behind this was because this type of error handling (Prometheus up) resulted in all other metrics being removed from the Scrape result as well, although the error may have affected only one host or Prometheus Collector.

One solution could have been to serve metrics from separate URLs for each Prometheus Collector and host. However, it was decided not to, as it would be

more convenient for all parties to serve all metrics on the same URL. Keeping this implementation meant that all metrics would have the same timestamps in Prometheus, as well as easier usage of the Prometheus Exporter.

Instead, an alternative solution was devised, to instead return a gauge metric to the Prometheus Server revealing the cause of the error and its affected hosts and Prometheus Collectors. The employers agreed to the solution because they already have similar approaches for other exporters in use. However, after some back and forth, they wished for a slightly different solution. Instead, every scrape should have a gauge metric that only reveals whether the Scrape was successful or not, with no explanation for its cause. This is because in Prometheus, a metric having different labels would also be displayed as different metrics. These values would be 1 or 0 as shown in the sample below.

```
scrape_status{hostip="10.233.230.11",hostname="host2"} 1
scrape_status{hostip="10.233.234.11",hostname="host1"} 0
```

## 6.3 The Go Prometheus Package

The product utilizes the Go Prometheus package [40] to a great extent. This package contains useful resources for developing Prometheus programs. In order to use the Go Prometheus package, the two interfaces that are going to be discussed; Collect and Describe, are mandatory. Failure to include these interfaces causes the package to give an error. In these interfaces, channels are used to insert data defined by the developer and to arrange it in such a way that they can be converted to Prometheus metrics by the Prometheus Go package behind the scenes.

### 6.3.1 The Describe Interface

The describe interface shown in Code Listing 6.1 takes predefined metric descriptions and sends them through the channel used by the Prometheus module. The channel is meant to receive definitions of metrics, which means variables such as the name of the metric, the description, and a list of labels for each metric.

**Code listing 6.1:** The describe interface in the Prometheus Go package.

---

```
1 func (collector *Metrics) Describe(c chan<- prometheus.Metric)
```

---

### 6.3.2 The Collect Interface

The collect interface shown in Code Listing 6.2 implements the collector. This means that it contains code responsible for both collecting data and for giving the metrics values and attributes that are sent through the channel.

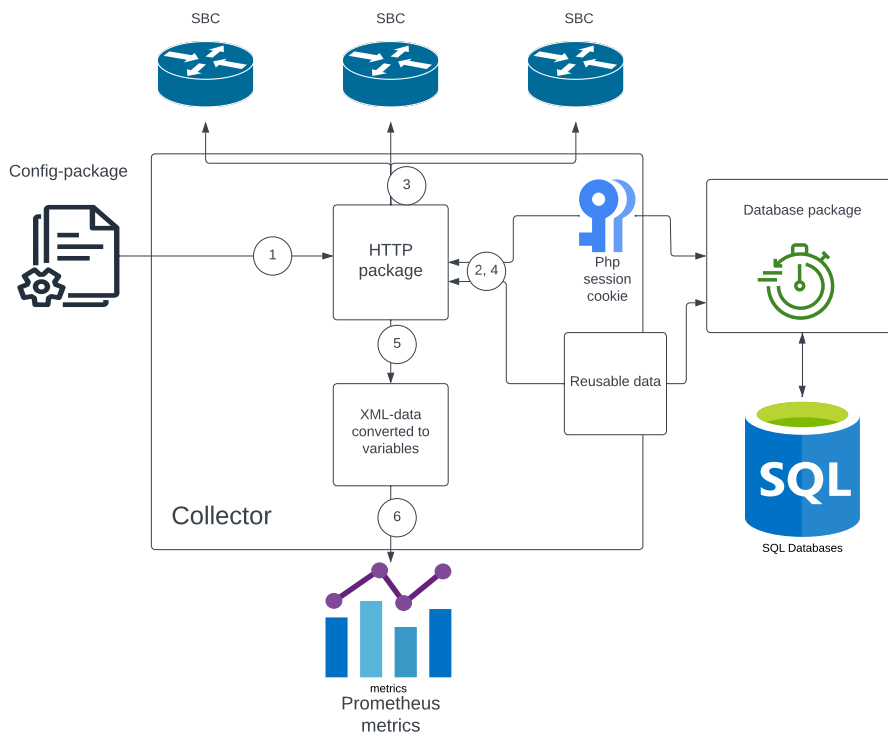
**Code listing 6.2:** The collect interface in the Prometheus Go package.

```
1 func (collector *Metrics) Collect(ch chan<- *prometheus.Desc)
```

## 6.4 Implementation of Code

The implemented code has been organized in such a way that code belonging together in each implemented package is listed as such in this chapter. Packages are used to organize the code. These packages include the main package, which contains only the main function, the packages collector, database, HTTP, config and utils.

Figure 6.1 again shows the architecture inside each collector in which most of these packages interact. However, the figure does not contain all the components of the exporter.



**Figure 6.1:** Each collector consist of these processes.

### 6.4.1 The HTTP Package

This section will explain in more detail how the HTTP package interacts with the database package to decide whether to use data from the database or to fetch new data from the SBCs. For this project, it was decided to use Golang's official net package [58] and HTTP package both for authentication and data collection. The development of this package also utilized the tool Curl-to-Go [59] which provided a good starting point for the functions that were created. However, changes and additional code were necessary to make it work. The curl commands that were being translated were found in the Ribbon Communications documentation [51].

#### Authentication to the SBCs

The beginning of the function `APISessionAuth()` shown in Code Listing 6.3 calls another function which attempts to fetch a PHP Session Cookie from the database entry containing the provided IP address. This session cookie can be used for up to 10 minutes before it expires.

Therefore, `APISessionAuth()` (Code Listing 6.3) uses the `GetSqliteKeyIfNotExpired()` function shown in Code Listing 6.4 to attempt to fetch the previous session cookie from the database. This function is responsible for comparing the string field "time" with the current time, using Golang's time package [60].

**Code listing 6.3:** The function for connecting to an SBC.

---

```
1 func APISessionAuth(username string, password string, ipaddress string) string
```

---

**Code listing 6.4:** The function for retrieving a session cookie before expiration.

---

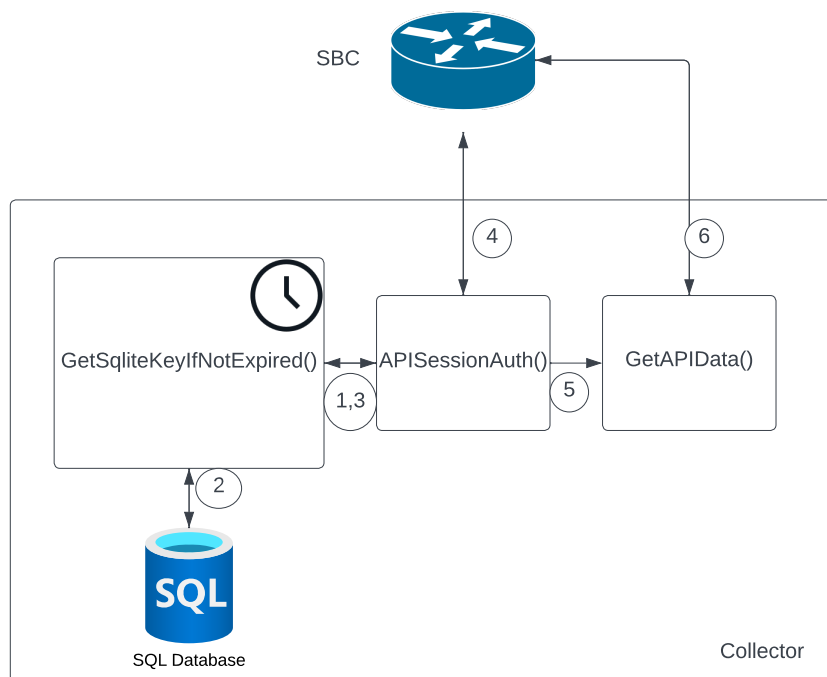
```
1 func GetSqliteKeyIfNotExpired(ipaddress string) (cookie string, err error) {
```

---

In Figure 6.2 you can see how the HTTP package interacts with the database, which includes processes that involve authentication and fetching data regarding the current collector.

- **1.** `APISessionAuth()` (Code Listing 6.3) first tries to fetch a PHP Session Cookie from the database
- **2 and 3.** If the database have an entry with the given session cookie and it has not expired, then `APISessionAuth()` returns this cookie.
- **4.** If either of the above statements are not valid, the `APISessionAuth()` function authenticates to the SBC, fetching a new session cookie and inserting it into the database.
- **5 and 6.** The `GetAPIData()` function (Code Listing 6.5) uses this session cookie to fetch any data needed by the current collector.

In Section 6.5, it is discussed how each collector converts these data into metrics.



**Figure 6.2:** How each collector collects new data.

## Requesting Data from SBCs

The `getAPIData()` function shown in Code Listing 6.5 uses Golang's official cookie management implementation in the `net` package, in order to request API data. A cookie is added to the HTTP request using the `httprequest.AddCookie()` method. As the response data arrive in a `bytestream`, we can use the package `ioutil` [61] to read the data within the response body with the function `ioutil.ReadAll()`.

The function finally returns a `bytestream` containing the response body. All data fetched with this function, a function that every collector uses, are in the XML file format. The argument "url" defines the API endpoint, meaning what data to collect. However, the collectors contain the code responsible for parsing this data into variables readable by Prometheus. As all API data resides in the XML format, the implementation uses the Golang XML package [62] and its method `xml.Unmarshal()` with appropriately written structs to convert the XML data into variables within the collector. An example of such a struct is shown in Code Listing 6.6.

**Code listing 6.5:** The function for retrieving API data.

---

```
1 func getAPIData(url string, phpsessid string) ([]byte, error)
```

---

**Code listing 6.6:** Example of a struct used by the collectors.

---

```
1 type lSBCdata struct {
2     XMLname    xml.Name    'xml:"root"'
3     LinecardData LinecardData 'xml:"linecard"'
4 }
5 type LinecardData struct {
6     Href        string    'xml:"href,attr"'
7     Rt_CardType int    'xml:"rt_CardType"'
8     Rt_Location int    'xml:"rt_Location"'
9     Rt_ServiceStatus int    'xml:"rt_ServiceStatus"'
10    Rt_Status   int    'xml:"rt_Status"'
11 }
```

---

## 6.4.2 Config Package

The first struct in Code Listing 6.7 defines the YAML config file, which is the exporter configuration that was discussed in chapter 5. The struct "IncludedHosts" is meant to be returned to the collector that is using the configuration, for example, every collector.

**Code listing 6.7:** Describing the configuration file.

---

```
1 type Config struct {
2     Hosts []Host
3     Authtimeout int    'yaml:"authtimeout"'
4 }
```

---

```

5 }
6 type Host struct {
7     HostName    string 'yaml:"hostname"'
8     Ipaddress   string 'yaml:"ipaddress"'
9     Username    string 'yaml:"username"'
10    Password    string 'yaml:"password"'
11    Exclude     []string 'yaml:"exclude"'
12    RoutingEntryTime float64 'yaml:"routing-database-hours"'
13 }
14
15 type IncludedHosts struct {
16     Ip          string
17     Hostname    string
18     Username    string
19     Password    string
20     RoutingEntryTime float64
21 }

```

The function `getConfig()` shown in Code Listing 6.8 retrieves a pointer to the configuration by reading the configuration file using the packages `ioutil` [61] and `YAML` [63]. It is a function that was found on Stack Overflow [64].

---

**Code listing 6.8:** Reading the configuration file.

---

```

1 func GetConf(c *Config) *Config

```

The `GetIncludedHosts()` function shown in Code Listing 6.9 is used by each collector to get the configuration of all the hosts included for this collector. The collector is identified with the collector names written in the `config.yaml` file. These names have to match exactly, meaning that the collector names in the configuration can not be misspelled. The function iterates through all hosts in the saved configuration, and if they do not have the specified collector excluded, it appends and returns a list of these hosts' configuration.

---

**Code listing 6.9:** Returning SBC hosts' configuration used by the collectors.

---

```

1 func GetIncludedHosts(collectorName string) []IncludedHosts {
2     cfg := GetConf(&Config{})
3     list := make([]IncludedHosts,0,8)
4     var excluded bool
5
6     for i := range cfg.Hosts {
7         for v := range cfg.Hosts[i].Exclude {
8             if (cfg.Hosts[i].Exclude[v] == collectorName) {
9                 excluded = true
10            }
11        }
12        if !excluded {
13            list = append(list, IncludedHosts{
14                cfg.Hosts[i].Ipaddress,
15                cfg.Hosts[i].HostName,
16                cfg.Hosts[i].Username,
17                cfg.Hosts[i].Password,
18                cfg.Hosts[i].RoutingEntryTime})
19        }

```



```

20     }
21     return list
22 }

```

---

### 6.4.3 Database Package

The exporter uses a database to solve performance problems related to the extensive, but required, use of HTTP calls to each SBC. In total, the usage of a database spared about 3-4 seconds on a scrape for all of the collectors with two hosts included. The chosen database is included in the `sqlite3` [65] package for Golang [66]. An alternative to `sqlite3` would be to write the data to a file format such as YAML, but the group chose `sqlite3` due to prior experience with Structured Query Language (SQL) databases, but also because it is scalable and known to be efficient and secure [67].

#### Stored Data

The database is used extensively to store data collected with one of the collectors called the routing entry collector, session cookies, and chassis information.

#### Session Cookie Data

As mentioned when discussing "Authentication to the SBCs" in Section 6.4.1, the session cookies that are retrieved after connecting to each SBC can be reused for 10 minutes. Therefore, it was decided to reuse them for this period of time. The collectors were already implemented by including iteration over each SBC host, and performing API calls against each of them. Therefore, it was decided to integrate the database functions inside these "for loops". This meant that the functions `GetCookieDB()` in Code Listing 6.10 and `InsertAuth()` in Code Listing 6.11 took the host's IP address within the current loop as parameters when either inserting or retrieving data from the database. Other functions associated with authentication include `Update()`, `RowExists()`, and `CreateAuthTable()`.

---

**Code listing 6.10:** Retrieving session cookie from database.

---

```

1 func GetCookieDB(db *sql.DB, ipaddress string) ([]*Cookie, error)

```

---



---

**Code listing 6.11:** Inserting session cookies into the database.

---

```

1 func InsertAuth(db *sql.DB, ipaddress string, phpsessid string, time string) error

```

---

#### Routing Entry Data

Routing entry data contains data gathered from the first two API calls made to a host. It was decided to store routing tables, time, and a map of the routing table in

an array of routing entries. This is because each routing table on the host contains a number of routing entries, and the use of a map was necessary to find the correct entries inside the routing entry collector. However, as was later discovered, it was impossible to store arrays within the sqlite3 database. Therefore, it was decided to arrange the database table so that each database entry was able to contain the same routing tables while having different routing entries, arranged over multiple database entries.

An example of the SQLite table structure for routing entries is shown in Table 6.1. As you can see, one routing table can contain many different routing entries.

**Table 6.1:** The Structure of the Table Storing Routing Data

IP	table nr	entry nr
ipaddress	2	1
ipaddress	2	2
ipaddress	2	3
ipaddress	5	1
ipaddress	5	2
ipaddress	5	3
ipaddress	5	4

As the collector already had loops iterating over each routing table, and thereafter loops for each routing entry, storing these data was done with the function `StoreRoutingEntries()` shown in Code Listing 6.12. The function takes the IP address, routing tables, and routing entries as parameters. As routing entries are given as an array, the function iterates over each routing entry and writes rows to the database, containing the given parameters.

**Code listing 6.12:** Storing routintables and entries into the database.

---

```

1 func StoreRoutingEntries(db *sql.DB, ipaddress string, time string,
2 routingTable string, routingEntries []string) error

```

---

When fetching routing entry data from the database, storing this data structure as a map was the most convenient way of integrating a database to the complicated problems faced inside the "routingentry" collector. The map consists of one table as the key and an array of routing entries as values. The function in Code Listing 6.13 fetches routing tables and entries from the database and stores them as a map, the last of which occurs on line 28. On line 33, the function extracts the keys of the map which are the routing tables. The function returns a map of routing tables and entries, an array of routing tables, its timestamp in string, and a potential error.

The database functions associated with routing data also include the functions `CreateRoutingSqlite()`, `DeleteRoutingTables()` and `RoutingTablesExists()`.

**Code listing 6.13:** Retrieving routing data from the database.

---

```

1 type RoutingT struct {
2     Id      int
3     Ippaddress string
4     Time    string
5     RoutingTable string
6     RoutingEntry string
7 }
8
9 func GetRoutingData(db *sql.DB,ipaddress string) (map[string][]string,[]string,
10 string, error) {
11     row, err := db.Query("SELECT * FROM routingtables")
12     if err != nil {
13         log.Print(err)
14         return nil, nil, "", err
15     }
16     defer row.Close()
17
18     var time string
19     var routingEntries = make(map[string][]string)
20     var tables []string
21
22     for row.Next() {
23         r := &RoutingT{}
24         err := row.Scan(&r.Id, &r.Ippaddress,&r.Time,&r.RoutingTable, &r.
25             RoutingEntry)
26         if err != nil{
27             log.Print(err)
28         }
29         if (r.Ippaddress == ipaddress) {
30             routingEntries[r.RoutingTable] = append(routingEntries[r.RoutingTable],
31                 r.RoutingEntry)
32             time = r.Time
33         }
34     }
35
36     for key, _ := range routingEntries {
37         tables = append(tables, key)
38     }
39     return routingEntries,tables,time,err
40 }

```

---

### Chassis Data

The function responsible for fetching chassis data starts with an API call to each of the SBCs, fetching its SBC type and serial number. The chassis type is either SBC1000 or SBC2000, *HDO* wanted these as labels for each metric related to the system resource collector. This is also stored in the database as long as the exporter is running, as this data does not change over time.

## DButils

DButils is the name of a file that contains the `initializeDB()` function, which deletes any previous databases in the root directory and creates a new database and all required tables. This function is executed inside the main function in order to alleviate some of the workload on the first scrape, making it less time consuming. It was decided to delete any previous instance of the database during run time as the main program would only be restarted usually around once a month, meaning the risk of potential errors with having previous data was greater than the performance benefits of keeping them.

### 6.4.4 Utils Package

The utils package contains the functions `GetChassisLabels()` and `Expired()`.

#### Chassis information

This package contains a function `GetChassiLabels()` shown in 6.14 to fetch chassis information over HTTP, and calls functions from the database package that insert and retrieve these data from the database.

The steps of this function are similar to both the collectors and the functions that utilize the database; it checks if any data containing a specified IP address already exist in the database. If it does, then retrieve these data; if not, then fetch the new chassis information from the SBC, and insert it into the database. This information is added as labels to some of the metrics. It includes SBC type, SBC1000 or SBC2000, and serialnumber.

---

**Code listing 6.14:** Fetching chassis information from either an SBC or database.

---

```
1 func GetChassisLabels(ipaddress string, phpsessid string)
2 (chassisType string, serialNumber string, err error)
```

---

#### Comparing Timestamps From the Database

The function `Expired()` in Code Listing 6.15 is used by the "Routing entry" collector to compare the timestamp of its data in the database to the current time. The code that is responsible for fetching PHP Session Cookie from the database also has a similar approach. It returns true if the data in the database have expired. In the "Routing entry" collector, this function's parameter "hours" is provided by the time schedule specified in the configuration file. The parameter "previoustime" is provided by the database because it is a timestamp when the previous data was stored.

Figure 6.3 displays how the `Expired()` function works. Whenever the previous time with the added time schedule (T3) is after the current time (T2), it returns

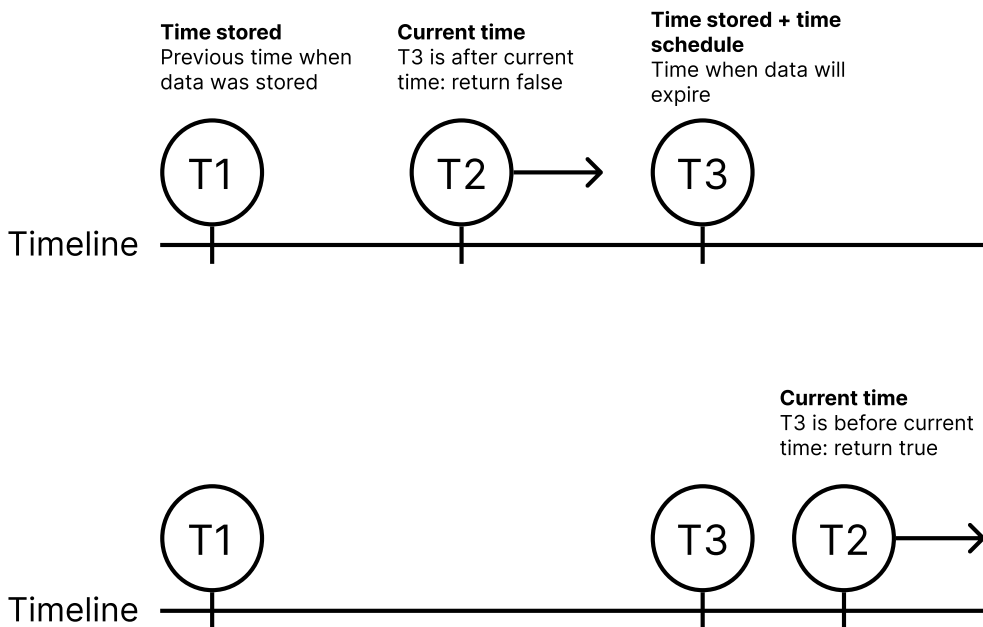
false, which means that it has not expired yet. As soon as the current time crosses T3, it will return true, and therefore the previous data will have expired.

**Code listing 6.15:** The function Expired().

```

1
2 func Expired(hours float64, previoustime time.Time) bool{
3     //The code here is excluded, for the purpose of simplicity
4
5     return previoustime.Add(duration).Before(timeNowParsed)
6 }

```



**Figure 6.3:** Shows when data expires according to a given time schedule.

### 6.4.5 Main Package

The main package consists only of one file that contains only the main function as shown in Code Listing 6.16. The main function starts off with calling the InitializeDB() function which creates a database and new tables. It also executes the APISessionAuth() function as shown in Code Listing 6.3 and GetChassisLabels() as shown in Code Listing 6.14 on each host, and inserts their returned data into the database.

Finally, the main function continuously waits for HTTP requests (scrape). The exporter starts collecting data whenever a request is sent to the exporter, which is initiated by our HTTP handler called "Probehandler" as shown in Code Listing 6.17. The end of the main function indicates that metrics are to be collected from

"http://host-IP:5123/metrics".

**Code listing 6.16:** The main function.

---

```

1
2 func main() {
3     //Creating database and tables
4     database.InitializedDB()
5
6     hosts := config.GetAllHosts()
7     for i := range hosts {
8         //Fetching sessioncookies and inserting them into the database
9         phpsessid, err := thishttp.APISessionAuth(hosts[i].Username, hosts[i].Password
10            , hosts[i].Ip)
11         if err!= nil {
12             log.Print(err)
13             continue
14         }
15         //Fetching SBC type and serialnumbers, and placing them in database
16         _, _, err = utils.GetChassisLabels(hosts[i].Ip, phpsessid)
17         if err!= nil {
18             log.Print(err)
19             continue
20         }
21     }
22     http.HandleFunc("/metrics", collector.ProbeHandler)
23
24     log.Fatal(http.ListenAndServe(":5123", nil))
25
26     log.Println("Edge exporter running, listening on 5123")
27     select {}
28 }

```

---

## 6.5 Collector Package

The Collector package is mainly responsible for collecting data from the SBCs via their API and converting the data into Prometheus metrics.

### 6.5.1 HTTP Handler and Probe Interface

The exporter uses a custom HTTP handler as shown in Code Listing 6.17 that is responsible for initializing the Probe interface whenever a HTTP request is made to the exporter. The Prometheus package registers Probe() as a collector and returns its final data on completion.

**Code listing 6.17:** The function ProbeHandler().

---

```

1 func ProbeHandler(w http.ResponseWriter, r *http.Request) {
2     registry := prometheus.NewRegistry()
3     pc := &AllCollectors{}
4     registry.MustRegister(pc)
5     pc.Probe()
6 }

```

```

7   h := promhttp.HandlerFor(registry, promhttp.HandlerOpts{})
8   h.ServeHTTP(w, r)
9 }

```

The current implementation is based on an approach found in the open source git repository "Fortigate exporter" [49]. In this implementation, all collectors share one Collect interface, which reads metrics from data previously assigned to the struct "AllCollectors", by the method Probe 6.18. Each collector adds metrics to this struct. The required Describe interface is empty, because metrics descriptions are now instead defined directly in each collector.

**Code listing 6.18:** Interface Probe and Prometheus interfaces.

```

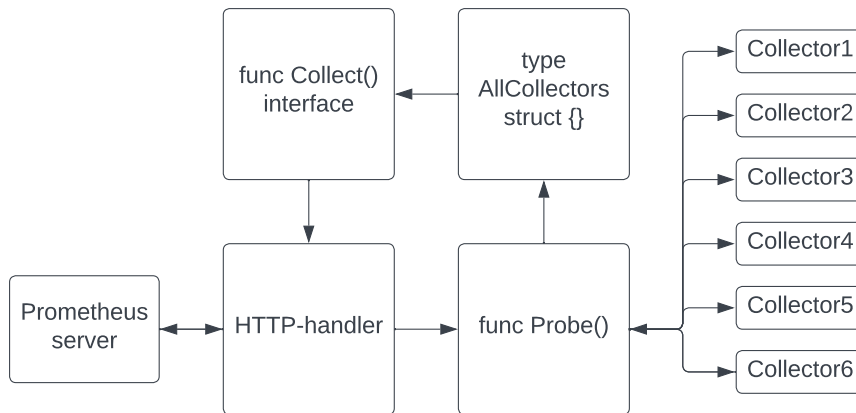
1
2 type AllCollectors struct{
3     metrics []prometheus.Metric
4 }
5
6 func (m *AllCollectors) Probe() {
7     metrics := SystemCollector()
8     for i := range metrics {
9         m.metrics= append(m.metrics, metrics[i])
10    }
11    metrics = LinecardCollector2()
12    for i := range metrics {
13        m.metrics= append(m.metrics, metrics[i])
14    }
15    ...
16    ...
17 }
18 func (collector *AllCollectors) Collect(c chan<- prometheus.Metric) {
19     for _, m := range collector.metrics {
20         c <- m
21     }
22 }
23 func (collector *AllCollectors) Describe(ch chan<- *prometheus.Desc) {
24 }

```

As shown in Figure 6.4, each HTTP request to the exporter is handled by our HTTP-handler which initializes each collector to start collecting metrics. These metrics are appended to the "AllCollectors" struct sequentially. When the Probe() function completes, the Collect interface uses this struct to send its metrics through its channel handled by the Prometheus package behind the scenes. The Prometheus package returns formatted Prometheus metrics back to the Prometheus server via the HTTP handler.

## 6.5.2 Collectors

The code shown in Code Listing 6.19 is one of the collectors, the Linecard collector, in the collector package. As you can see, it is quite large. This collector was chosen as an example as it is one of the simpler ones to follow compared to the rest of the collectors.



**Figure 6.4:** Shows the initialization of collectors and their flow of data.

The Linecard collector begins by describing the structure of the output gathered, which is in the XML file format. The struct "LinecardData" from lines 12 to 18 describes the metrics to be read from the output.

The following is a walk through of the process of what is happening behind each step within the collector called the Linecard collector as shown in Code Listing 6.19.

- The collector starts by gathering included hosts (targets) and their configurations for the current collector (line 21).
- It proceeds by defining variables containing Prometheus metric descriptions (from line 26), which will later be used when defining each metric with values and labels.
- It then begins a loop through each host as we want to execute API calls on them (line 37), fetching data from these SBC hosts.
- The loop then starts by fetching a PHP session cookie (line 39) using the previously discussed `APISessionAuth()` function in Code Listing 6.3. In most cases, the session cookie already resides inside the database, but if its timestamp has expired, it fetches a new session cookie.
- Then, a similar process applies for the function `utils.GetChassisLabels()` (Code Listing 6.14), which fetches the label "chassis type" (line 46). As this function is initialized in the main function the same way as `APISessionAuth()`, these labels should already be residing inside the database. It will attempt



to fetch new labels if the database is empty.

- For this particular collector, the targets are fixed in terms of API endpoints, meaning that the collector only has to execute an API call on IDs that are the same for every SBCs, if they have the same SBC type.

As you can see from lines 71 to 74 in Code Listing 6.19, SBC1000 always has the linecard IDs 7 and 8, while SBC2000 has the IDs 1 and 2. Therefore, it uses "if tests" to determine which IDs to use. It was also decided to include "else statements" in case the linecard IDs are not correct, meaning that there has been an error over several steps.

- Now, the collector starts a loop over each linecard ID (line 62) and performs an API call on each of them using `GetAPIData()` (line 64). As the data are returned as a bytestream from this function, one may then use the official XML package alongside the previously built structs to convert the data into variables (lines 70-79).
- Finally, the collector creates metrics using the Prometheus package's `MustNewConstMetric()` function (lines 81-82), specifying each individual metrics description, type of metric (always as a gauge, as requested by *HDO*), its value, and its labels. Each of the metrics is appended to the metrics array, which will be returned by the function.

Code listing 6.19: The Linecard collector.

```

1 package collector
2
3 import (
4     ... //Removed imports in this example for less space
5 )
6
7 // /rest/linecard
8 type lSBCdata struct {
9     XMLName      xml.Name  'xml:"root"'
10    LinecardData LinecardData 'xml:"linecard"'
11 }
12 type LinecardData struct {
13     Href          string  'xml:"href,attr"'
14     Rt_CardType   string  'xml:"rt_CardType"'
15     Rt_Location   string  'xml:"rt_Location"'
16     Rt_ServiceStatus int    'xml:"rt_ServiceStatus"'
17     Rt_Status     int     'xml:"rt_Status"'
18 }
19
20 func LinecardCollector2() (m []prometheus.Metric) {
21     hosts := config.GetIncludedHosts("linecard")//retrieving targets for this
22         collector
23     if (len(hosts) <= 0) {
24         log.Print("no hosts, linecard")
25         return nil
26     }
27 }

```

```

25 }
26 var ( // Declaring Prometheus descriptions used when defining metrics
27     Rt_ServiceStatus = prometheus.NewDesc("rt_ServiceStatus",
28         "The service status of the module.",
29         []string{"hostip", "hostname", "job", "linecardID", "rt_CardType", "
30             rt_Location"}, nil,
31     )
32     Rt_Status = prometheus.NewDesc("rt_Status",
33         "Indicates the hardware initialization state for this card.",
34         []string{"hostip", "hostname", "job", "linecardID"}, nil,
35     )
36 )
37 for i := range hosts {
38     phpsessid, err := http.APISessionAuth(hosts[i].Username, hosts[i].Password,
39         hosts[i].Ip)
40     if err != nil {
41         log.Print("Error auth", hosts[i].Ip, err)
42         continue
43     }
44
45     //chassis labels from db or http
46     chassisType, _, err := utils.GetChassisLabels(hosts[i].Ip, phpsessid)
47     if err != nil {
48         chassisType = "db chassisData failed"
49         log.Print(err)
50     }
51
52     var linecardID []string
53     // There are two linecard linecardIDs which are different for type of SBC
54     // router
55     if (chassisType == "SBC1000") {
56         linecardID = []string {"7", "8"}
57     } else if (chassisType == "SBC2000") {
58         linecardID = []string {"1", "2"}
59     } else {
60         //Couldnt fetch chassis type from db or http: try next host
61         continue
62     }
63     for j := range linecardID {
64         url := "https://" + hosts[i].Ip + "/rest/linecard/" + linecardID[j]
65         _, data, err := http.GetAPIData(url, phpsessid)
66         if err != nil {
67             log.Print(err)
68             continue
69         }
70
71         lData := &lSBCdata{}
72         err = xml.Unmarshal(data, &lData) //Converting XML data to variables
73         if err != nil {
74             log.Print("XML error linecard", err)
75             continue
76         }
77         labelCardType := lData.LinecardData.Rt_CardType
78         labelLocation := lData.LinecardData.Rt_Location
79         metricValue3 := float64(lData.LinecardData.Rt_ServiceStatus)
80         metricValue4 := float64(lData.LinecardData.Rt_Status)
81
82         m = append(m, prometheus.MustNewConstMetric(Rt_ServiceStatus, prometheus

```

```

        .GaugeValue, metricValue3, hosts[i].Ip, hosts[i].Hostname, "
        linecard", linecardID[j], labelCardType, labelLocation))
82     m = append(m, prometheus.MustNewConstMetric(Rt_Status, prometheus.
        GaugeValue, metricValue4, hosts[i].Ip, hosts[i].Hostname, "linecard
        ", linecardID[j]))
83     }
84 }
85 return m
86 }

```

---

### 6.5.3 System Collector

The system collector collects data associated with the use of system resources on the SBCs. The system collector's layout is slightly different from the other collectors in that it includes an error gauge. In all other collectors, as soon as an error occurs in the code, it uses the "continue" statement in its current for loop. This breaks from the current loop and begins the next step. As requested by *HDO*, the system collector is meant to provide an error metric to their Prometheus server, indicating if a Scrape is successful or not, represented as the values 1 or 0 respectively.

This error metric is appended to the collector's metric array, followed by a continue statement, or if the Scrape was successful; at the end of the outer loop. Each metric in the system collector also includes the labels; "SBC type" and "serial number".

### 6.5.4 Routingentry Collector

Almost none of the collectors uses exactly the same approach. Some Prometheus Collectors required multiple API calls to collect data on the same hosts before the final data could be collected. "Routingentry" is an example of such a collector. In this case, the API calls fetch data containing routing tables, and then runs API calls on each of these tables, fetching all existing entries for each router.

*HDO* wanted all the data from these calls, which means that each metric would contain the labels; host IP address, routing table number, routing table entry, and the metric values for all of these metrics. As a result, the final number of metrics for this collector could be several dozens depending on the number of hosts, routing tables, and entries.

The following is an example of what such a metric looks like in Prometheus:

```

rt_RoundTripDelay{hostip="10.233.230.11",
hostname="host2",job="routingentry",
routing_entry="2",routing_table="2"} 9999

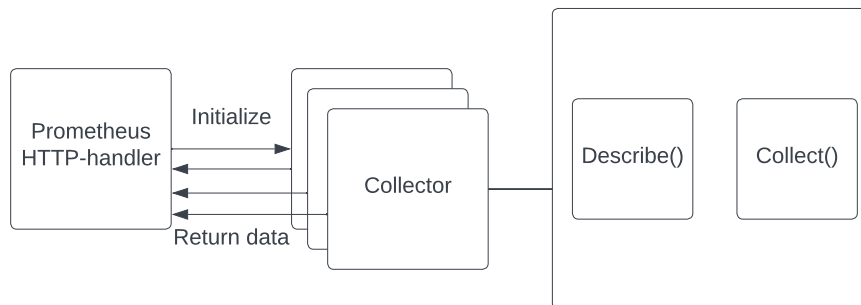
```

As discussed under the Database package in Section 6.4.3, the data members routing tables and routing entries are stored in the database for a number of hours

specified by the user in the configuration file. For our employer *HDO*, they are usually stored for 24 hours.

### 6.5.5 Comprehensive Changes to the Implementation

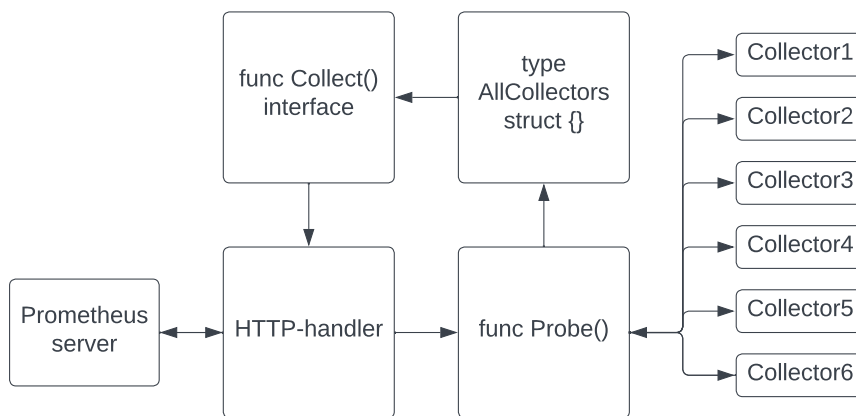
By the end of development, in early May, it was decided to change the implementation involving the collector package, in particular. It was realized halfway through development that changing the structure of the code in several ways would be highly preferable, although it was decided not to until the end of development. The previous version had one Collect and Describe interface for each collector (see Section Prometheus Package 6.3), as can be seen in Figure 6.5. The figure shows a HTTP-handler provided by the Prometheus package, which initializes all collectors concurrently. The new version moved the code previously residing in each Collect interface into new functions. This means that logically, there is now only one collector, having separate functions handling separate data groupings. Nevertheless, for simplicity purposes, each of these functions was referred to as collectors.



**Figure 6.5:** The first version of the implementation.

The previous version had the drawbacks of making it difficult to refactor or add new code. The prospect of changing this implementation was ignored until *HDO* discovered an efficiency and security issue after testing the product. It was discovered that each time the exporter made a connection to an SBC, five to six simultaneous but separate authentications took place instead of one per host, causing unnecessary use of system resources by the SBCs. The cause of this problem was due to the fact that each exporter collector was running concurrently by the Prometheus package and therefore previous database entries containing a PHP Session Cookie did not exist, meaning that each collector would fetch a new cookie instead.

The new version as shown in Figure 6.6, instead runs each collector sequentially, continuously appending their metrics to the AllCollectors struct, before returning it to our custom HTTP-handler called ProbeHandler() as shown in Code listing 6.18. The run time of the code was not improved by the new version, but the advantage was that the new version was more scalable because of the removal of the rigid structure of the Collect interfaces. By running each collector sequentially, the issue of several simultaneous connections to each SBC was also fixed.



**Figure 6.6:** The current implementation version.



## Chapter 7

# Collected Data

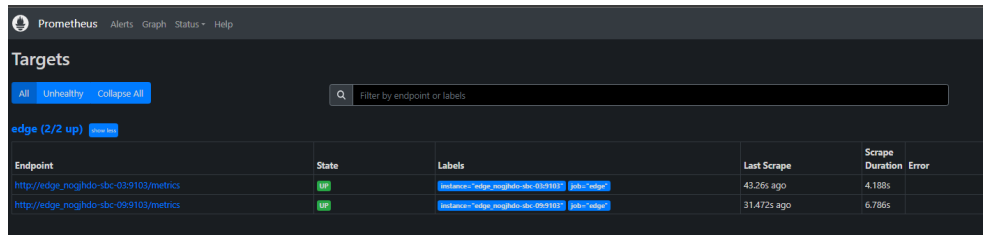
The SBC Edge REST API contains a large number of endpoints; this chapter will discuss the availability of the data that are collected, why they are collected and what kind of value they create. The list of data can be found in Appendix A.

### 7.1 Availability

The data that are collected from SBCs are available from the Ribbon SBC Edge REST API that the employer provided to the group. Data that may be collected through this REST API should always be available when the SBCs are in operation. In the case of *HDO*, these SBCs are always running, making sure that emergency calls reach and communicate with the correct reception. Therefore, the availability of the data that can be collected from the SBCs through the REST API is high.

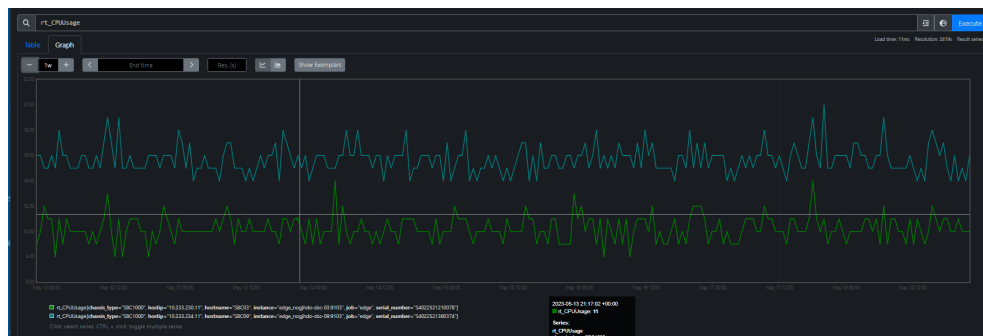
### 7.2 Why is the Data Collected?

The main goal of collecting the data from these SBC targets is to monitor them and find trends that may indicate potential hardware problems or other issues, such as call quality. Figure 7.1 shows the two SBCs targets inside of Prometheus ready to provide metrics to the Prometheus server through the exporter. As mentioned above, the goal of monitoring is to address issues before they arise. Whenever *HDO* discovers issues with the SBCs at this point, it is impossible for them to know the causes because the data are usually only stored for one hour on the SBCs. Therefore, data collection has the potential to create value for *HDO* if the data collected helps operators at *HDO* address issues that otherwise would cause problems in the network.



**Figure 7.1:** This figure illustrates two test SBCs inside of Prometheus as targets. It is very small and difficult to see, so you may have to zoom in to get a better picture.

An example of such data would be the CPU utilization as shown in Figure 7.2. High CPU usage can indicate that the router is struggling to handle the amount of data traffic on the network, which can result in degraded network performance, including dropped calls, poor call quality, or slow data transfer rates. If *HDO*'s operators catch the trend of high CPU usage and address the issue before it becomes an issue for emergency calls, then the value that collecting data creates is immense.



**Figure 7.2:** This figure illustrates the CPU usage of the two aforementioned SBCs. It is very small and difficult to see, so you may have to zoom in to get a better picture.

### 7.3 Data Groups Used by the Exporter

This section discusses the data groups defined by the SBC API documentation [51], and these data groups are also defined as such in each Prometheus Collector of the exporter. All quoted text is taken from the SBC API documentation, and examples of metrics are from its usage in the exporter.



### System call

This data group gives, among other information, information about the success rate of all calls. Examples of metrics:

- **"rt\_NumCallFailed"**: "Total number of failed calls system wide since system came up."
- **"rt\_NumCallAttempts"**: "Total number of call attempts system wide since system came up."
- **"rt\_NumCallUnAnswered"**: "Number of unanswered calls system wide since system came up."

### Disk Partition

This data group contains statistics of the resource usage of each disk partition. Examples of metrics:

- **"rt\_CurrentUsage"**: "Amount of memory used by this partition, expressed as percentage"
- **"rt\_MemoryAvailable"**: "Amount of memory in bytes, available for use in the file system."
- **"rt\_PartitionType"**: "Identifies the user-friendly physical device holding the partition."

### Ethernetport

This data group contains ethernet port statistics that are specific to this technology. Examples of metrics:

- **"rt\_ifInOverSizedPkts"**: "Displays the number of Oversized Packet errors detected on this port."
- **"rt\_ifInUnknwnProto"**: "Displays the number of Unknown Protocol errors detected on this port."
- **"rt\_ifOutBroadcastPkts"**: "Displays the number of transmitted broadcast packets on this port."

### Linecard

This data group contains information about each line card. Examples of metrics:

- **"rt\_ServiceStatus"**: "The service status of the module."
- **"rt\_Status"**: "Indicates the hardware initialization state for this card."

### Routingentry

This data group contains information about the routing table statistics. Examples of metrics:

- **"rt\_RuleUsage"**: "Displays the number of times this call route has been selected for a call."
- **"rt\_RoundTripDelay"**: "Displays the average round trip delay for this call route."
- **"rt\_QualityFailed"**: "Displays if this call route is currently passing or failing the associated quality metrics. If true, then the rule is failing, if false, then it is passing."

### System

This data group contains information about the utilization of the system resource by the SBC. Examples of metrics:

- **"rt\_CPUUsage"**: "Average percent usage of the CPU."
- **"r\_MemoryUsage"**: "Average percent usage of system memory."
- **"rt\_CPULoadAverage1m"**: "Average number of processes over the last one minute waiting to run because CPU is busy."

# Chapter 8

## Usage

This chapter explains how to install, configure, and run the exporter either using or not using Docker. After the product is installed, configured, and started, metrics can then be gathered from:

[exporter-hostip:5123/metrics](http://exporter-hostip:5123/metrics)

The first step is to either download the source code from the Git repository [68], or download a Docker image from Docker Hub as described in Section 8.2.1. This setup presumes that the Docker engine is already installed.

### 8.1 Configuration of the Exporter

The exporter requires configuration in the "config.yml" file, which is located in the root folder of the source code. In the config as shown in Code Listing 8.1 you can see the layout of a config.yml file that contains 3 hosts with dummy data. It is required to use a hostname, ipaddress, username, and password. You can choose which collectors you want to exclude for each host by adding them to the list "exclude" as shown below the last host. The name of the collectors has to match exactly as spelled in this example shown in Code Listing 8.1.

"Authtimeout" is the maximum time chosen to attempt authentication to a host. It is usually not reachable if the duration is more than 1-2 seconds.

"routing-database-hours" is the duration for which the data related to the routingentry collector are stored within the database.

Fetching new data through HTTP takes several extra seconds per scrape. Metrics are never stored, only data such as routing tables and their routing entries. It is recommended not to configure too many hosts per Docker instance because of performance issues.

Code listing 8.1: Layout of the configuration file.

---

```
1 ---
2 authtimeout: 3 #all hosts will have max 3 sec timeout
3 hosts:
4 - hostname: Host1
5   ipaddress: 11.111.111.11
6   username: Username1
7   password: Password1
8   routing-database-hours: 24 #For routingentry collector, data is
   stored in the database for 24 hours for this host.
9 - hostname: Host2
10  ipaddress: 11.111.111.12
11  username: Username2
12  password: Password2
13  routing-database-hours: 24
14  exclude: # Collectors that are excluded for this host
15    - diskpartition
16    - linecard
17 - hostname: Host3
18  ipaddress: 11.111.111.13
19  username: Username3
20  password: Password3
21  routing-database-hours: 24
22  exclude:
23    - routingentry
24    - system
25    - diskpartition
26    - systemcallstats
27    - linecard
28    - ethernetport
```

---

## 8.2 Installation and Deployment

This section walks you through the installation and deployment of the exporter.

### 8.2.1 Deployment of the Exporter as a Docker Image

There are two alternatives to deploy the exporter as a Docker image. You can run the docker file provided in the source code, or download a Docker image from the Docker Hub [69].

### Deployment Running the Provided Docker File

Navigate to the source code directory and build the Docker image by running the command:

```
sudo docker build -t edge_exporter .
```

Initialize a container instance:

```
sudo docker run -p 5123:5123 edge_exporter
```

### Deployment of the Docker Image from Docker Hub

Download the Docker image from the groups repository from Docker Hub [69]

```
docker pull sondrjor/edge_exporter:latest
```

When pulling the Docker image from Docker Hub you have to configure and run an external config.yml file:

```
sudo docker run -v path/to/your/config.yml:/usr/src/exporter/config.yml  
sondrjor/edge_exporter
```

## 8.2.2 Installation and Deployment Without Docker

The exporter is developed and tested for the official Ubuntu server image [48].

Download Golang using the official download page: "install golang", and remember to reboot. To start the exporter and download all necessary packages, navigate to the edge\_exporter directory and run "go install".

### Run or Test the Exporter

Use "go build ." or "go run ." from the main directory of the code, then use [curl localhost:5123/metrics](http://localhost:5123/metrics) in another shell window to view metrics data from the SBCs.

### Installation of Go on HDO's Virtual Machines

Because of security measures, root folders are not accessible on *HDO*'s VMs, therefore it is needed to install Go to the home directory.

- Download the latest version of Go to home directory, from Go's official website [70]
- Unzip the file with tar
- Run the following commands:

```
export GOPATH=$HOME/go
export PATH=$PATH:$GOPATH/bin
```

If starting Go gives you a message that it has not yet been installed, create a start script that executes the line `source .bashrc` from the home directory.

## 8.3 Monitoring Metrics Produced by the Exporter

In this bachelor's thesis the group has delimited from discussing the usage of Prometheus and Grafana, because our employer *HDO* already has an existing Prometheus and Grafana solution in use. However, for cases where the product is going to be used as open source by other parties than *HDO*, it will be discussed how to install and deploy these technologies using either Grafana cloud or Grafana hosted locally, both with Docker compose. This setup assumes that Docker is already installed. Section 9.1.4 discusses security concerns that should be addressed when using this setup.

- **Grafana Hosted Locally** Navigate to the following directory of the source code:

```
edge_exporter/0ther/Grafana-Prometheus/grafanalocal/
```

This directory contains the configuration files used for Grafana hosted locally. Execute the command `sudo docker-compose up -d`. The Grafana dashboard can then be found at `hostip:9090`

- **Grafana Cloud** Navigate to the following directory of the source code:

```
edge_exporter/0ther/Grafana-Prometheus/grafanacloud/
```

This directory contains the configuration files used for Grafana Cloud. Follow the setup of how to create a Grafana API key [71], and insert the username and API key under variable `basic-auth` in the `prometheus.yml` config file. Execute the command `sudo docker-compose up -d`. The Grafana dashboard can then be found at `hostip:9090`

## Chapter 9

# Discussion

This chapter will discuss the security aspects related to the product that has been developed and some technical difficulties discovered after testing by *HDO* and how they were addressed. Additionally, different aspects of the project related to the process of creating the product and thesis will be discussed.

### 9.1 Technical Discussion

The technical discussion section of the discussion chapter will discuss security concerns related to the product. It will also discuss some of the technical difficulties that the product might face in the future.

#### 9.1.1 Preexisting Security Measures

As mentioned in the introduction, *HDO* is responsible for delivering services that realize the Norwegian emergency call network. This is critical infrastructure and security is a high priority. The entire network is closed off from the public Internet and can be accessed by connecting to a Virtual Private Network (VPN). To connect to the network, one must also go through a two-factor authentication system. Furthermore, *HDO* follows the zero trust security model, also known as Zero Trust Architecture (ZTA) [72], which means that no device is trusted by default. All ports in the network are closed off by default and need to be opened by an administrator to allow access. Another feature of *HDO*'s network is that everything is microsegmented. This is the act of dividing the network into segments, separating workload and applying network rules to it, in *HDO*'s case of zero trust. All data flowing through *HDO*'s network are also securely transferred in encrypted channels. In addition to all these security measures, *HDO* uses a firewall to prevent devices from accessing the rest of the network.

### 9.1.2 API Authentication

One notable security concern regarding the product was how to handle authentication information, specifically API authentication. In order to make use of the SBC's integrated API, a username and password is required. As requested by the employer, this information is to be included in the product's configuration file. This means that the inclusion of the username and password in the configuration file is required for the exporter to successfully retrieve data using the SBC Edge REST API. These passwords are not hashed and are therefore written in plain text as part of the configuration.

This is obviously a security concern that the group acknowledges, which was also discussed and passed on to *HDO*. However, this configuration feature was requested, and because there were no simple solutions or workarounds to this problem, the risk remains. A possible risk is that someone will use SQL injection [73] to perform malicious activity, such as destroying a database or adding false information to a database, if they get access to the REST API. A common way to solve this problem is to use API tokens for authentication. However, the REST API that is available on the SBCs provided does not have a solution to authenticate with the API tokens, which would have been the better solution. An argument can be made against the use of API tokens, because all traffic in *HDO*'s network is transferred over encrypted channels. Additionally, authenticating to the SBCs like how it is being done now is a simple solution, and implementing a solution with API tokens would require even more development, which would take more time and add more complexity.

But what potential risk does this problem pose? The potential danger would be if someone accesses *HDO*'s private network, and furthermore accesses an instance of the exporter, retrieving the username and password from the configuration file. With this information, the attacker is able to delete the configuration file rendering the exporter useless until fixed. Another possibility is to gain access to these SBCs and perform a DoS attack and potentially crash one of these SBCs. Our employer at *HDO* managed to crash these SBCs while testing as a result of too many requests sent to them.

However, the amount of harm one can cause in relation to this information is limited to the use of the *HDO* Edge REST API that the exporter was monitoring. In this hypothetical scenario, someone was able to access *HDO*'s internal network, bypassing several security measures to access instances of the exporter. The potential harm related to this API authentication is rather insignificant compared to the potential harm that such access would already allow. Meaning that access at this level, necessary to access the exporter, is a much greater threat than access to the API authentication. However, this does not suggest that security related to the exporter and its stored API authentication should not be taken seriously and



ideally passwords should always be hashed according to best practice, in order to limit access. Additionally, the SBCs should all have different passwords and usernames to prevent damage caused by reuse of login credentials for the SBCs.

### 9.1.3 Security of the Code

As mentioned above when discussing API authentication in subsection 9.1.2, the employer at *HDO* managed to crash one of these SBCs as a result of too many requests via its REST API. This would certainly be classified as a weakness of our code, being able to crash these SBCs. The employer mentioned that peak CPU usage of the SBC was very high at one point. This might have caused the SBC to crash. A possible solution to this issue is to exclude some of the collectors. This would reduce the number of requests, and as a result the SBCs might not crash. However, this also means that some of the metrics will not be collected and will not be monitored. Another measure is to increase the scrape interval in the configuration to reduce the overall usage, thus reducing the strain of the CPU.

According to Snyk Advisor, an open source advisor that evaluates whether a package is safe to use or not [74], many of the packages that were used for the implementation are safe to use. This includes Go packages such as Prometheus [75], sqlite [76], yaml [77], ioutil [78] and net [79]. Whether Snyk Advisor is trustworthy or not is difficult to say because of the lack of relevant research on this topic. However, the group could not find any reports or articles that discussed any security concerns of the packages used that have not been fixed already.

### 9.1.4 Using the Exporter as Open Source

The use of the product is similar in nature when used by other parties than *HDO*. As *HDO* already has Prometheus systems in use, we decided to delimit from discussing Prometheus and Grafana monitoring systems in greater detail. If other corporations wish to use the product to monitor their SBC routers, they will have to integrate it with a Prometheus and Grafana monitoring solution on their own terms. As a starting point and possible solution to deploy Grafana, such a solution is provided in Section 8.3. Here we discuss how to monitor metrics with Prometheus and Grafana when the product is being used as open source.

The option of using "Grafana Cloud" [80] however, has some significant security risks because the ports and IP traffic are not encrypted. As information such as SBC IP addresses, authentication details and sensitive data concerning networking infrastructure flows through the Internet. When using this monitoring solution in integration with the exporter, it is highly recommended to implement additional security measures such as Transport Layer Security (TLS) in Prometheus [81].

The option of using Grafana hosted locally with docker is only secure if the solution resides within a secure infrastructure because the ports are not secured

by default in this option either.

Other parties than *HDO* that wishes to use the exporter may find that the product may not be suited to their needs, which is understandable because it has been developed in cooperation with *HDO* and therefore is specifically tailored to their needs and suggestions. However, the product to some degree provides customization to what type of data that one decides to collect (see Section 8.1. As it is open source, anyone can also further develop the source code to fit their needs.

### 9.1.5 Issues That Were Revealed During Testing

After some testing of the product by *HDO* on their SBC hosts and Prometheus server, some technical issues were revealed.

*HDO* had initially overestimated the capacity of the SBCs to handle the load the exporter would place on them. This was not an issue with the exporter's performance, but rather the demands *HDO* needed the exporter to pose on the SBCs in terms of requesting data. As the amount of data to be monitored was quite large, it resulted in the SBC's CPU usage to increase by up to 40% in some cases. To handle this issue, it was decided to increase the scrape intervals from 15 seconds to 1 minute. The employers are also considering the possibility of excluding certain collectors that are deemed less important, such as disk partition, in order to reduce the workload.

The first tests revealed that the exporter authenticated to the SBCs up to six times in a matter of seconds, although it should have only been once per 9 minutes. This issue was the reason for our decision to change the overall architecture of the exporter, as discussed in Section 6.5.5.

The run time of the exporter turned out to be quite large, which is mainly due to all the API calls that were made to the SBCs. For example, the "routingentry" collector had to make in most cases several dozens API calls to the SBCs to fetch all the data that *HDO* wanted. As a result, this collector alone took six seconds to complete. It was decided to try to improve this issue and the use of a database was implemented to do so, as described in Section 6.4.3. This was considered a good solution that was able to save several seconds.

## 9.2 Project Execution

The project execution section of the discussion chapter will go through the communication between group members, supervisor, and employer. Additionally, it will further discuss the working process between group members, project planning, meetings between group members, supervisor and employer, time tracking, report writing, and the development model.

### 9.2.1 Communication

Communication between group members, supervisor and employer happened over two main applications. Most of the communication between the groups' members took place over Discord. However, before the Discord server for the group was established in the middle of January, all communication was done through Microsoft Teams. Communication between group members, supervisor, and employer occurred mainly via Microsoft Teams. However, some of the communication also happened via email.

### 9.2.2 Working Process

Prior to the first introductory lecture on the Bachelor's thesis, the members of the group had very little to no experience working together. Additionally, none of the group members knew each other on a personal level. After the first lecture the group had their first meeting and began creating a project plan, contact their supervisor, plan how the group would work in the future with regards to meetings, and lastly they delegated each member of the group their respective role.

At first, the meetings and teamwork went relatively smoothly. However, towards the end of January, the momentum of the group decreased and the communication within the group could have been better. The group also decided to have only one person work on the development of the product, while the others worked on the report. In hindsight, we should not have made this decision, as it worsened communication within the group by separating members.

### 9.2.3 Planning

The initial planning of the project was carried out mainly in the first month of the project in January as part of the project plan. In the project plan, the group planned how they would work on the project, milestones, meetings, tools that they would use, risks and more (see Appendix D).

### 9.2.4 Meetings

During the creation of the project plan, the group initially planned to have two physical meetings every week. In these meetings, the group would both work on

the project together and discuss the work that had been done the previous week. Additionally, the group would plan the work that was to be done for the current week of the meeting. One of these meetings was planned to take place after the meeting with the supervisor on Thursdays, and the other on a different day of the week.

The group managed to maintain the schedule for the first month. However, after the first month, the group had fewer weekly meetings. As a result of the lack of meetings between group members, the quality and speed of the workflow seemed to be a little less than desired. Looking back, a greater amount of effort in the planning phase would have been beneficial. A well-planned and structured project would have helped with many of the issues that the group faced during the execution of both the development of the project and the writing of the report.

Weekly meetings with the supervisor were priceless in terms of help and guidance. Not only did the supervisor help with technical difficulties related to the development of the product, but they also helped the group a lot with the project report by giving suggestions and commenting on what could be improved. Finally, they also helped the group with internal communication problems.

### **9.2.5 Time Tracking**

After some back and forth discussion at the early stages of the project in January, the group decided to fill in a shared Microsoft Excel document as their method of tracking time spent working on the bachelor's thesis. This document can be found in Appendix H. The main reason the group chose this tool was because of its simplicity. Another reason for ending up using Excel was because of the group's inexperience using other time tracking tools. Furthermore, these tools seemed unnecessarily cumbersome to use for the group.

Looking back at this decision and having gathered some valuable insight from the experience of other groups who used other time tracking tools. It might have been a better idea to use a different method or tool to track time other than Excel, despite them being more complex to use. One of the tools that some other groups used is called Toggl Track [82], which tracks time spent in real time. This helped push the members to work for the amount of hours they had planned.

The experience that the group received from using Excel as a tracking time solution was that it was quite inaccurate. The reason the group experienced the use of Excel for time tracking as inaccurate is due to human errors related to time tracking. These human errors include, but are not limited to, simply forgetting how long they spent on working or forgetting to log hours completely. This often resulted in an inaccurate representation of the amount of hours logged.

### 9.2.6 Report

The group began writing the project report in the middle of February. At first, the writing of the report was slow, as only one of the group members was working on it. The supervisor suggested that the group send him three completed chapters before Easter break. About two weeks before this deadline, the group only had one chapter finished and about half a chapter in the works. During the next weeks, the pace picked up as the developer of the product changed focus and assisted on the project report. The last member of the project also began contributing to the project report at this time.

Due to the decision to divide the responsibilities with respect to the development of the product and the writing of the report, the writing often halted. Many chapters of the report required knowledge about the product being developed. Having only one member developing it and having the most knowledge about it left the other group members with little of the necessary knowledge required to write some of these chapters.

In hindsight, the group should have had at least two of the members develop the product. This would probably have resulted in a better written report thanks to more people being able to write and help other members write some of the chapters. Having two developers might also have resulted in the product being finished ahead of schedule, meaning more time for writing the report.

### 9.2.7 Development Model

During the writing of the project plan, the group discussed which development model they would use to develop the product and write the project plan. The group ended up agreeing on using the Scrum model.

However, the use of the Scrum development model throughout the project was quite limited. This was mainly due to the fact that the group lacked a sufficient number of meetings to properly follow the scrum model. Additionally, the group did not assign Scrum roles and responsibilities. This led to a messy and unstructured use of the Scrum development model. However, the little use of Scrum was perceived as useful to structure tasks for the week and to discuss what had been done the week before.

The group also originally planned to use a CI/CD pipeline for the development of the product. However, the development of the exporter was proven to be a greater task than anticipated, and the group decided to prioritize the development of the exporter without an integrated CI/CD pipeline. The group felt that the addition of the CI/CD pipeline would take a long time to integrate and decided not to prioritize it and instead focused on spending time developing a better product.



# Chapter 10

## Conclusion

This chapter concludes the report and discusses what the group has been able to achieve. Furthermore, it will discuss further work related to the project and the possible improvements that can be made to the product.

### 10.1 What Has the Group Achieved?

As a result of working on this project, the group has achieved a working monitoring solution that *HDO* can add to their current repertoire of monitoring solutions. The monitoring solution that the group has created is called a Prometheus exporter and will be used by *HDO* to monitor the SBCs in their infrastructure to find potential faults and address them before they cause issues with the quality of emergency calls. In addition, the group has learned a lot about working as a group.

### 10.2 Further Work

There are some improvements that could be made for the exporter to make it more efficient and convenient for *HDO*. These will be discussed in this section.

#### 10.2.1 Potential Improvements

During development of the exporter, it was discovered that data can be stored in memory as long as the program is running. It is even possible for different functions to interact with the same data stored here. However, in the first architecture version, the group failed to implement this for reasons that we do not fully understand. As soon as each of the Collect interfaces was completed, all data stored in memory were removed. There may have been two reasons for this;

1. It may have been necessary to use interfaces for this to be possible. In the new architecture, the exporter successfully stored Prometheus metrics in memory, which means that the group now knows a possible solution to store other data here as well.
2. Another possible cause is that Prometheus by default deletes all data after a scrape is executed, which means that utilizing memory storage would not be possible.

Therefore, the group used a database in all cases of reusing data, which had both its benefits and drawbacks. The benefits being the scalability of the product and saving work for the memory as a large amount of data was being used, which comes with the risks of memory thrashing [83] on smaller systems.

The drawbacks of using a database in our implementation is that each collector is opening the database to insert or retrieve data, meaning that the database was being opened and queried six times during each scrape, instead of once. This causes the exporter to use more disk and CPU. This is because all 6 collectors shared this implementation. Some performance gain could possibly be achieved by replacing the database with more memory allocation or moving the database logic into the `Probe()` function shown in the Code Listing 6.18.

### 10.2.2 Ways to Improve the Implementation

With the new custom function "Probehandler" shown in Code Listing 6.17 and its integration with the function "Probe", it is now possible to remove much of the duplicated code in the Collectors into this function. This includes the retrieval of hosts, session cookies, and all database-related functions. Data related to these elements can now be given as parameters and return values to the collectors instead, which is something to consider for further work.

During the end of development, *HDO* mentioned the possibility of renaming all metrics to be more concise, because the Prometheus systems they use contain metrics from a large number of different exporters. This means, among other aspects, that searching for metrics produced by our exporters would return a very large list of different metrics, making it difficult to separate these metrics from others.

As mentioned in Chapter 4, there were performance issues with regards to the SBCs handling the demand of each scrape. *HDO* mentioned a possibility to change the "disk partition" collector to instead collect from a smaller number of disks as this collector was particularly demanding.



# Bibliography

- [1] Ribbon Communications, *Ip optical networking and communication | ribbon*, [Online; accessed 17-May-2023]. [Online]. Available: <https://ribboncommunications.com/>.
- [2] Helsetjenestens driftsorganisasjon for nødnett HF, *Helsetjenestens driftsorganisasjon for nødnett*, [Online; accessed 17-May-2023]. [Online]. Available: <https://www.hdo.no>.
- [3] Direktoratet for samfunnssikkerhet og beredskap, *Hva er nødnett?* [Online; accessed 17-May-2023]. [Online]. Available: <https://www.nodnett.no/om-nodnett/hva-er-nodnett/>.
- [4] Ribbon Communications, *Session border controllers for service providers*, [Online; accessed 17-May-2023]. [Online]. Available: <https://ribboncommunications.com/products/service-provider-products/session-border-controllers-service-providers>.
- [5] Wikipedia contributors, *Session border controller — Wikipedia, the free encyclopedia*, [Online; accessed 1-May-2023], 2023. [Online]. Available: [https://en.wikipedia.org/w/index.php?title=Session\\_border\\_controller&oldid=1140919721](https://en.wikipedia.org/w/index.php?title=Session_border_controller&oldid=1140919721).
- [6] Docker, *Use containers to build, share and run your applications*, [Online; accessed 17-May-2023]. [Online]. Available: <https://www.docker.com/resources/what-container/>.
- [7] Grafana, *Grafana | query, visualize, alerting observability platform*, [Online; accessed 17-May-2023]. [Online]. Available: <https://grafana.com/grafana/>.
- [8] Prometheus Authors, *Overview*, [Online; accessed 16-April-2023], 2023. [Online]. Available: <https://prometheus.io/docs/introduction/overview>.
- [9] Grafana, *Grafana loki*, [Online; accessed 17-May-2023]. [Online]. Available: <https://grafana.com/oss/loki/>.
- [10] NTNU, *Digital infrastruktur og cybersikkerhet*, [Online; accessed 17-May-2023]. [Online]. Available: <https://www.ntnu.no/studier/bdigsec>.
- [11] NTNU, *Department of information security and communication technology*, [Online; accessed 17-May-2023]. [Online]. Available: <https://www.ntnu.edu/iik>.

- [12] Wikipedia contributors, *Event monitoring — Wikipedia, the free encyclopedia*, [Online; accessed 03-April-2023], 2023. [Online]. Available: [https://en.wikipedia.org/w/index.php?title=Event\\_monitoring&oldid=1136480691](https://en.wikipedia.org/w/index.php?title=Event_monitoring&oldid=1136480691).
- [13] Paul Dix, *Why time series matters for metrics, real-time analytics and sensor data*, [Online; accessed 13-April-2023], 2021. [Online]. Available: <https://get.influxdata.com/rs/972-GDU-533/images/why%20time%20series.pdf>.
- [14] MongoDB, *What is mongodb?* [Online; accessed 17-May-2023]. [Online]. Available: <https://www.mongodb.com/what-is-mongodb>.
- [15] Graphite, *Overview*, [Online; accessed 17-May-2023]. [Online]. Available: <https://graphiteapp.org/#overview>.
- [16] InfluxData, *Introducing influxdb 3.0*, [Online; accessed 17-May-2023]. [Online]. Available: <https://www.influxdata.com/products/influxdb-overview/>.
- [17] D. Namiot, 'Time series databases,' *DAMDID/RCDL*, vol. 1536, pp. 132–137, 2015.
- [18] T. Pelkonen, S. Franklin, J. Teller, P. Cavallaro, Q. Huang, J. Meza and K. Veeraraghavan, 'Gorilla: A fast, scalable, in-memory time series database,' *Proc. VLDB Endow.*, vol. 8, no. 12, pp. 1816–1827, 2015, ISSN: 2150-8097. DOI: 10.14778/2824032.2824078. [Online]. Available: <https://doi.org/10.14778/2824032.2824078>.
- [19] IBM, *What is containerization?* [Online; accessed 13-April-2023]. [Online]. Available: <https://www.ibm.com/topics/containerization>.
- [20] Axigen, *5 benefits of containerization — limitless computing speed and agility*, [Online; accessed 13-April-2023], 2021. [Online]. Available: [https://www.axigen.com/articles/benefits-of-containerization\\_117.html](https://www.axigen.com/articles/benefits-of-containerization_117.html).
- [21] J. Turnbull, *The Docker Book: Containerization is the new virtualization*. James Turnbull, 2014.
- [22] A. M. Potdar, D. Narayan, S. Kengond and M. M. Mulla, 'Performance evaluation of docker container and virtual machine,' *Procedia Computer Science*, vol. 171, pp. 1419–1428, 2020.
- [23] IBM, *What is container orchestration?* [Online; accessed 14-April-2023]. [Online]. Available: <https://www.ibm.com/topics/container-orchestration>.
- [24] Kubernetes, *Overview*, [Online; accessed 14-April-2023], 2023. [Online]. Available: <https://kubernetes.io/docs/concepts/overview>.
- [25] M. Masse, *REST API design rulebook: designing consistent RESTful web service interfaces*. " O'Reilly Media, Inc.", 2011.

- [26] Prometheus Authors, *Understanding and using the multi-target exporter pattern*, [Online; accessed 17-May-2023]. [Online]. Available: <https://prometheus.io/docs/guides/multi-target-exporter/>.
- [27] Prometheus Authors, *Metric types*, [Online; accessed 17-May-2023]. [Online]. Available: [https://prometheus.io/docs/concepts/metric\\_types/](https://prometheus.io/docs/concepts/metric_types/).
- [28] Prometheus Authors, *Exporters and integrations*, [Online; accessed 16-April-2023], 2023. [Online]. Available: <https://prometheus.io/docs/instrumenting/exporters>.
- [29] Shivang Sarawagi, *What is grafana? why use it? everything you should know about it*, [Online; accessed 16-April-2023]. [Online]. Available: <https://scaleyourapp.com/what-is-grafana-why-use-it-everything-you-should-know-about-it/>.
- [30] MySQL Authors, *What is mysql?* [Online; accessed 17-May-2023]. [Online]. Available: <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>.
- [31] Elasticsearch Authors, *What is elasticsearch?* [Online; accessed 17-May-2023]. [Online]. Available: <https://www.elastic.co/what-is/elasticsearch>.
- [32] Docker Authors, *Docker overview*, [Online; accessed 17-May-2023]. [Online]. Available: <https://docs.docker.com/get-started/overview/>.
- [33] Microsoft Authors, *Microservice architecture style*, [Online; accessed 16-April-2023], 2023. [Online]. Available: <https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices>.
- [34] IBM, *What is containerization?* [Online; accessed 18-April-2023]. [Online]. Available: <https://www.ibm.com/topics/containerization>.
- [35] Go, *Golang*, [Online; accessed 16-May-2023], 2023. [Online]. Available: <https://go.dev/>.
- [36] Wikipedia contributors, *Go (programming language) — Wikipedia, the free encyclopedia*, [Online; accessed 16-May-2023], 2023. [Online]. Available: [https://en.wikipedia.org/w/index.php?title=Go\\_\(programming\\_language\)&oldid=1153277087](https://en.wikipedia.org/w/index.php?title=Go_(programming_language)&oldid=1153277087).
- [37] Wikipedia contributors, *Xml — Wikipedia, the free encyclopedia*, [Online; accessed 14-May-2023], 2023. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=XML&oldid=1155412116>.
- [38] Tuan Nguyen, *Golang performance comparison | why is go fast?* [Online; accessed 30-April-2023]. [Online]. Available: <https://www.golinuxcloud.com/golang-performance/>.
- [39] Go, *Documentation*, [Online; accessed 30-April-2023]. [Online]. Available: <https://go.dev/doc/>.

- [40] Go, *Prometheus*, [Online; accessed 28-April-2023]. [Online]. Available: [https://pkg.go.dev/github.com/prometheus/client\\_golang/prometheus#pkg-index](https://pkg.go.dev/github.com/prometheus/client_golang/prometheus#pkg-index).
- [41] Anan Hossain, *Go vs php syntax comparison*, [Online; accessed 30-April-2023], 2019. [Online]. Available: <https://engineering.carsguide.com.au/go-vs-php-syntax-comparison-c1465380b8ff>.
- [42] Go, *Help*, [Online; accessed 30-April-2023]. [Online]. Available: <https://go.dev/help>.
- [43] GitHub Authors, *Github: Let's build from here*, [Online; accessed 17-May-2023]. [Online]. Available: <https://github.com/>.
- [44] Stack Overflow Authors, *Stack overflow - where developers learn share & build careers*, [Online; accessed 17-May-2023]. [Online]. Available: <https://stackoverflow.com/>.
- [45] Git, *Reference*, [Online; accessed 05-May-2023], 2023. [Online]. Available: <https://git-scm.com/docs>.
- [46] CURL, *Curl*, [Online; accessed 10-May-2023], 2023. [Online]. Available: <https://curl.se/>.
- [47] Scrum, *Scrum*, [Online; accessed 10-May-2023], 2023. [Online]. Available: <https://www.scrum.org/resources/what-scrum-module>.
- [48] Ubuntu, *Get ubuntu server*, [Online; accessed 30-April-2023], 2023. [Online]. Available: <https://ubuntu.com/download/server>.
- [49] Christian Svensson, *Fortigate\_exporter*, [Online; accessed 28-April-2023]. [Online]. Available: [https://github.com/bluecmd/fortigate\\_exporter/](https://github.com/bluecmd/fortigate_exporter/).
- [50] Anton Putra, *Nginx-exporter*, [Online; accessed 10-May-2023], 2023. [Online]. Available: <https://github.com/antonputra/tutorials/tree/main/lessons/141/prometheus-nginx-exporter>.
- [51] R. Communications, *Sbc edge rest api documentation 9.0*, 2020.
- [52] mattn, *Go-sqlite3*, [Online; accessed 15-May-2023]. [Online]. Available: <https://pkg.go.dev/github.com/mattn/go-sqlite3>.
- [53] Wikipedia contributors, *Http handler — Wikipedia, the free encyclopedia*, [Online; accessed 20-May-2023], 2023. [Online]. Available: [https://en.wikipedia.org/w/index.php?title=HTTP\\_handler&oldid=1137776466](https://en.wikipedia.org/w/index.php?title=HTTP_handler&oldid=1137776466).
- [54] Go, *Interfaces*, [Online; accessed 28-April-2023]. [Online]. Available: <https://go.dev/tour/methods/9>.
- [55] Go, *Pointer*, [Online; accessed 28-April-2023]. [Online]. Available: <https://pkg.go.dev/golang.org/x/tools/go/pointer>.
- [56] Anshul Agarwal, *Pointers in golang*, [Online; accessed 03-May-2023]. [Online]. Available: <https://www.geeksforgeeks.org/pointers-in-golang/>.

- [57] Prometheus, *Writing exporters*, [Online; accessed 15-May-2023]. [Online]. Available: [https://prometheus.io/docs/instrumenting/writing\\_exporters/](https://prometheus.io/docs/instrumenting/writing_exporters/).
- [58] Go, *Net package*, [Online; accessed 3-May-2023]. [Online]. Available: <https://pkg.go.dev/net>.
- [59] Matt Holt, *Curl-to-go*, [Online; accessed 28-April-2023]. [Online]. Available: <https://mholt.github.io/curl-to-go>.
- [60] Go, *Time package*, [Online; accessed 3-May-2023]. [Online]. Available: <https://pkg.go.dev/time>.
- [61] Go, *Ioutil package*, [Online; accessed 3-May-2023]. [Online]. Available: <https://pkg.go.dev/io/ioutil>.
- [62] Go, *Xml package*, [Online; accessed 3-May-2023]. [Online]. Available: <https://pkg.go.dev/encoding/xml>.
- [63] Go, *Yaml package*, [Online; accessed 3-May-2023]. [Online]. Available: <https://pkg.go.dev/gopkg.in/yaml.v2>.
- [64] qwertmax, *How to read a yaml file*, [Online; accessed 28-April-2023]. [Online]. Available: <https://stackoverflow.com/questions/30947534/how-to-read-a-yaml-file>.
- [65] SQLite Authors, *What is sqlite?* [Online; accessed 17-May-2023]. [Online]. Available: <https://sqlite.org/index.html>.
- [66] Go, *Sql package*, [Online; accessed 3-May-2023]. [Online]. Available: <https://pkg.go.dev/database/sql>.
- [67] SQLite, *Defense against the dark arts*, [Online; accessed 4-May-2023], 2022. [Online]. Available: <https://www.sqlite.org/security.html>.
- [68] Sondre Jørgensen, *Edge\_exporter*, [Online; accessed 30-April-2023], 2023. [Online]. Available: [https://github.com/Sonjorg/edge\\_exporter](https://github.com/Sonjorg/edge_exporter).
- [69] Sondre Jørgensen, *Sondrjor/edge\_exporter*, [Online; accessed 30-April-2023], 2023. [Online]. Available: [https://hub.docker.com/r/sondrjor/edge\\_exporter](https://hub.docker.com/r/sondrjor/edge_exporter).
- [70] Go, *Download and install*, [Online; accessed 30-April-2023], 2023. [Online]. Available: <https://go.dev/doc/install>.
- [71] Grafana, *Create grafana cloud api keys*, [Online; accessed 01-May-2023], 2023. [Online]. Available: <https://grafana.com/docs/grafana-cloud/reference/create-api-key/>.
- [72] Wikipedia contributors, *Zero trust security model — Wikipedia, the free encyclopedia*, [Online; accessed 8-May-2023], 2023. [Online]. Available: [https://en.wikipedia.org/w/index.php?title=Zero\\_trust\\_security\\_model&oldid=1152969199](https://en.wikipedia.org/w/index.php?title=Zero_trust_security_model&oldid=1152969199).

- [73] Wikipedia contributors, *Sql injection* — *Wikipedia, the free encyclopedia*, [Online; accessed 17-May-2023], 2023. [Online]. Available: [https://en.wikipedia.org/w/index.php?title=SQL\\_injection&oldid=1154592887](https://en.wikipedia.org/w/index.php?title=SQL_injection&oldid=1154592887).
- [74] Snyk Advisor Authors, *Snyk open source avisor | snyk*, [Online; accessed 9-May-2023], 2023. [Online]. Available: <https://snyk.io/advisor/golang>.
- [75] Snyk Advisor Authors, *Prometheus*, [Online; accessed 17-May-2023]. [Online]. Available: <https://snyk.io/advisor/golang/github.com/takattila/prometheus>.
- [76] Snyk Advisor Authors, *Sqlite*, [Online; accessed 17-May-2023]. [Online]. Available: <https://snyk.io/advisor/golang/modernc.org/sqlite>.
- [77] Snyk Advisor Authors, *Yaml*, [Online; accessed 17-May-2023]. [Online]. Available: <https://snyk.io/advisor/golang/gopkg.in/yaml.v3>.
- [78] Snyk Advisor Authors, *Ioutil*, [Online; accessed 17-May-2023]. [Online]. Available: <https://snyk.io/advisor/golang/github.com/whosonfirst/go-ioutil>.
- [79] Snyk Advisor Authors, *Net*, [Online; accessed 17-May-2023]. [Online]. Available: <https://snyk.io/advisor/golang/github.com/libp2p/go-libp2p-net>.
- [80] Grafana, *Your observability platform, managed as a service*, [Online; accessed 19-May-2023], 2023. [Online]. Available: <https://grafana.com/products/cloud/>.
- [81] Prometheus Authors, *Https and authentication*, [Online; accessed 4-May-2023], 2023. [Online]. Available: <https://prometheus.io/docs/prometheus/latest/configuration/https/>.
- [82] Toggl Authors, *Simple time tracking to save you time and money*. [Online; accessed 17-May-2023]. [Online]. Available: <https://toggl.com/track/>.
- [83] Wikipedia contributors, *Thrashing (computer science)* — *Wikipedia, the free encyclopedia*, [Online; accessed 28-April-2023], 2023. [Online]. Available: [https://en.wikipedia.org/w/index.php?title=Thrashing\\_\(computer\\_science\)&oldid=1135385898](https://en.wikipedia.org/w/index.php?title=Thrashing_(computer_science)&oldid=1135385898).

## **Appendix A**

# **Data Collected**

This appendix includes a list of the data that are collected with their respective API endpoint.





## Collectors and Metrics

List of Collectors, the API endpoints they use and metrics they support:

### System Call Stats Collector

#### Endpoints:

- REST API Method: GET /rest/systemcallstats

#### Metrics:

- Rt\_NumCallAttempts - Total number of call attempts system wide since system came up.
- Rt\_NumCallSucceeded - Total number of successful calls system wide since system came up.
- Rt\_NumCallFailed - Total number of failed calls system wide since system came up.
- Rt\_NumCallCurrentlyUp - Number of currently connected calls system wide.
- Rt\_NumCallAbandonedNoTrunk - Number of rejected calls due to no channel available system wide since system came up.
- Rt\_NumCallUnAnswered - Number of unanswered calls system wide since system came up.

### Diskpartition Collector

#### API endpoints:

- REST API Method: GET /rest/diskpartition - retrieves disk identifier
- REST API Method: GET /rest/diskpartition/{identifier}

#### Metrics:

- Rt\_CurrentUsage - Amount of memory used by this partition, expressed as percentage
- Rt\_MaximumSize - Specifies the maximum amount of memory, in bytes available in this partition.
- Rt\_MemoryAvailable - Amount of memory in bytes, available for use in the filesystem.
- Rt\_MemoryUsed - Amount of memory in bytes, used by the existing files in the filesystem
- Rt\_PartitionName - The name of the disk partition.
- Rt\_PartitionType - Identifies the user-friendly physical device holding the partition.

### Ethernetport Collector

#### API endpoints:

- REST API Method: GET /rest/ethernetport - retrieve ethernetport identifier
- REST API Method: GET /rest/ethernetport/{identifier}/historicalstatistics

## Metrics:

- IfRedundancy - No information found in the Ribbon SBC Edge REST API Documentation 9.0
- IfRedundantPort - No information found in the Ribbon SBC Edge REST API Documentation 9.0
- Rt\_ifInBroadcastPkts - Displays the number of received broadcast packets on this port.
- Rt\_ifInDiscards - Displays the number of discard errors detected on this port.
- Rt\_ifInErrors - Displays the number of errors detected on this port.
- Rt\_ifInFCSErrors - Displays the number of discard Frame Check Sequence errors detected on this port.
- Rt\_ifInFragmentedPkts - Displays the number of Fragmented Packet errors detected on this port
- Rt\_ifInMulticastPkts - Displays the number of received multicast packets on this port.
- Rt\_ifInOctets - Displays the number of received octets on this port.
- Rt\_ifInOverSizedPkts - Displays the number of Oversized Packet errors detected on this port.
- Rt\_ifInUcastPkts - Displays the number of received unicast packets on this port.
- Rt\_ifInUndersizedPkts - Displays the number of Undersized Packet errors detected on this port
- Rt\_ifInUnknwnProto - Displays the number of Unknown Protocol errors detected on this port.
- Rt\_ifInterfaceIndex - No information found in the Ribbon SBC Edge REST API Documentation 9.0
- Rt\_ifLastChange - The value of sysUpTime at the time the interface entered its current operational state.
- Rt\_ifMtu - The size of the largest packet which can be sent/received on the interface.
- Rt\_ifOperatorStatus - The operational status of the interface - 0: IF\_OPER\_UP or 1: IF\_OPER\_DOWN
- Rt\_ifOutBroadcastPkts - Displays the number of transmitted broadcast packets on this port.
- Rt\_ifOutDeferredTransmissions - Displays the number of Deferred Transmission errors detected on this port.
- Rt\_ifOutDiscards - Displays the number of discard errors detected on this port.
- Rt\_ifOutErrors - Displays the number of errors detected on this port.
- Rt\_ifOutLateCollissions - Displays the number of Late Collision errors detected on this port.
- Rt\_ifOutMulticastPkts - Displays the number of transmitted multicast packets on this port.
- Rt\_ifOutOctets - Displays the number of transmitted octets on this port.

- Rt\_ifOutUcastPkts - Displays the number of transmitted unicast packets on this port.
- Rt\_ifSpeed - An estimate of the interface's current bandwidth in bits per second
- Rt\_redundancyRole - When redundancy is configured for "Failover", indicates if it's role is "Primary" or "Secondary".
- Rt\_redundancyState - When redundancy is configured for "Failover", indicates if it's state is "Online" or "Backup".

## Linecard Collector

### API endpoints:

- REST API Method: GET /rest/linecard/{identifier}

### Metrics:

- Rt\_CardType - The type of hardware module.
- Rt\_Location - The hardware module's location within the SBC.
- Rt\_ServiceStatus - The service status of the module.
- Rt\_Status - Indicates the hardware initialization state for this card.
- Routing Entry Collector
- API endpoints:
- REST API Method: GET /rest/routingtable REST API Method: GET /rest/routingtable/[routingtable]/routingentry REST API Method: GET /rest/routingtable/[routingtable]/routingentry/[routingentry]historicalstatistics/1
- Metrics:
- Rt\_RuleUsage - Displays the number of times this call route has been selected for a call.
- Rt\_ASR - Displays the Answer-Seizure Ratio for this call route. (ASR is calculated by dividing the number of call attempts answered by the number of call attempts.)
- Rt\_RoundTripDelay - Displays the average round trip delay for this call route.
- Rt\_Jitter - Displays the average jitter for this call route.
- Rt\_MOS - Displays the Mean Opinion Score (MOS) for this call route.
- Rt\_QualityFailed - Displays if this call route is currently passing or failing the associated quality metrics. If true then the rule is failing, if false then it is passing.

## System Collector

### API endpoints:

- REST API Method: GET /rest/system/historicalstatistics/1

**Metrics:**

- Rt\_CPUUsage - Average percent usage of the CPU.
- Rt\_MemoryUsage - Average percent usage of system memory.
- Rt\_CPUUptime - The total duration in seconds, that the system CPU has been UP and running.
- Rt\_FDUsage - Number of file descriptors used by the system.
- Rt\_CPULoadAverage1m - Average number of processes over the last one minute waiting to run because CPU is busy.
- Rt\_CPULoadAverage5m - Average number of processes over the last five minutes waiting to run because CPU is busy.
- Rt\_CPULoadAverage15m - Average number of processes over the last fifteen minutes waiting to run because CPU is busy.
- Rt\_TmpPartUsage - Percentage of the temporary partition used.
- Rt\_LoggingPartUsage - Percentage of the logging partition used. This is applicable only for the SBC2000.

## Appendix B

# Metrics Output

This Appendix includes all produced metrics from two hosts running the exporter on one scrape. The format is in Prometheus metrics, where the first value is the name, the data inside the curly brackets are labels, and the number at the end is its value. The data after each field "HELP", are the descriptions of each metric.



# HELP Rt\_LoggingPartUsage Percentage of the logging partition used. This is applicable only for the SBC2000.

# TYPE Rt\_LoggingPartUsage gauge

Rt\_LoggingPartUsage{chassis\_type="SBC1000",hostip="10.233.230.11",hostname="host2",serial\_number="S4022521210078"} 0

Rt\_LoggingPartUsage{chassis\_type="SBC1000",hostip="10.233.234.11",hostname="host1",serial\_number="S4022521360374"} 0

# HELP Rt\_TmpPartUsage Percentage of the temporary partition used.

# TYPE Rt\_TmpPartUsage gauge

Rt\_TmpPartUsage{chassis\_type="SBC1000",hostip="10.233.230.11",hostname="host2",serial\_number="S4022521210078"} 0

Rt\_TmpPartUsage{chassis\_type="SBC1000",hostip="10.233.234.11",hostname="host1",serial\_number="S4022521360374"} 0

# HELP ifRedundancy ethernetport

# TYPE ifRedundancy gauge

ifRedundancy{ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0

ifRedundancy{ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0

ifRedundancy{ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 2

ifRedundancy{ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 2

ifRedundancy{ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 2

ifRedundancy{ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 2

# HELP ifRedundantPort ethernetport

# TYPE ifRedundantPort gauge

ifRedundantPort{ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 3

ifRedundantPort{ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 3

ifRedundantPort{ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 2

```
ifRedundantPort{ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 2
```

# HELP rt\_ASR Displays the Answer-Seizure Ratio for this call route. (ASR is calculated by dividing the number of call attempts answered by the number of call attempts.)

```
# TYPE rt_ASR gauge
```

```
rt_ASR{hostip="10.233.230.11",hostname="host2",routing_entry="1",routing_table="2"} 0
rt_ASR{hostip="10.233.230.11",hostname="host2",routing_entry="1",routing_table="4"} 100
rt_ASR{hostip="10.233.230.11",hostname="host2",routing_entry="2",routing_table="2"} 0
rt_ASR{hostip="10.233.230.11",hostname="host2",routing_entry="2",routing_table="4"} 0
rt_ASR{hostip="10.233.230.11",hostname="host2",routing_entry="3",routing_table="2"} 0
rt_ASR{hostip="10.233.230.11",hostname="host2",routing_entry="4",routing_table="2"} 0
rt_ASR{hostip="10.233.234.11",hostname="host1",routing_entry="1",routing_table="4"} 0
rt_ASR{hostip="10.233.234.11",hostname="host1",routing_entry="1",routing_table="6"} 100
rt_ASR{hostip="10.233.234.11",hostname="host1",routing_entry="1",routing_table="8"} 80
rt_ASR{hostip="10.233.234.11",hostname="host1",routing_entry="2",routing_table="4"} 0
rt_ASR{hostip="10.233.234.11",hostname="host1",routing_entry="2",routing_table="6"} 0
rt_ASR{hostip="10.233.234.11",hostname="host1",routing_entry="2",routing_table="8"} 0
rt_ASR{hostip="10.233.234.11",hostname="host1",routing_entry="3",routing_table="4"} 0
rt_ASR{hostip="10.233.234.11",hostname="host1",routing_entry="3",routing_table="6"} 0
rt_ASR{hostip="10.233.234.11",hostname="host1",routing_entry="3",routing_table="8"} 0
rt_ASR{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="2"} 62
rt_ASR{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="4"} 0
rt_ASR{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="5"} 90
rt_ASR{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="7"} 90
rt_ASR{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="9"} 0
rt_ASR{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="2"} 0
rt_ASR{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="4"} 0
rt_ASR{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="5"} 0
rt_ASR{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="7"} 0
rt_ASR{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="9"} 0
rt_ASR{hostip="10.233.234.11",hostname="host1",routing_entry="6",routing_table="4"} 0
```



```
# HELP rt_CPULoadAverage15m Average number of processes over the last fifteen minutes waiting to run because CPU is busy.

# TYPE rt_CPULoadAverage15m gauge

rt_CPULoadAverage15m{chassis_type="SBC1000",hostip="10.233.230.11",hostname="host2",serial_number="S4022521210078"} 11

rt_CPULoadAverage15m{chassis_type="SBC1000",hostip="10.233.234.11",hostname="host1",serial_number="S4022521360374"} 11

# HELP rt_CPULoadAverage1m Average number of processes over the last one minute waiting to run because CPU is busy.

# TYPE rt_CPULoadAverage1m gauge

rt_CPULoadAverage1m{chassis_type="SBC1000",hostip="10.233.230.11",hostname="host2",serial_number="S4022521210078"} 12

rt_CPULoadAverage1m{chassis_type="SBC1000",hostip="10.233.234.11",hostname="host1",serial_number="S4022521360374"} 9

# HELP rt_CPULoadAverage5m Average number of processes over the last five minutes waiting to run because CPU is busy.

# TYPE rt_CPULoadAverage5m gauge

rt_CPULoadAverage5m{chassis_type="SBC1000",hostip="10.233.230.11",hostname="host2",serial_number="S4022521210078"} 10

rt_CPULoadAverage5m{chassis_type="SBC1000",hostip="10.233.234.11",hostname="host1",serial_number="S4022521360374"} 10

# HELP rt_CPUPuptime The total duration in seconds, that the system CPU has been UP and running.

# TYPE rt_CPUPuptime gauge

rt_CPUPuptime{chassis_type="SBC1000",hostip="10.233.230.11",hostname="host2",serial_number="S4022521210078"} 7.45501e+06

rt_CPUPuptime{chassis_type="SBC1000",hostip="10.233.234.11",hostname="host1",serial_number="S4022521360374"} 810307

# HELP rt_CPUUsage Average percent usage of the CPU.

# TYPE rt_CPUUsage gauge

rt_CPUUsage{chassis_type="SBC1000",hostip="10.233.230.11",hostname="host2",serial_number="S4022521210078"} 15

rt_CPUUsage{chassis_type="SBC1000",hostip="10.233.234.11",hostname="host1",serial_number="S4022521360374"} 16

# HELP rt_CurrentUsage Amount of memory used by this partition, expressed as percentage

# TYPE rt_CurrentUsage gauge
```

```
rt_CurrentUsage{disk_partition_id="0",disk_partition_name="/dev/root",hostip="10.233.230.11",hostname="host2"} 97

rt_CurrentUsage{disk_partition_id="0",disk_partition_name="/dev/root",hostip="10.233.234.11",hostname="host1"} 97

rt_CurrentUsage{disk_partition_id="1",disk_partition_name="tmpfs",hostip="10.233.230.11",hostname="host2"} 4

rt_CurrentUsage{disk_partition_id="1",disk_partition_name="tmpfs",hostip="10.233.234.11",hostname="host1"} 4

rt_CurrentUsage{disk_partition_id="2",disk_partition_name="none",hostip="10.233.230.11",hostname="host2"} 3

rt_CurrentUsage{disk_partition_id="2",disk_partition_name="none",hostip="10.233.234.11",hostname="host1"} 3

rt_CurrentUsage{disk_partition_id="3",disk_partition_name="/dev/mtdblock5",hostip="10.233.230.11",hostname="host2"} 10

rt_CurrentUsage{disk_partition_id="3",disk_partition_name="/dev/mtdblock5",hostip="10.233.234.11",hostname="host1"} 10

rt_CurrentUsage{disk_partition_id="4",disk_partition_name="none",hostip="10.233.230.11",hostname="host2"} 0

rt_CurrentUsage{disk_partition_id="4",disk_partition_name="none",hostip="10.233.234.11",hostname="host1"} 0

rt_CurrentUsage{disk_partition_id="5",disk_partition_name="none",hostip="10.233.230.11",hostname="host2"} 17

rt_CurrentUsage{disk_partition_id="5",disk_partition_name="none",hostip="10.233.234.11",hostname="host1"} 26

rt_CurrentUsage{disk_partition_id="6",disk_partition_name="/dev/eusb2",hostip="10.233.230.11",hostname="host2"} 4

rt_CurrentUsage{disk_partition_id="6",disk_partition_name="/dev/eusb2",hostip="10.233.234.11",hostname="host1"} 4

rt_CurrentUsage{disk_partition_id="7",disk_partition_name="/dev/eusb3",hostip="10.233.230.11",hostname="host2"} 41

rt_CurrentUsage{disk_partition_id="7",disk_partition_name="/dev/eusb3",hostip="10.233.234.11",hostname="host1"} 44

# HELP rt_FDUsage Number of file descriptors used by the system.

# TYPE rt_FDUsage gauge

rt_FDUsage{chassis_type="SBC1000",hostip="10.233.230.11",hostname="host2",serial_number="S4022521210078"} 1420
```

```
rt_FDUsage{chassis_type="SBC1000",hostip="10.233.234.11",hostname="host1",serial_number="S402  
2521360374"} 1479
```

```
# HELP rt_Jitter Displays the average jitter for this call route.
```

```
# TYPE rt_Jitter gauge
```

```
rt_Jitter{hostip="10.233.230.11",hostname="host2",routing_entry="1",routing_table="2"} 0  
rt_Jitter{hostip="10.233.230.11",hostname="host2",routing_entry="1",routing_table="4"} 0  
rt_Jitter{hostip="10.233.230.11",hostname="host2",routing_entry="2",routing_table="2"} 0  
rt_Jitter{hostip="10.233.230.11",hostname="host2",routing_entry="2",routing_table="4"} 3000  
rt_Jitter{hostip="10.233.230.11",hostname="host2",routing_entry="3",routing_table="2"} 3000  
rt_Jitter{hostip="10.233.230.11",hostname="host2",routing_entry="4",routing_table="2"} 3000  
rt_Jitter{hostip="10.233.234.11",hostname="host1",routing_entry="1",routing_table="4"} 0  
rt_Jitter{hostip="10.233.234.11",hostname="host1",routing_entry="1",routing_table="6"} 0  
rt_Jitter{hostip="10.233.234.11",hostname="host1",routing_entry="1",routing_table="8"} 0  
rt_Jitter{hostip="10.233.234.11",hostname="host1",routing_entry="2",routing_table="4"} 0  
rt_Jitter{hostip="10.233.234.11",hostname="host1",routing_entry="2",routing_table="6"} 3000  
rt_Jitter{hostip="10.233.234.11",hostname="host1",routing_entry="2",routing_table="8"} 3000  
rt_Jitter{hostip="10.233.234.11",hostname="host1",routing_entry="3",routing_table="4"} 3000  
rt_Jitter{hostip="10.233.234.11",hostname="host1",routing_entry="3",routing_table="6"} 3000  
rt_Jitter{hostip="10.233.234.11",hostname="host1",routing_entry="3",routing_table="8"} 3000  
rt_Jitter{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="2"} 0  
rt_Jitter{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="4"} 3000  
rt_Jitter{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="5"} 0  
rt_Jitter{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="7"} 0  
rt_Jitter{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="9"} 3000  
rt_Jitter{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="2"} 3000  
rt_Jitter{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="4"} 3000  
rt_Jitter{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="5"} 3000  
rt_Jitter{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="7"} 0  
rt_Jitter{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="9"} 3000  
rt_Jitter{hostip="10.233.234.11",hostname="host1",routing_entry="6",routing_table="4"} 3000
```

```
# HELP rt_MOS Displays the Mean Opinion Score (MOS) for this call route.
# TYPE rt_MOS gauge
rt_MOS{hostip="10.233.230.11",hostname="host2",routing_entry="1",routing_table="2"} 50
rt_MOS{hostip="10.233.230.11",hostname="host2",routing_entry="1",routing_table="4"} 41
rt_MOS{hostip="10.233.230.11",hostname="host2",routing_entry="2",routing_table="2"} 50
rt_MOS{hostip="10.233.230.11",hostname="host2",routing_entry="2",routing_table="4"} 0
rt_MOS{hostip="10.233.230.11",hostname="host2",routing_entry="3",routing_table="2"} 0
rt_MOS{hostip="10.233.230.11",hostname="host2",routing_entry="4",routing_table="2"} 0
rt_MOS{hostip="10.233.234.11",hostname="host1",routing_entry="1",routing_table="4"} 50
rt_MOS{hostip="10.233.234.11",hostname="host1",routing_entry="1",routing_table="6"} 50
rt_MOS{hostip="10.233.234.11",hostname="host1",routing_entry="1",routing_table="8"} 50
rt_MOS{hostip="10.233.234.11",hostname="host1",routing_entry="2",routing_table="4"} 50
rt_MOS{hostip="10.233.234.11",hostname="host1",routing_entry="2",routing_table="6"} 0
rt_MOS{hostip="10.233.234.11",hostname="host1",routing_entry="2",routing_table="8"} 0
rt_MOS{hostip="10.233.234.11",hostname="host1",routing_entry="3",routing_table="4"} 0
rt_MOS{hostip="10.233.234.11",hostname="host1",routing_entry="3",routing_table="6"} 0
rt_MOS{hostip="10.233.234.11",hostname="host1",routing_entry="3",routing_table="8"} 0
rt_MOS{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="2"} 50
rt_MOS{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="4"} 0
rt_MOS{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="5"} 50
rt_MOS{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="7"} 50
rt_MOS{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="9"} 0
rt_MOS{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="2"} 0
rt_MOS{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="4"} 0
rt_MOS{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="5"} 0
rt_MOS{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="7"} 50
rt_MOS{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="9"} 0
rt_MOS{hostip="10.233.234.11",hostname="host1",routing_entry="6",routing_table="4"} 0
# HELP rt_MaximumSize Specifies the maximum amount of memory, in bytes available in this partition.
# TYPE rt_MaximumSize gauge
```

rt\_MaximumSize{disk\_partition\_id="0",disk\_partition\_name="/dev/root",hostip="10.233.230.11",hostname="host2"} 3.0408704e+07

rt\_MaximumSize{disk\_partition\_id="0",disk\_partition\_name="/dev/root",hostip="10.233.234.11",hostname="host1"} 3.0408704e+07

rt\_MaximumSize{disk\_partition\_id="1",disk\_partition\_name="tmpfs",hostip="10.233.230.11",hostname="host2"} 524288

rt\_MaximumSize{disk\_partition\_id="1",disk\_partition\_name="tmpfs",hostip="10.233.234.11",hostname="host1"} 524288

rt\_MaximumSize{disk\_partition\_id="2",disk\_partition\_name="none",hostip="10.233.230.11",hostname="host2"} 1.2582912e+07

rt\_MaximumSize{disk\_partition\_id="2",disk\_partition\_name="none",hostip="10.233.234.11",hostname="host1"} 1.2582912e+07

rt\_MaximumSize{disk\_partition\_id="3",disk\_partition\_name="/dev/mtdblock5",hostip="10.233.230.11",hostname="host2"} 5.767168e+06

rt\_MaximumSize{disk\_partition\_id="3",disk\_partition\_name="/dev/mtdblock5",hostip="10.233.234.11",hostname="host1"} 5.767168e+06

rt\_MaximumSize{disk\_partition\_id="4",disk\_partition\_name="none",hostip="10.233.230.11",hostname="host2"} 3.3554432e+07

rt\_MaximumSize{disk\_partition\_id="4",disk\_partition\_name="none",hostip="10.233.234.11",hostname="host1"} 3.3554432e+07

rt\_MaximumSize{disk\_partition\_id="5",disk\_partition\_name="none",hostip="10.233.230.11",hostname="host2"} 1.048576e+06

rt\_MaximumSize{disk\_partition\_id="5",disk\_partition\_name="none",hostip="10.233.234.11",hostname="host1"} 1.048576e+06

rt\_MaximumSize{disk\_partition\_id="6",disk\_partition\_name="/dev/eusb2",hostip="10.233.230.11",hostname="host2"} 3.11809024e+08

rt\_MaximumSize{disk\_partition\_id="6",disk\_partition\_name="/dev/eusb2",hostip="10.233.234.11",hostname="host1"} 3.11809024e+08

rt\_MaximumSize{disk\_partition\_id="7",disk\_partition\_name="/dev/eusb3",hostip="10.233.230.11",hostname="host2"} 1.472360448e+09

rt\_MaximumSize{disk\_partition\_id="7",disk\_partition\_name="/dev/eusb3",hostip="10.233.234.11",hostname="host1"} 1.472360448e+09

# HELP rt\_MemoryAvailable Amount of memory in bytes, available for use in the filesystem.

# TYPE rt\_MemoryAvailable gauge

rt\_MemoryAvailable{disk\_partition\_id="0",disk\_partition\_name="/dev/root",hostip="10.233.230.11",hostname="host2"} 839680

rt\_MemoryAvailable{disk\_partition\_id="0",disk\_partition\_name="/dev/root",hostip="10.233.234.11",hostname="host1"} 811008

rt\_MemoryAvailable{disk\_partition\_id="1",disk\_partition\_name="tmpfs",hostip="10.233.230.11",hostname="host2"} 503808

rt\_MemoryAvailable{disk\_partition\_id="1",disk\_partition\_name="tmpfs",hostip="10.233.234.11",hostname="host1"} 503808

rt\_MemoryAvailable{disk\_partition\_id="2",disk\_partition\_name="none",hostip="10.233.230.11",hostname="host2"} 1.2230656e+07

rt\_MemoryAvailable{disk\_partition\_id="2",disk\_partition\_name="none",hostip="10.233.234.11",hostname="host1"} 1.222656e+07

rt\_MemoryAvailable{disk\_partition\_id="3",disk\_partition\_name="/dev/mtdblock5",hostip="10.233.230.11",hostname="host2"} 5.193728e+06

rt\_MemoryAvailable{disk\_partition\_id="3",disk\_partition\_name="/dev/mtdblock5",hostip="10.233.234.11",hostname="host1"} 5.173248e+06

rt\_MemoryAvailable{disk\_partition\_id="4",disk\_partition\_name="none",hostip="10.233.230.11",hostname="host2"} 3.3554432e+07

rt\_MemoryAvailable{disk\_partition\_id="4",disk\_partition\_name="none",hostip="10.233.234.11",hostname="host1"} 3.3554432e+07

rt\_MemoryAvailable{disk\_partition\_id="5",disk\_partition\_name="none",hostip="10.233.230.11",hostname="host2"} 872448

rt\_MemoryAvailable{disk\_partition\_id="5",disk\_partition\_name="none",hostip="10.233.234.11",hostname="host1"} 774144

rt\_MemoryAvailable{disk\_partition\_id="6",disk\_partition\_name="/dev/eusb2",hostip="10.233.230.11",hostname="host2"} 2.85174784e+08

rt\_MemoryAvailable{disk\_partition\_id="6",disk\_partition\_name="/dev/eusb2",hostip="10.233.234.11",hostname="host1"} 2.85174784e+08

rt\_MemoryAvailable{disk\_partition\_id="7",disk\_partition\_name="/dev/eusb3",hostip="10.233.230.11",hostname="host2"} 8.2307072e+08

rt\_MemoryAvailable{disk\_partition\_id="7",disk\_partition\_name="/dev/eusb3",hostip="10.233.234.11",hostname="host1"} 7.83679488e+08

# HELP rt\_MemoryUsage Average percent usage of system memory.

# TYPE rt\_MemoryUsage gauge

rt\_MemoryUsage{chassis\_type="SBC1000",hostip="10.233.230.11",hostname="host2",serial\_number="S4022521210078"} 38

rt\_MemoryUsage{chassis\_type="SBC1000",hostip="10.233.234.11",hostname="host1",serial\_number="S4022521360374"} 39

```
# HELP rt_MemoryUsed Amount of memory in bytes, used by the existing files in the filesystem
# TYPE rt_MemoryUsed gauge

rt_MemoryUsed{disk_partition_id="0",disk_partition_name="/dev/root",hostip="10.233.230.11",hostname="host2"} 2.9569024e+07

rt_MemoryUsed{disk_partition_id="0",disk_partition_name="/dev/root",hostip="10.233.234.11",hostname="host1"} 2.9597696e+07

rt_MemoryUsed{disk_partition_id="1",disk_partition_name="tmpfs",hostip="10.233.230.11",hostname="host2"} 20480

rt_MemoryUsed{disk_partition_id="1",disk_partition_name="tmpfs",hostip="10.233.234.11",hostname="host1"} 20480

rt_MemoryUsed{disk_partition_id="2",disk_partition_name="none",hostip="10.233.230.11",hostname="host2"} 352256

rt_MemoryUsed{disk_partition_id="2",disk_partition_name="none",hostip="10.233.234.11",hostname="host1"} 356352

rt_MemoryUsed{disk_partition_id="3",disk_partition_name="/dev/mtdblock5",hostip="10.233.230.11",hostname="host2"} 573440

rt_MemoryUsed{disk_partition_id="3",disk_partition_name="/dev/mtdblock5",hostip="10.233.234.11",hostname="host1"} 593920

rt_MemoryUsed{disk_partition_id="4",disk_partition_name="none",hostip="10.233.230.11",hostname="host2"} 0

rt_MemoryUsed{disk_partition_id="4",disk_partition_name="none",hostip="10.233.234.11",hostname="host1"} 0

rt_MemoryUsed{disk_partition_id="5",disk_partition_name="none",hostip="10.233.230.11",hostname="host2"} 176128

rt_MemoryUsed{disk_partition_id="5",disk_partition_name="none",hostip="10.233.234.11",hostname="host1"} 274432

rt_MemoryUsed{disk_partition_id="6",disk_partition_name="/dev/eusb2",hostip="10.233.230.11",hostname="host2"} 1.0535936e+07

rt_MemoryUsed{disk_partition_id="6",disk_partition_name="/dev/eusb2",hostip="10.233.234.11",hostname="host1"} 1.0535936e+07

rt_MemoryUsed{disk_partition_id="7",disk_partition_name="/dev/eusb3",hostip="10.233.230.11",hostname="host2"} 5.74496768e+08

rt_MemoryUsed{disk_partition_id="7",disk_partition_name="/dev/eusb3",hostip="10.233.234.11",hostname="host1"} 6.13888e+08

# HELP rt_NumCallAbandonedNoTrunk Number of rejected calls due to no channel available system wide since system came up.
```

```
# TYPE rt_NumCallAbandonedNoTrunk gauge
rt_NumCallAbandonedNoTrunk{hostip="10.233.230.11",hostname="host2"} 0
rt_NumCallAbandonedNoTrunk{hostip="10.233.234.11",hostname="host1"} 0
# HELP rt_NumCallAttempts Total number of call attempts system wide since system came up.
# TYPE rt_NumCallAttempts gauge
rt_NumCallAttempts{hostip="10.233.230.11",hostname="host2"} 17
rt_NumCallAttempts{hostip="10.233.234.11",hostname="host1"} 175
# HELP rt_NumCallCurrentlyUp Number of currently connected calls system wide.
# TYPE rt_NumCallCurrentlyUp gauge
rt_NumCallCurrentlyUp{hostip="10.233.230.11",hostname="host2"} 0
rt_NumCallCurrentlyUp{hostip="10.233.234.11",hostname="host1"} 0
# HELP rt_NumCallFailed Total number of failed calls system wide since system came up.
# TYPE rt_NumCallFailed gauge
rt_NumCallFailed{hostip="10.233.230.11",hostname="host2"} 0
rt_NumCallFailed{hostip="10.233.234.11",hostname="host1"} 3
# HELP rt_NumCallSucceeded Total number of successful calls system wide since system came up.
# TYPE rt_NumCallSucceeded gauge
rt_NumCallSucceeded{hostip="10.233.230.11",hostname="host2"} 13
rt_NumCallSucceeded{hostip="10.233.234.11",hostname="host1"} 84
# HELP rt_NumCallUnAnswered Number of unanswered calls system wide since system came up.
# TYPE rt_NumCallUnAnswered gauge
rt_NumCallUnAnswered{hostip="10.233.230.11",hostname="host2"} 14
rt_NumCallUnAnswered{hostip="10.233.234.11",hostname="host1"} 88
# HELP rt_PartitionType Identifies the user-friendly physical device holding the partition.
# TYPE rt_PartitionType gauge
rt_PartitionType{disk_partition_id="0",disk_partition_name="/dev/root",hostip="10.233.230.11",hostname="host2"} 8
rt_PartitionType{disk_partition_id="0",disk_partition_name="/dev/root",hostip="10.233.234.11",hostname="host1"} 8
rt_PartitionType{disk_partition_id="1",disk_partition_name="tmpfs",hostip="10.233.230.11",hostname="host2"} 7
```



```
rt_PartitionType{disk_partition_id="1",disk_partition_name="tmpfs",hostip="10.233.234.11",hostname="host1"} 7
```

```
rt_PartitionType{disk_partition_id="2",disk_partition_name="none",hostip="10.233.230.11",hostname="host2"} 2
```

```
rt_PartitionType{disk_partition_id="2",disk_partition_name="none",hostip="10.233.234.11",hostname="host1"} 2
```

```
rt_PartitionType{disk_partition_id="3",disk_partition_name="/dev/mtdblock5",hostip="10.233.230.11",hostname="host2"} 0
```

```
rt_PartitionType{disk_partition_id="3",disk_partition_name="/dev/mtdblock5",hostip="10.233.234.11",hostname="host1"} 0
```

```
rt_PartitionType{disk_partition_id="4",disk_partition_name="none",hostip="10.233.230.11",hostname="host2"} 5
```

```
rt_PartitionType{disk_partition_id="4",disk_partition_name="none",hostip="10.233.234.11",hostname="host1"} 5
```

```
rt_PartitionType{disk_partition_id="5",disk_partition_name="none",hostip="10.233.230.11",hostname="host2"} 6
```

```
rt_PartitionType{disk_partition_id="5",disk_partition_name="none",hostip="10.233.234.11",hostname="host1"} 6
```

```
rt_PartitionType{disk_partition_id="6",disk_partition_name="/dev/eusb2",hostip="10.233.230.11",hostname="host2"} 3
```

```
rt_PartitionType{disk_partition_id="6",disk_partition_name="/dev/eusb2",hostip="10.233.234.11",hostname="host1"} 3
```

```
rt_PartitionType{disk_partition_id="7",disk_partition_name="/dev/eusb3",hostip="10.233.230.11",hostname="host2"} 1
```

```
rt_PartitionType{disk_partition_id="7",disk_partition_name="/dev/eusb3",hostip="10.233.234.11",hostname="host1"} 1
```

# HELP rt\_QualityFailed Displays if this call route is currently passing or failing the associated quality metrics. If true then the rule is failing, if false then it is passing.

# TYPE rt\_QualityFailed gauge

```
rt_QualityFailed{hostip="10.233.230.11",hostname="host2",routing_entry="1",routing_table="2"} 0
```

```
rt_QualityFailed{hostip="10.233.230.11",hostname="host2",routing_entry="1",routing_table="4"} 0
```

```
rt_QualityFailed{hostip="10.233.230.11",hostname="host2",routing_entry="2",routing_table="2"} 0
```

```
rt_QualityFailed{hostip="10.233.230.11",hostname="host2",routing_entry="2",routing_table="4"} 0
```

```
rt_QualityFailed{hostip="10.233.230.11",hostname="host2",routing_entry="3",routing_table="2"} 0
```

```
rt_QualityFailed{hostip="10.233.230.11",hostname="host2",routing_entry="4",routing_table="2"} 0
```

```
rt_QualityFailed{hostip="10.233.234.11",hostname="host1",routing_entry="1",routing_table="4"} 0
rt_QualityFailed{hostip="10.233.234.11",hostname="host1",routing_entry="1",routing_table="6"} 0
rt_QualityFailed{hostip="10.233.234.11",hostname="host1",routing_entry="1",routing_table="8"} 0
rt_QualityFailed{hostip="10.233.234.11",hostname="host1",routing_entry="2",routing_table="4"} 0
rt_QualityFailed{hostip="10.233.234.11",hostname="host1",routing_entry="2",routing_table="6"} 0
rt_QualityFailed{hostip="10.233.234.11",hostname="host1",routing_entry="2",routing_table="8"} 0
rt_QualityFailed{hostip="10.233.234.11",hostname="host1",routing_entry="3",routing_table="4"} 0
rt_QualityFailed{hostip="10.233.234.11",hostname="host1",routing_entry="3",routing_table="6"} 0
rt_QualityFailed{hostip="10.233.234.11",hostname="host1",routing_entry="3",routing_table="8"} 0
rt_QualityFailed{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="2"} 0
rt_QualityFailed{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="4"} 0
rt_QualityFailed{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="5"} 0
rt_QualityFailed{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="7"} 0
rt_QualityFailed{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="9"} 0
rt_QualityFailed{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="2"} 0
rt_QualityFailed{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="4"} 0
rt_QualityFailed{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="5"} 0
rt_QualityFailed{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="7"} 0
rt_QualityFailed{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="9"} 0
rt_QualityFailed{hostip="10.233.234.11",hostname="host1",routing_entry="6",routing_table="4"} 0
# HELP rt_RoundTripDelay Displays the average round trip delay for this call route.
# TYPE rt_RoundTripDelay gauge
rt_RoundTripDelay{hostip="10.233.230.11",hostname="host2",routing_entry="1",routing_table="2"} 0
rt_RoundTripDelay{hostip="10.233.230.11",hostname="host2",routing_entry="1",routing_table="4"} 0
rt_RoundTripDelay{hostip="10.233.230.11",hostname="host2",routing_entry="2",routing_table="2"} 0
rt_RoundTripDelay{hostip="10.233.230.11",hostname="host2",routing_entry="2",routing_table="4"}
9999
rt_RoundTripDelay{hostip="10.233.230.11",hostname="host2",routing_entry="3",routing_table="2"}
9999
rt_RoundTripDelay{hostip="10.233.230.11",hostname="host2",routing_entry="4",routing_table="2"}
9999
```

```
rt_RoundTripDelay{hostip="10.233.234.11",hostname="host1",routing_entry="1",routing_table="4"} 0
rt_RoundTripDelay{hostip="10.233.234.11",hostname="host1",routing_entry="1",routing_table="6"} 0
rt_RoundTripDelay{hostip="10.233.234.11",hostname="host1",routing_entry="1",routing_table="8"} 0
rt_RoundTripDelay{hostip="10.233.234.11",hostname="host1",routing_entry="2",routing_table="4"} 0
rt_RoundTripDelay{hostip="10.233.234.11",hostname="host1",routing_entry="2",routing_table="6"}
9999
rt_RoundTripDelay{hostip="10.233.234.11",hostname="host1",routing_entry="2",routing_table="8"}
9999
rt_RoundTripDelay{hostip="10.233.234.11",hostname="host1",routing_entry="3",routing_table="4"}
9999
rt_RoundTripDelay{hostip="10.233.234.11",hostname="host1",routing_entry="3",routing_table="6"}
9999
rt_RoundTripDelay{hostip="10.233.234.11",hostname="host1",routing_entry="3",routing_table="8"}
9999
rt_RoundTripDelay{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="2"} 0
rt_RoundTripDelay{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="4"}
9999
rt_RoundTripDelay{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="5"} 0
rt_RoundTripDelay{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="7"} 0
rt_RoundTripDelay{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="9"}
9999
rt_RoundTripDelay{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="2"}
9999
rt_RoundTripDelay{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="4"}
9999
rt_RoundTripDelay{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="5"}
9999
rt_RoundTripDelay{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="7"} 0
rt_RoundTripDelay{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="9"}
9999
rt_RoundTripDelay{hostip="10.233.234.11",hostname="host1",routing_entry="6",routing_table="4"}
9999
# HELP rt_RuleUsage Displays the number of times this call route has been selected for a call.
# TYPE rt_RuleUsage gauge
```

```
rt_RuleUsage{hostip="10.233.230.11",hostname="host2",routing_entry="1",routing_table="2"} 0
rt_RuleUsage{hostip="10.233.230.11",hostname="host2",routing_entry="1",routing_table="4"} 0
rt_RuleUsage{hostip="10.233.230.11",hostname="host2",routing_entry="2",routing_table="2"} 0
rt_RuleUsage{hostip="10.233.230.11",hostname="host2",routing_entry="2",routing_table="4"} 0
rt_RuleUsage{hostip="10.233.230.11",hostname="host2",routing_entry="3",routing_table="2"} 0
rt_RuleUsage{hostip="10.233.230.11",hostname="host2",routing_entry="4",routing_table="2"} 0
rt_RuleUsage{hostip="10.233.234.11",hostname="host1",routing_entry="1",routing_table="4"} 0
rt_RuleUsage{hostip="10.233.234.11",hostname="host1",routing_entry="1",routing_table="6"} 0
rt_RuleUsage{hostip="10.233.234.11",hostname="host1",routing_entry="1",routing_table="8"} 0
rt_RuleUsage{hostip="10.233.234.11",hostname="host1",routing_entry="2",routing_table="4"} 0
rt_RuleUsage{hostip="10.233.234.11",hostname="host1",routing_entry="2",routing_table="6"} 0
rt_RuleUsage{hostip="10.233.234.11",hostname="host1",routing_entry="2",routing_table="8"} 0
rt_RuleUsage{hostip="10.233.234.11",hostname="host1",routing_entry="3",routing_table="4"} 0
rt_RuleUsage{hostip="10.233.234.11",hostname="host1",routing_entry="3",routing_table="6"} 0
rt_RuleUsage{hostip="10.233.234.11",hostname="host1",routing_entry="3",routing_table="8"} 0
rt_RuleUsage{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="2"} 0
rt_RuleUsage{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="4"} 0
rt_RuleUsage{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="5"} 0
rt_RuleUsage{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="7"} 0
rt_RuleUsage{hostip="10.233.234.11",hostname="host1",routing_entry="4",routing_table="9"} 0
rt_RuleUsage{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="2"} 0
rt_RuleUsage{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="4"} 0
rt_RuleUsage{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="5"} 0
rt_RuleUsage{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="7"} 0
rt_RuleUsage{hostip="10.233.234.11",hostname="host1",routing_entry="5",routing_table="9"} 0
rt_RuleUsage{hostip="10.233.234.11",hostname="host1",routing_entry="6",routing_table="4"} 0
# HELP rt_ServiceStatus The service status of the module.
# TYPE rt_ServiceStatus gauge
rt_ServiceStatus{hostip="10.233.230.11",hostname="host2",job="linecard",linecardID="7",rt_CardType="26",rt_Location="21"} 1
```

```

rt_ServiceStatus{hostip="10.233.230.11",hostname="host2",job="linecard",linecardID="8",rt_CardType="26",rt_Location="22"} 1

rt_ServiceStatus{hostip="10.233.234.11",hostname="host1",job="linecard",linecardID="7",rt_CardType="26",rt_Location="21"} 1

rt_ServiceStatus{hostip="10.233.234.11",hostname="host1",job="linecard",linecardID="8",rt_CardType="26",rt_Location="22"} 1

# HELP rt_Status Indicates the hardware initialization state for this card.

# TYPE rt_Status gauge

rt_Status{hostip="10.233.230.11",hostname="host2",job="linecard",linecardID="7"} 4

rt_Status{hostip="10.233.230.11",hostname="host2",job="linecard",linecardID="8"} 4

rt_Status{hostip="10.233.234.11",hostname="host1",job="linecard",linecardID="7"} 4

rt_Status{hostip="10.233.234.11",hostname="host1",job="linecard",linecardID="8"} 4

# HELP rt_ifInBroadcastPkts Displays the number of received broadcast packets on this port.

# TYPE rt_ifInBroadcastPkts gauge

rt_ifInBroadcastPkts{ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 249389

rt_ifInBroadcastPkts{ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 13546

rt_ifInBroadcastPkts{ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 249034

rt_ifInBroadcastPkts{ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0

rt_ifInBroadcastPkts{ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 3

rt_ifInBroadcastPkts{ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 0

# HELP rt_ifInDiscards Displays the number of discard errors detected on this port.

# TYPE rt_ifInDiscards gauge

rt_ifInDiscards{ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0

rt_ifInDiscards{ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0

rt_ifInDiscards{ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0

```

```
rt_ifInDiscards {ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0
```

```
rt_ifInDiscards {ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 1
```

```
rt_ifInDiscards {ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 6
```

# HELP rt\_ifInErrors Displays the number of errors detected on this port.

# TYPE rt\_ifInErrors gauge

```
rt_ifInErrors {ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0
```

```
rt_ifInErrors {ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0
```

```
rt_ifInErrors {ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0
```

```
rt_ifInErrors {ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0
```

```
rt_ifInErrors {ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 1
```

```
rt_ifInErrors {ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 6
```

# HELP rt\_ifInFCSErrors Displays the number of discard Frame Check Sequence errors detected on this port.

# TYPE rt\_ifInFCSErrors gauge

```
rt_ifInFCSErrors {ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0
```

```
rt_ifInFCSErrors {ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0
```

```
rt_ifInFCSErrors {ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0
```

```
rt_ifInFCSErrors {ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0
```

```
rt_ifInFCSErrors {ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 0
```

```
rt_ifInFCSErrors {ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 0
```

# HELP rt\_ifInFragmentedPkts Displays the number of Fragmented Packet errors detected on this port

# TYPE rt\_ifInFragmentedPkts gauge

rt\_ifInFragmentedPkts {ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0

rt\_ifInFragmentedPkts {ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0

rt\_ifInFragmentedPkts {ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0

rt\_ifInFragmentedPkts {ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0

rt\_ifInFragmentedPkts {ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 0

rt\_ifInFragmentedPkts {ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 3

# HELP rt\_ifInMulticastPkts Displays the number of received multicast packets on this port.

# TYPE rt\_ifInMulticastPkts gauge

rt\_ifInMulticastPkts {ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 3.570987e+06

rt\_ifInMulticastPkts {ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 1.198147e+06

rt\_ifInMulticastPkts {ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 3.959414e+06

rt\_ifInMulticastPkts {ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 1.649113e+06

rt\_ifInMulticastPkts {ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 2

rt\_ifInMulticastPkts {ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 11

# HELP rt\_ifInOctets Displays the number of received octets on this port.

# TYPE rt\_ifInOctets gauge

rt\_ifInOctets {ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 1.431542685e+09

rt\_ifInOctets {ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 4.4678806e+08

rt\_ifInOctets {ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 1.327937704e+09

```
rt_ifInOctets {ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 4.81174993e+08

rt_ifInOctets {ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 320

rt_ifInOctets {ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 776

# HELP rt_ifInOverSizedPkts Displays the number of Oversized Packet errors detected on this port.

# TYPE rt_ifInOverSizedPkts gauge

rt_ifInOverSizedPkts {ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0

rt_ifInOverSizedPkts {ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0

rt_ifInOverSizedPkts {ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0

rt_ifInOverSizedPkts {ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0

rt_ifInOverSizedPkts {ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 0

rt_ifInOverSizedPkts {ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 0

# HELP rt_ifInUcastPkts Displays the number of received unicast packets on this port.

# TYPE rt_ifInUcastPkts gauge

rt_ifInUcastPkts {ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 1.1375715e+07

rt_ifInUcastPkts {ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 3.911751e+06

rt_ifInUcastPkts {ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 6.7576e+06

rt_ifInUcastPkts {ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 2.722781e+06

rt_ifInUcastPkts {ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 5

rt_ifInUcastPkts {ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 11

# HELP rt_ifInUndersizedPkts Displays the number of Undersized Packet errors detected on this port.
```



```
# TYPE rt_ifInUndersizedPkts gauge

rt_ifInUndersizedPkts {ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0

rt_ifInUndersizedPkts {ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0

rt_ifInUndersizedPkts {ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0

rt_ifInUndersizedPkts {ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0

rt_ifInUndersizedPkts {ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 0

rt_ifInUndersizedPkts {ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 1

# HELP rt_ifInUnknwnProto Displays the number of Unknown Protocol errors detected on this port.

# TYPE rt_ifInUnknwnProto gauge

rt_ifInUnknwnProto {ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0

rt_ifInUnknwnProto {ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0

rt_ifInUnknwnProto {ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0

rt_ifInUnknwnProto {ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0

rt_ifInUnknwnProto {ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 0

rt_ifInUnknwnProto {ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 0

# HELP rt_ifInterfaceIndex ethernetport

# TYPE rt_ifInterfaceIndex gauge

rt_ifInterfaceIndex {ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 11

rt_ifInterfaceIndex {ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 11

rt_ifInterfaceIndex {ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 12
```

```
rt_ifInterfaceIndex {ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 12

rt_ifInterfaceIndex {ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 15

rt_ifInterfaceIndex {ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 15

# HELP rt_ifLastChange The value of sysUpTime at the time the interface entered its current operational state.

# TYPE rt_ifLastChange gauge

rt_ifLastChange {ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0

rt_ifLastChange {ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0

rt_ifLastChange {ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0

rt_ifLastChange {ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0

rt_ifLastChange {ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 0

rt_ifLastChange {ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 0

# HELP rt_ifMtu The size of the largest packet which can be sent/received on the interface.

# TYPE rt_ifMtu gauge

rt_ifMtu {ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 1500

rt_ifMtu {ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 1500

rt_ifMtu {ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 1500

rt_ifMtu {ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 1500

rt_ifMtu {ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 1500

rt_ifMtu {ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 1500
```

```

# HELP rt_ifOperatorStatus The operational status of the interface - 0 = IF_OPER_UP or 1 =
IF_OPER_DOWN.

# TYPE rt_ifOperatorStatus gauge

rt_ifOperatorStatus{ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",if
Name="Ethernet 1",job="ethernetport"} 0

rt_ifOperatorStatus{ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",if
Name="Ethernet 1",job="ethernetport"} 0

rt_ifOperatorStatus{ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",i
fName="Ethernet 2",job="ethernetport"} 0

rt_ifOperatorStatus{ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",i
fName="Ethernet 2",job="ethernetport"} 0

rt_ifOperatorStatus{ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",i
fName="Ethernet 3",job="ethernetport"} 0

rt_ifOperatorStatus{ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",i
fName="Ethernet 3",job="ethernetport"} 0

# HELP rt_ifOutBroadcastPkts Displays the number of transmitted broadcast packets on this port.

# TYPE rt_ifOutBroadcastPkts gauge

rt_ifOutBroadcastPkts{ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT
",ifName="Ethernet 1",job="ethernetport"} 124257

rt_ifOutBroadcastPkts{ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT
",ifName="Ethernet 1",job="ethernetport"} 13505

rt_ifOutBroadcastPkts{ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA
0",ifName="Ethernet 2",job="ethernetport"} 124441

rt_ifOutBroadcastPkts{ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA
0",ifName="Ethernet 2",job="ethernetport"} 13505

rt_ifOutBroadcastPkts{ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA
1",ifName="Ethernet 3",job="ethernetport"} 43

rt_ifOutBroadcastPkts{ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA
1",ifName="Ethernet 3",job="ethernetport"} 42

# HELP rt_ifOutDeferredTransmissions Displays the number of Deferred Transmission errors detected on
this port.

# TYPE rt_ifOutDeferredTransmissions gauge

rt_ifOutDeferredTransmissions{ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias=
"MGMT",ifName="Ethernet 1",job="ethernetport"} 0

rt_ifOutDeferredTransmissions{ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias=
"MGMT",ifName="Ethernet 1",job="ethernetport"} 0

```

```
rt_ifOutDeferredTransmissions {ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0
```

```
rt_ifOutDeferredTransmissions {ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0
```

```
rt_ifOutDeferredTransmissions {ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 0
```

```
rt_ifOutDeferredTransmissions {ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 0
```

# HELP rt\_ifOutDiscards Displays the number of discard errors detected on this port.

# TYPE rt\_ifOutDiscards gauge

```
rt_ifOutDiscards {ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0
```

```
rt_ifOutDiscards {ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0
```

```
rt_ifOutDiscards {ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0
```

```
rt_ifOutDiscards {ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0
```

```
rt_ifOutDiscards {ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 0
```

```
rt_ifOutDiscards {ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 0
```

# HELP rt\_ifOutErrors Displays the number of errors detected on this port.

# TYPE rt\_ifOutErrors gauge

```
rt_ifOutErrors {ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0
```

```
rt_ifOutErrors {ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0
```

```
rt_ifOutErrors {ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0
```

```
rt_ifOutErrors {ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0
```

```
rt_ifOutErrors {ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 0
```

```
rt_ifOutErrors {ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 0
```

```
# HELP rt_ifOutLateCollissions Displays the number of Late Collision errors detected on this port.
# TYPE rt_ifOutLateCollissions gauge
rt_ifOutLateCollissions{ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0
rt_ifOutLateCollissions{ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0
rt_ifOutLateCollissions{ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0
rt_ifOutLateCollissions{ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0
rt_ifOutLateCollissions{ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 0
rt_ifOutLateCollissions{ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 0
# HELP rt_ifOutMulticastPkts Displays the number of transmitted multicast packets on this port.
# TYPE rt_ifOutMulticastPkts gauge
rt_ifOutMulticastPkts{ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0
rt_ifOutMulticastPkts{ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 0
rt_ifOutMulticastPkts{ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0
rt_ifOutMulticastPkts{ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0
rt_ifOutMulticastPkts{ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 28
rt_ifOutMulticastPkts{ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 82
# HELP rt_ifOutOctets Displays the number of transmitted octets on this port.
# TYPE rt_ifOutOctets gauge
rt_ifOutOctets{ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 2.17152525e+09
rt_ifOutOctets{ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 8.19801378e+08
```

```
rt_ifOutOctets {ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 9.78217974e+08
```

```
rt_ifOutOctets {ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 3.91589683e+08
```

```
rt_ifOutOctets {ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 28027
```

```
rt_ifOutOctets {ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 17412
```

# HELP rt\_ifOutUcastPkts Displays the number of transmitted unicast packets on this port.

# TYPE rt\_ifOutUcastPkts gauge

```
rt_ifOutUcastPkts {ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 9.092624e+06
```

```
rt_ifOutUcastPkts {ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 3.184242e+06
```

```
rt_ifOutUcastPkts {ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 2.412616e+06
```

```
rt_ifOutUcastPkts {ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 1.08696e+06
```

```
rt_ifOutUcastPkts {ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 114
```

```
rt_ifOutUcastPkts {ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 135
```

# HELP rt\_ifSpeed An estimate of the interface's current bandwidth in bits per second.

# TYPE rt\_ifSpeed gauge

```
rt_ifSpeed {ethernetportID="23",hostip="10.233.230.11",hostname="host2",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 2
```

```
rt_ifSpeed {ethernetportID="23",hostip="10.233.234.11",hostname="host1",ifAlias="MGMT",ifName="Ethernet 1",job="ethernetport"} 2
```

```
rt_ifSpeed {ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 2
```

```
rt_ifSpeed {ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 2
```

```
rt_ifSpeed {ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 2
```

```
rt_ifSpeed {ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 2
```

```
# HELP rt_redundancyRole When redundancy is configured for 'Failover', indicates if it's role is 'Primary' or 'Secondary'.
```

```
# TYPE rt_redundancyRole gauge
```

```
rt_redundancyRole{ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0
```

```
rt_redundancyRole{ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0
```

```
rt_redundancyRole{ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 1
```

```
rt_redundancyRole{ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 1
```

```
# HELP rt_redundancyState When redundancy is configured for 'Failover', indicates if it's state is 'Online' or 'Backup'.
```

```
# TYPE rt_redundancyState gauge
```

```
rt_redundancyState{ethernetportID="24",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0
```

```
rt_redundancyState{ethernetportID="24",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA0",ifName="Ethernet 2",job="ethernetport"} 0
```

```
rt_redundancyState{ethernetportID="29",hostip="10.233.230.11",hostname="host2",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 1
```

```
rt_redundancyState{ethernetportID="29",hostip="10.233.234.11",hostname="host1",ifAlias="MEDIA1",ifName="Ethernet 3",job="ethernetport"} 1
```

```
# HELP scrape_status /rest/system/
```

```
# TYPE scrape_status gauge
```

```
scrape_status{hostip="10.233.230.11",hostname="host2"} 1
```

```
scrape_status{hostip="10.233.234.11",hostname="host1"} 1
```





## **Appendix C**

# **Standard Agreement and Confidentiality Agreement**

This appendix includes a signed version of the Standard Agreement and Confidentiality Agreement.



*Fastsatt av prorektor for utdanning 10.12.2020*

## **STANDARDAVTALE**

### **om utføring av studentoppgave i samarbeid med ekstern virksomhet**

Avtalen er ufravikelig for studentoppgaver (heretter oppgave) ved NTNU som utføres i samarbeid med ekstern virksomhet.

#### **Forklaring av begrep**

##### **Opphavsrett**

Er den rett som den som skaper et åndsverk har til å fremstille eksemplarer av åndsverket og gjøre det tilgjengelig for allmennheten. Et åndsverk kan være et litterært, vitenskapelig eller kunstnerisk verk. En studentoppgave vil være et åndsverk.

##### **Eiendomsrett til resultater**

Betyr at den som eier resultatene bestemmer over disse. Utgangspunktet er at studenten eier resultatene fra sitt studentarbeid. Studenten kan også overføre eiendomsretten til den eksterne virksomheten.

##### **Bruksrett til resultater**

Den som eier resultatene kan gi andre en rett til å bruke resultatene, f.eks. at studenten gir NTNU og den eksterne virksomheten rett til å bruke resultatene fra studentoppgaven i deres virksomhet.

##### **Prosjektbakgrunn**

Det partene i avtalen har med seg inn i prosjektet, dvs. som vedkommende eier eller har rettigheter til fra før og som brukes i det videre arbeidet med studentoppgaven. Dette kan også være materiale som tredjepersoner (som ikke er part i avtalen) har rettigheter til.

##### **Utsatt offentliggjøring**

Betyr at oppgaven ikke blir tilgjengelig for allmennheten før etter en viss tid, f.eks. før etter tre år. Da vil det kun være veileder ved NTNU, sensorene og den eksterne virksomheten som har tilgang til studentarbeidet de tre første årene etter at studentarbeidet er innlevert.

## 1. Avtaleparter

Norges teknisk-naturvitenskapelige universitet (NTNU) Institutt: Institutt for informasjonssikkerhet og kommunikasjonsteknologi
Veileder ved NTNU: Ernst Gunnar Gran e-post og tlf.: <a href="mailto:ernst.g.gran@ntnu.no">ernst.g.gran@ntnu.no</a> / +4799644916
Ekstern virksomhet: Helsetjenestens Driftsorganisasjon for Nødnett HF Ekstern virksomhet sin kontaktperson, e-post og tlf.: Stig Atle Haugen / <a href="mailto:Stig@hdo.no">Stig@hdo.no</a> / +4791300260
Student: Johannes Hansen Aas Fødselsdato: 18.03.1999
Ev. flere studenter <sup>1</sup> Student: Sondre Jørgensen Fødselsdato: 24.09.1993
Student: Sang Ngoc Nguyen Fødselsdato: 29.04.2001

Partene har ansvar for å klarere eventuelle immaterielle rettigheter som studenten, NTNU, den eksterne eller tredjeperson (som ikke er part i avtalen) har til prosjektbakgrunn før bruk i forbindelse med utførelse av oppgaven. Eierskap til prosjektbakgrunn skal fremgå av eget vedlegg til avtalen der dette kan ha betydning for utførelse av oppgaven.

## 2. Utførelse av oppgave

Studenten skal utføre: (sett kryss)

Masteroppgave	
Bacheloroppgave	X
Prosjektoppgave	
Annen oppgave	

Startdato: 13.01.2023
Sluttdato: 22.05.2023

Oppgavens arbeidstitel er: *Monitorering av Telefoni infrastruktur for mottak av nødsamtaler*

<sup>1</sup> Dersom flere studenter skriver oppgave i fellesskap, kan alle føres opp her. Rettigheter ligger da i fellesskap mellom studentene. Dersom ekstern virksomhet i stedet ønsker at det skal inngås egen avtale med hver enkelt student, gjøres dette.

Ansvarlig veileder ved NTNU har det overordnede faglige ansvaret for utforming og godkjenning av prosjektbeskrivelse og studentens læring.

### **3. Ekstern virksomhet sine plikter**

Ekstern virksomhet skal stille med en kontaktperson som har nødvendig faglig kompetanse til å gi studenten tilstrekkelig veiledning i samarbeid med veileder ved NTNU. Ekstern kontaktperson fremgår i punkt 1.

Formålet med oppgaven er studentarbeid. Oppgaven utføres som ledd i studiet. Studenten skal ikke motta lønn eller lignende godtgjørelse fra den eksterne for studentarbeidet. Utgifter knyttet til gjennomføring av oppgaven skal dekkes av den eksterne. Aktuelle utgifter kan for eksempel være reiser, materialer for bygging av prototyp, innkjøp av prøver, tester på lab, kjemikalier. Studenten skal klarere dekning av utgifter med ekstern virksomhet på forhånd.

Ekstern virksomhet skal dekke følgende utgifter til utførelse av oppgaven:
--

Dekning av utgifter til annet enn det som er oppført her avgjøres av den eksterne underveis i arbeidet.

### **4. Studentens rettigheter**

Studenten har opphavsrett til oppgaven<sup>2</sup>. Alle resultater av oppgaven, skapt av studenten alene gjennom arbeidet med oppgaven, eies av studenten med de begrensninger som følger av punkt 5, 6 og 7 nedenfor. Eiendomsretten til resultatene overføres til ekstern virksomhet hvis punkt 5 b er avkrysset eller for tilfelle som i punkt 6 (overføring ved patenterbare oppfinnelser).

I henhold til lov om opphavsrett til åndsverk beholder alltid studenten de ideelle rettigheter til eget åndsverk, dvs. retten til navngivelse og vern mot krenkende bruk.

Studenten har rett til å inngå egen avtale med NTNU om publisering av sin oppgave i NTNUs institusjonelle arkiv på Internett (NTNU Open). Studenten har også rett til å publisere oppgaven eller deler av den i andre sammenhenger dersom det ikke i denne avtalen er avtalt begrensninger i adgangen til å publisere, jf. punkt 8.

### **5. Den eksterne virksomheten sine rettigheter**

Der oppgaven bygger på, eller videreutvikler materiale og/eller metoder (prosjektbakgrunn) som eies av den eksterne, eies prosjektbakgrunnen fortsatt av den eksterne. Hvis studenten

---

<sup>2</sup> Jf. Lov om opphavsrett til åndsverk mv. av 15.06.2018 § 1

skal utnytte resultater som inkluderer den eksterne sin prosjektbakgrunn, forutsetter dette at det er inngått egen avtale om dette mellom studenten og den eksterne virksomheten.

#### Alternativ a) (sett kryss) Hovedregel

<input checked="" type="checkbox"/>	Ekstern virksomhet skal ha bruksrett til resultatene av oppgaven
-------------------------------------	--

Dette innebærer at ekstern virksomhet skal ha rett til å benytte resultatene av oppgaven i egen virksomhet. Retten er ikke-eksklusiv.

#### Alternativ b) (sett kryss) Unntak

<input type="checkbox"/>	Ekstern virksomhet skal ha eiendomsretten til resultatene av oppgaven og studentens bidrag i ekstern virksomhet sitt prosjekt
--------------------------	---

Begrunnelse for at ekstern virksomhet har behov for å få overført eiendomsrett til resultatene:

#### 6. Godtgjøring ved patenterbare oppfinnelser

Dersom studenten i forbindelse med utførelsen av oppgaven har nådd frem til en patenterbar oppfinnelse, enten alene eller sammen med andre, kan den eksterne kreve retten til oppfinnelsen overført til seg. Dette forutsetter at utnyttelsen av oppfinnelsen faller inn under den eksterne sitt virksomhetsområde. I så fall har studenten krav på rimelig godtgjøring. Godtgjøringen skal fastsettes i samsvar med arbeidstakeroppfinnelsesloven § 7. Fristbestemmelsene i § 7 gis tilsvarende anvendelse.

#### 7. NTNU sine rettigheter

De innleverte filer av oppgaven med vedlegg, som er nødvendig for sensur og arkivering ved NTNU, tilhører NTNU. NTNU får en vederlagsfri bruksrett til resultatene av oppgaven, inkludert vedlegg til denne, og kan benytte dette til undervisnings- og forskningsformål med de eventuelle begrensninger som fremgår i punkt 8.

#### 8. Utsatt offentliggjøring

Hovedregelen er at studentoppgaver skal være offentlige.

Sett kryss

<input checked="" type="checkbox"/>	Opgaven skal være offentlig
-------------------------------------	-----------------------------

I særlige tilfeller kan partene bli enige om at hele eller deler av oppgaven skal være undergitt utsatt offentliggjøring i maksimalt tre år. Hvis oppgaven unntas fra offentliggjøring, vil den kun være tilgjengelig for student, ekstern virksomhet og veileder i denne perioden. Sensurkomiteen vil ha tilgang til oppgaven i forbindelse med sensur. Student, veileder og sensorer har taushetsplikt om innhold som er unntatt offentliggjøring.

Opgaven skal være underlagt utsatt offentliggjøring i (sett kryss hvis dette er aktuelt):

Sett kryss	Sett dato
<input type="checkbox"/>	ett år
<input type="checkbox"/>	to år
<input type="checkbox"/>	tre år

Behovet for utsatt offentliggjøring er begrunnet ut fra følgende:

Dersom partene, etter at oppgaven er ferdig, blir enig om at det ikke er behov for utsatt offentliggjøring, kan dette endres. I så fall skal dette avtales skriftlig.

Vedlegg til oppgaven kan unntas ut over tre år etter forespørsel fra ekstern virksomhet. NTNU (ved instituttet) og student skal godta dette hvis den eksterne har saklig grunn for å be om at et eller flere vedlegg unntas. Ekstern virksomhet må sende forespørsel før oppgaven leveres.

De delene av oppgaven som ikke er undergitt utsatt offentliggjøring, kan publiseres i NTNUs institusjonelle arkiv, jf. punkt 4, siste avsnitt. Selv om oppgaven er undergitt utsatt offentliggjøring, skal ekstern virksomhet legge til rette for at studenten kan benytte hele eller deler av oppgaven i forbindelse med jobbsøknader samt videreføring i et master- eller doktorgradsarbeid.

## 9. Generelt

Denne avtalen skal ha gyldighet foran andre avtaler som er eller blir opprettet mellom to av partene som er nevnt ovenfor. Dersom student og ekstern virksomhet skal inngå avtale om konfidensialitet om det som studenten får kjennskap til i eller gjennom den eksterne virksomheten, kan NTNUs standardmal for konfidensialitetsavtale benyttes.

Den eksterne sin egen konfidensialitetsavtale, eventuell konfidensialitetsavtale den eksterne har inngått i samarbeidprosjekter, kan også brukes forutsatt at den ikke inneholder punkter i motstrid med denne avtalen (om rettigheter, offentliggjøring mm). Dersom det likevel viser seg at det er motstrid, skal NTNUs standardavtale om utføring av studentoppgave gå foran. Eventuell avtale om konfidensialitet skal vedlegges denne avtalen.

Eventuell uenighet som følge av denne avtalen skal søkes løst ved forhandlinger. Hvis dette ikke fører frem, er partene enige om at tvisten avgjøres ved voldgift i henhold til norsk lov. Tvisten avgjøres av sorenskriveren ved Sør-Trøndelag tingrett eller den han/hun oppnevner.

Denne avtale er signert i fire eksemplarer hvor partene skal ha hvert sitt eksemplar. Avtalen er gyldig når den er underskrevet av NTNU v/instituttleder.

### Signaturer:

Instituttleder:	
Dato:	
Veileder ved NTNU:	
Dato:	
Ekstern virksomhet:	
Dato:	17/1-23 Stig Arle Høy
Student:	
Dato:	17/1-23 Sondre Jørgensen
Student:	
Dato:	17/1-23 Jonas
Student:	
Dato:	17.01.2023 Dang Ngoc Nguyen





Norges teknisk-naturvitenskapelige universitet

Fastsatt av prorektor for utdanning 10.12.2020

**STANDARDMAL ved avtale om konfidensialitet** mellom student og ekstern virksomhet i forbindelse med studentens utførelse av oppgave (master-, bachelor- eller annen oppgave) i samarbeid med ekstern virksomhet, jf. punkt 9 i standardavtale om utføring av oppgave i samarbeid med ekstern virksomhet.

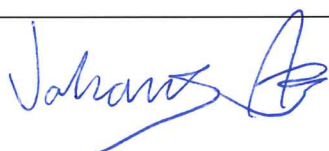
Student ved NTNU: Johannes Hansen Aas Fødselsdato: 18.03.1999
Hvis flere studenter: Student ved NTNU: Sondre Jørgensen Fødselsdato: 24.09.1993
Student ved NTNU: Sang Ngoc Nguyen Fødselsdato: 29.04.2001
Ekstern virksomhet: Helsetjenestens driftsorganisasjon for nødnett HF, Hans Mustads gate 31, Postboks 72, 2801 Gjøvik

1. Studenten skal utføre oppgave i samarbeid med ekstern virksomhet som ledd i sitt studium ved NTNU.
2. Studenten forplikter seg til å bevare taushet om det han/hun får vite om tekniske innretninger og fremgangsmåter samt drifts- og forretningsforhold som det vil være av konkurransemessig betydning å hemmeligholde for den eksterne virksomheten. Det er den eksterne sitt ansvar å sørge for å synliggjøre og tydeliggjøre hvilken informasjon dette omfatter.
3. Studenten er forpliktet til å bevare taushet om dette i 5 år regnet fra sluttdato.
4. Kravet om konfidensialitet gjelder ikke informasjon som:
  - a) var allment tilgjengelig da den ble mottatt
  - b) ble mottatt lovlig fra tredjeperson uten avtale om taushetsplikt
  - c) ble utviklet av studenten uavhengig av mottatt informasjon
  - d) partene er forpliktet til å gi opplysninger om i samsvar med lov eller forskrift eller etter pålegg fra offentlig myndighet.

#### Signaturer

Student: <i>Sang Ngoc Nguyen</i>
Dato: <i>17.01.2023</i>

Student:

Johann 

Dato:

17/1 - 23

Student:

Sondre Jørgensen

Dato:

17/01/2023

Ekstern virksomhet:

Stig Ole Heuge 

Dato:

17/1 -23

## **Appendix D**

# **The Project Plan**

This appendix includes a signed version of the Standard Agreement and the Confidentiality Agreement.





# NTNU

Kunnskap for en bedre verden

DCSG2900 – Bachelor Thesis

Project Plan – Group 119

Johannes Hansen Aas

Sang Ngoc Nguyen

Sondre Jørgensen

January 2023

## Table of Contents

Table of Contents.....	2
1. Background and Goals .....	3
1.1. Result Goals .....	3
1.2. Effect Goals .....	3
1.3. Learning Goals .....	3
1.4. Resource Needs .....	4
2. Scope.....	4
2.1. Task Description.....	4
2.2. Delimitations.....	6
3. Project organization.....	6
3.1. Routines and Group Rules .....	7
4. Planning, follow-up and reporting.....	8
4.1. Description of How the group will follow the development model.....	8
4.2. Plan for status meetings and decision moments in the period.....	9
5. Organization of quality assurance .....	9
5.1. Plan for inspections and testing .....	10
5.2. Risk Analysis .....	10
5.3. Risk Assessment .....	11
5.4. Measures .....	12
6. Plan for project execution .....	13
6.1. Activities, Milestones and Decision Moments .....	13
7. Bibliography .....	15

## 1. Background and Goals

HDO (The health service' operation-organization for emergency network) delivers services for realizing the Norwegian emergency reporting service called "Nødnett". It aims to deliver efficient, reliable and user-friendly services for the emergency network in all municipalities of Norway. During the recent years HDO has rebuilt their infrastructure for the Norwegian emergency call numbers and created solutions for analyzing and uncovering errors within the core of the infrastructure in real-time (Haugen, 2022).

HDO wishes to expand its collection of data from emergency calls and Session Border Controllers, by utilizing an already existing API that communicates with these SBCs that are a part of their infrastructure. They wish to do so by collecting data from a REST API, using container technology and open-source tools like Grafana, Prometheus and Loki. As of January 2023, HDO does not utilize the API on the SBCs for anything, therefore they want the group to make use of this API to collect and analyze data for monitoring of both trends on emergency calls and performance trends of these SBCs. To achieve this HDO wants the group to make a program that can collect data from the API and put the collected data into Prometheus. In addition to that, this program called an exporter must run in Docker.

Prometheus is a technology used for monitoring and alerting by collecting and storing numeric measurements called metrics.

### 1.1. Result Goals

The goal for this project is to deliver a monitoring system to HDO that continuously gathers and visualizes data from their edge routers, which their employees can use to monitor their networks. This system should be able to be deployed on any virtual machine in HDO's network, it should be secure, and it should be easy to set up and use. There should also be provided a user manual for the system that describes both how to deploy and use it.

### 1.2. Effect Goals

The product being developed in this project aims to make emergency call data more accessible and will potentially make monitoring trends in the call network easier. The effect of our product aims to make useful use of the already existing API to uncover trends in both call data and performance data from these SBCs to avoid unnecessary downtime of these SBCs and keep the quality of the emergency calls to a maximum.

### 1.3. Learning Goals

Through this project, the group's goal is to gain insight in how to secure and operate social critical infrastructure and contribute to Norway's emergency preparedness, by making emergency call data

more accessible. We will also acquire knowledge about tools used to develop, operate and detect events in applications and infrastructure, which are widely used in Site reliability engineering (SRE)/DevOps.

#### 1.4. Resource Needs

This project will require:

1. A test environment consisting of Linux servers to run docker containers on. This will be provided by HDO.
2. Access to the Ribbon Session Border Controller 1000/2000. This is the hardware that contains various data from emergency calls.
3. The SBC Edge 9.0.x REST API
4. There might be a need for one of Grafana's paid plans.

## 2. Scope

This project will cover multiple subject areas; such as building infrastructure, which includes programming and utilizing technologies like Prometheus and Docker, but also understanding the complex network that handles emergency calls. We will also work with DevOps practices such as CI/CD pipeline.

### 2.1. Task Description

HDO wants to utilize the existing SBC Edge REST API that comes standard with the equipment in their infrastructure by making a system for monitoring data collected from this infrastructure, as well as notifying its users during specific events. Our task is to create a centralized monitoring system based on the technologies Prometheus and Grafana and utilizing docker containers for packaging each technology (Haugen, 2022).

The system should be developed using the following technologies and tools:

- **Prometheus**

Prometheus is an open-source technology for pulling data from a wide array of sources and formats and processing and storing this data. Prometheus also has functionality like generating alerts when certain user defined thresholds are met, and more. Prometheus utilizes exporters, which are plugins developed for pulling and filtering data from a unique source. This exporter ensures that the data is converted into a format readable by Prometheus' database system.

- **Golang**



Using a programming language is necessary for this project because the system to be developed is unique, meaning it does not have any documentation on Prometheus' official website. Because Prometheus is open source and has many contributions from the community, it is often possible to create a monitoring system using these prebuilt exporters. For this project however, there needs to be developed a custom exporter for Prometheus written in a programming language such as Golang, as there are no available exporters online for HDO's routing equipment (SBCs).

Golang (Go) is a programming language developed by google which has benefits over older languages, such as CPU concurrency, efficiency in both compile time and run time. There are several programming languages that can be used for writing Prometheus exporters. This group has chosen Go because of its large support on its internet community, which may aid in the development process.

- **Grafana**

While Prometheus collects and stores data, Grafana is a tool for monitoring data from several different sources, where Prometheus is only one example of such a source. Grafana visualizes data using dashboards that can be customized by the user.

- **Docker**

In the case for this project, Docker will simplify the process of deploying a stack consisting of containers for each technology, using prebuilt images downloaded from docker's official pages. This is implemented using a docker-compose file and settings files for each container. Docker has benefits such as efficiency and ease of deployment and maintaining code, less reliant on software dependencies, and better security.

### **Overview**

This monitoring system will help to uncover errors and bottlenecks within HDO's infrastructure and ensure the overall quality of the emergency network. They want this system to be implemented by using opens source tools like Prometheus, Grafana, Loki and Docker. The development should also utilize a CI/CD pipeline. In figure 1, we see the flow of data from HDO's edge routers through the parts that will make up the project. The Data will first be fetched, then processed and stored on a virtual machine in HDO's network. The monitoring software with visualized data on it should be able to be accessed using this virtual machine's ip-address and the software's port number.

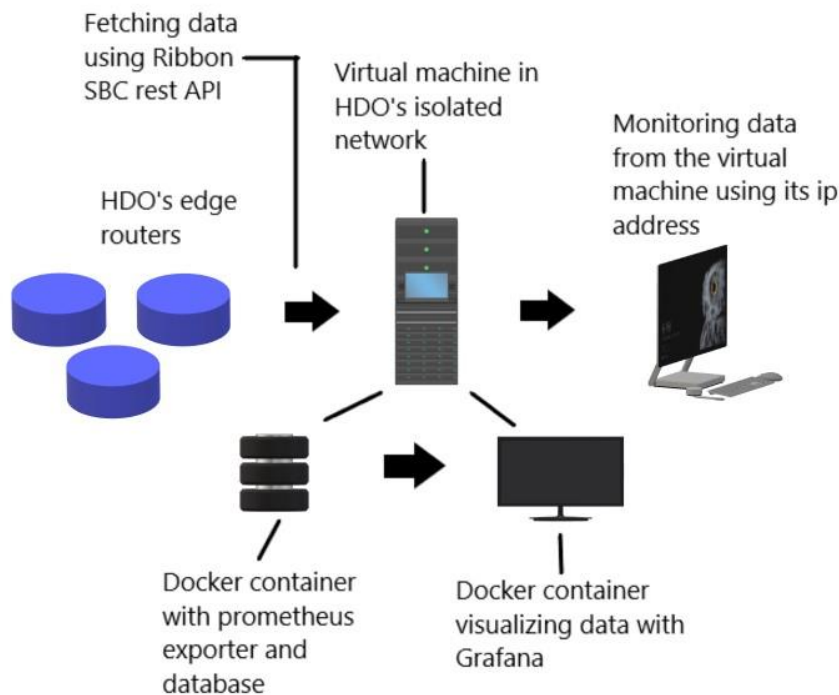


Figure 1: The flow of data from HDO's edge routers to visualizing it with Grafana.

## 2.2. Delimitations

Our solution will be limited to creating a product for monitoring data gathered from the SBC Edge REST API, and with this product creates a system for alerting specific users during events related to errors or weaknesses within HDO's infrastructure. We are explicitly going to work on collecting data from the Ribbon SBC Edge REST API. We could create our solution to be able to collect data from other APIs as well, but we have chosen not to because it is not an integral part of making our solution work for the API specified in the task description, which is the SBC Edge REST API.

According to Prometheus' documentation, to create a perfect result, a solution containing a substantial number of metrics (in the hundreds) will require a lot of work whereas to create an imperfect solution with fewer metrics is a lot easier (Prometheus, 2023). The goal will be to create a Prometheus exporter that collects all metrics from the SBC API and associated measurements for HDO's infrastructure, depending on the needs of our employer. However, the difficulty level will play a role in how many metrics we have time to include.

## 3. Project organization

The roles and responsibilities in this project are as follows:

Team members:

- Project leader: Johannes Aas
- Secretary: Sang Ngoc Nguyen
- Contact person: Sondre Jørgensen

Project supervisor: Ernst Gunnar Gran, NTNU

HDO via: Stig Atle Haugen, HDO

### 3.1. Routines and Group Rules

#### **Time logging**

All group members will log hours spent working on the project, containing date, time spent, and a brief description of the work. This will be done on an Excel spreadsheet. Every team member is expected to work around 30 hours per week.

#### **Meetings**

Attending meetings is mandatory for all group members. If a team member falls ill or cannot attend meetings or work activities, the group members must be notified in advance.

#### **General rule**

If we face any internal issues in the group, our first step is to discuss the issue internally with all group members. Disagreeing on how we should do a task is an example of an internal issue the group could face. Deciding on how we will do that task will be done democratically with voting where the majority will decide. If the issue persists, then we will ask a third party such as the supervisor or client to help us decide what to do.

#### **Work hours**

We have decided that if one or more group members have a considerably lower number of hours worked, then we should be able to demand that they work more to “catch up” on the work that needs to be completed from that point on until the amount of work done is more equally distributed between the members.

## 4. Planning, follow-up and reporting

For the choice of development model for our project, we had many options to choose from. The three main candidates for the development model that we felt were best suited were Kanban, Scrum and Lean. After a lot of discussion, we decided to go with the “Scrum” software development model.

The reason that we have chosen to go with Scrum is because we feel like that Scrum was the best suited development model for our project in terms of our goals and how we plan to achieve them. Scrum allows us some flexibility in terms of adapting based on feedback and needs that we observe while working on the product between the sprints. While Scrum doesn't allow for as much flexibility in terms of changes to requirements and scope compared to a development model like Kanban, we still chose to go with Scrum because we felt that it is beneficial to have some sort of structure and a plan to our project (Lucidchart, n.d.).

We feel that software development models such as the Waterfall method are too strict when it comes to progress and therefore chose to exclude them. We think that the probability of changes to the requirements or scope happening is quite low, so we didn't mind less flexibility and the overall structured feel of Scrum. We also discussed Kanban but felt that it was too free and could end up leading to a messy and unstructured implementation to our project. Therefore, we concluded that Scrum was a good middle ground between a structured and a flexible software development model that fit the group the best.

Another possibility was to use the Lean Agile development model. It mostly suits our needs in terms of producing an MVP (Minimum Viable Product) and adapting our project based on feedback. However, it does not consider other matters such as meetings and documentation of our work (Blake, 2021). Hence, we chose not to go with the Lean development model.

### 4.1. Description of How the group will follow the development model

We plan to use Scrum by structuring the project into 1-week sprints with one stand-up meeting each week where we review our progress and how things are going. For example, if a member has finished their work for the sprint, they may assist others with their work. Before each sprint we will do a sprint review where we review the progress that we have made during the last sprints. We will also have a session before each sprint where we will plan our next sprint so that we have a plan that we can follow so that we don't work without a goal or direction.

#### 4.2. Plan for status meetings and decision moments in the period

We plan to hold meetings every Monday at 12:00. In this meeting, we will discuss topics such as what the plan is for the next week and what we have worked on the previous week as part of the pre-Scrum meetings. These regular meetings are a mix of planning and working and are planned to last for about 4 hours but could be longer or shorter depending on our needs.

Meetings with our supervisor are planned to take place every Thursday between 10:30 and 11:00, and meetings with our client are planned to be allocated dynamically. The plan is to have them every other week, either physically at HDO's workplace or online on Teams if a physical meeting is not required for the meeting's purpose. The meetings with the supervisor are planned to take place physically but can also be online if needed. In the meetings with the supervisor and client, we plan to discuss issues that we have faced, questions that we may have and ask for feedback and advice on our work so far.

### 5. Organization of quality assurance

We plan to use several tools for organizing our project, these include:

#### **SharePoint**

We plan to use SharePoint for storing and working with all shared files not containing code. We considered using google drive but concluded that SharePoint allows for more space to be stored, as a license is paid for by NTNU. Files within our SharePoint site include an excel sheet for work hours, documentation, and other resources.

#### **Microsoft Teams**

This is where we will arrange all online meetings if we are unable to meet physically. These meetings include both the ones within our group as well as meetings with our employer at HDO and our supervisor.

#### **Microsoft Excel**

We will use Excel for time tracking.

#### **Discord**

Discord will be used for all online discussions, including planning sessions, general discussions and sharing small notes and resources.

## **Git**

All group members have experience in working with Git and we all understand its key features. Git simplifies the process of cooperating on the same code as well as coding individually by using version control, merging of code from multiple developers and resetting the local code environment to the last stable version and more.

We have a shared understanding of how we want to use Git. We should commit (upload) smaller pieces with understandable and logical comments whenever a new component is ready. This is to ensure better version control with easier debugging as the entire system could fail even if the error resides only within a smaller part. All code should work integrated with the entire system before it is pushed to the main branch in Git unless it is agreed to for each case. This is to ensure the quality of the code and because working with multiple errors in the code removes the principle of proof of concept, meaning it will be impossible to know when something is working if the system doesn't work initially.

We will also use Git to keep track of tasks that need to be finished. For this we will use the issue feature in Git to create issues that need to be worked on.

## **Overleaf**

The bachelor thesis shall be written in latex using overleaf as it is quite common and highly recommended for academic writing. This tool allows all team members to collaborate on the same version and makes keeping track of changes easy.

### [5.1. Plan for inspections and testing](#)

All code development in our project will be test-driven development, and all code should produce a desired outcome before being pushed to our shared git repository. This means that whenever new code is added, we should have a test to see whether this has had the desirable outcome. If a newly added piece of code (component) is only tested in isolation, then we should perform an integration test to ensure this component works on the entire system before pushing it to our Git repository.

Most of what we do in terms of developing the product shall be documented individually for each group member. Some of this documentation is to be used indirectly for the bachelor thesis, for example, to write the user manual for the system. However, it is also to allow the other group members to learn from it and for us to remember what we have done.

### [5.2. Risk Analysis](#)

We have taken risk into account by listing several scenarios that may result in the failure of our delivery, or an unsatisfactory result. By knowing in advance, the risk scenarios that can occur during the project we can respond appropriately to prevent them from occurring or escalating. Having a predetermined

understanding of these risky events also allows us to create a set of rules more easily we can agree to within our group and to remove prolonged disagreements if they occur.

### 5.3. Risk Assessment

We have listed several cases that may result in a failure of our delivery or an unsatisfactory result. These include:

#### **1. Lack of competence result in not being able to develop a satisfactory product**

As neither of us have any experience working with the technologies we have at hand, the learning process may take longer than expected or may not be sufficient for our project. We have considered this risk as the most likely to occur and could have a very negative impact on our project. In order to combat this risk, it is important that all group members work sufficiently, structured and search for help when difficult problems arise.

#### **2. Unexpected errors in the system without finding the cause**

Unexpected errors that are highly demanding are common during software development and in worst case scenario it may take longer than a week to solve.

#### **3. HDO's servers fail**

If there are problems with the resources delivered by HDO that are essential for doing our work, then this could be critical to developing the product as it depends on data from their services. One way of solving this is to create dummy data for the exporter and a mock component instead of the SBC Edge REST API provided by HDO's infrastructure.

#### **4. One or more of our group members get sick for prolonged periods of time**

This can have serious consequences if the other group members lack the skills and time to replace his effort.

#### **5. One or more of our group members are not working sufficiently**

As stated in the rules we have decided that if one or more group members have a considerably lower number of hours worked, then we should be able to demand that they work more hours from that point on until the numbers are more equally distributed between the members.

**6. Our employer at HDO is unable to help us with a critical problem**

Our employer may get sick or unable to help us sufficiently as he has stated he will have a lot of work this spring.

**7. The work on writing the bachelor thesis has been neglected as all the time has been spent on developing the product.**

This may occur if developing the product has been highly demanding. It is important to distribute the time according to the plan and adjust when needed.

In this diagram we assess the risk scenarios based on their probability and consequence of occurring. Combining their consequences and probability gives four degrees of dangers; green, yellow, orange and red. The commas are separation of different risk scenarios.

	Consequence			
	Insignificant	Small	Serious	Critical
Very high probability				
High probability				1
Unlikely			4, 5	2, 7
Very unlikely		6	3	

Figure 2: Assessment of each risk scenarios

**5.4. Measures**

We have a set of rules (3.2 Routines and group rules) that determine how a risk scenario should be solved and the group have a common understanding of certain measures that may help us. These measures and rules include:

- Regular group meetings that attempt to solve the issues.
- A democratic process that works as final decisions for issues concerning the overall work. For example, if there are disagreements on the details of the product then the majority decides. However, this does not mean we should not have discussions about the issues. It is beneficial to hear all opinions thoroughly before deciding.
- Communications with supervisor in case of major disagreements or other problems for our group.
- Regular meetings with our supervisor and our client at HDO will help in solving issues.



## 6. Plan for project execution

ACTIVITY	PLAN START	PLAN DURATION (weeks)	WEEK nr.																			
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Establish regular communication with employer	2	2		■	■																	
Establish regular communication with supervisor	2	2		■	■																	
Confidentiality agreement and standard agreement	2	2		■	■																	
Create project plan	2	4		■	■	■	■															
Testing/learning the API	4	3			■	■	■															
Research					■	■	■	■	■	■	■	■										
Proof of concept/testing	5	4			■	■	■	■	■													
Developing	9	5								■	■	■	■	■								
CI/CD pipeline	10	3								■	■	■										
Create minimum viable product (MVP)	12	2										■	■									
Documenting	5	9			■	■	■	■	■	■	■	■	■									
Create user manual	13	1										■										
Bachelor thesis (first draft)	13	5										■	■	■	■	■						
Finalize thesis	18	3																	■	■	■	

Figure 3: Gantt-diagram for scheduling time working with the project as a whole

### 6.1. Activities, Milestones and Decision Moments

#### Milestone 1:

For our first milestone we plan to finish the project plan, finish signing the confidentiality- and standard agreement.

#### Milestone 2:

For our second milestone we plan to finish necessary research to create a proof of concept for the exporter.

#### Milestone 3:

For our third milestone we plan to start development of the actual product, make a CI/CD pipeline and finish the MVP.

**Milestone 4:**

For our fourth milestone we plan to create a user manual for the exporter and to start on our first draft of the project report/bachelor thesis. Before easter break we will finish 3 chapters of the first draft and send it to our supervisor Ernst.

**Milestone 5:**

For our fifth milestone we plan to finalize the project report/bachelor thesis to be delivered.

## 7. Bibliography

Ribbon Communications (n.d.) *SBC 1000/2000 API Home*. Available at: <https://support.sonus.net/display/UXAPIDOC/> (Accessed: 26 January 2023).

Ribbon Communications (n.d.) *REST API User's Guide*. Available at: <https://support.sonus.net/display/UXDOC90/REST+API+User%27s+Guide> (Accessed: 26 January 2023)

Lucidchart (n.d.) *Agile vs. Waterfall vs. Kanban vs. Scrum: What's the Difference?*. Available at: <https://www.lucidchart.com/blog/agile-vs-waterfall-vs-kanban-vs-scrum> (Accessed: 26 January 2023)

Haugen, SA (2022) *Monitoring of telephone infrastructure for reception of emergency calls. Description of bachelor thesis by HDO*. (Accessed: 31 October 2022)

Prometheus (n.d.) *Instrumentation*. Available at: <https://prometheus.io/docs/practices/instrumentation/> (Accessed 13 January 2023)

Hjelmås, E. (2022) *DCSG2900 - Bachelor Thesis Bachelor of Science in Digital Infrastructure and Cyber Security*. Available at: <https://www.ntnu.edu/studies/courses/DCSG2900#tab=omEmnet> (Accessed: 26 January 2023)

Blake, S (2021) *Understanding Lean Agile and the 5 Lean Principles*. Available at: <https://www.easyagile.com/blog/lean-agile/> (Accessed: 26 January 2023)



## Appendix E

# Task Description

This appendix includes the task description by *HDO*.



# Monitorering av Telefoni infrastruktur for mottak av nødsamtaler

*HDO – landsdekkende,  
tilgjengelig og nyskapende*



# Helsetjenestens driftsorganisasjon for nødnett HF (HDO)

## Om

HDO skal bidra til å realisere de samlede målsetninger for den nasjonale medisinske nødmeldetjenesten. Selskapet skal yte effektive og brukervennlige tjenester for brukere av Nødnett i den akuttmedisinske kjeden i alle de regionale helseforetakene, i alle landets kommuner, og for andre relevante samarbeidspartnere. Vår oppgave er å sørge for enhetlige og stabile kommunikasjonsløsninger og fagsystemer, herunder teknisk utvikling, test, implementering, drift og opplæring av brukere. HDO er organisert som en del av spesialisthelsetjenesten og er eid av de 4 helseregionene.

HDO har over de siste årene bygget opp en ny og omfattende infrastruktur for håndtering av mottak av telefoni 113 (nødanrop) og 116 117. Denne infrastrukturen består et stort antall komponenter som til sammen utgjør et nasjonalt telefoni nettverk.

For de sentrale komponentene har HDO etablert en rekke løsninger for analyse og varsling for hendelser i infrastrukturen. På grunn av kritikaliteten i løsningen er målet hele tiden å kunne avdekke feil før de eskalerer, eller merkes. For å få til dette må HDO samle og analysere data fra ulike komponenter i nettverket i sanntid.

## Oppgaven

HDO ønsker nå å utvide innsamlingen av informasjon for alle kantelementer som er utplassert lokalt på ulike lokasjoner i Norge.

For å realisere dette så ønsker oppdragsgiver å nyttiggjøre seg bedre de eksisterende REST API'er, som infrastrukturen har. Oppdragsgiver har valgt å bygge løsningen for analyse og monitorering basert på kontainer teknologi (Docker), og Open Source verktøy som: Prometheus, Loki, Grafana etc. I tillegg kommer en rekke Prometheus exportere og andre elementer.

Hovedkomponenten for innsamling, analyse og varsling er Prometheus. Prometheus samler inn og lagrer sine beregninger som tidsseriedata, det vil si at metrikkinformasjon lagres med tidsstempelen den ble registrert på, sammen med valgfrie nøkkelverdi-par, kalt etiketter.

Igjennom oppgaven vil gruppen få innsikt i hva som kreves for å sikre og operere samfunnskritisk infrastruktur, som er kritisk for beredskapen i Norge. De vil også tilegne seg kunnskap rundt verktøy som benyttes for å kunne utvikle, drifte og detektere hendelser i applikasjoner og infrastruktur, som er mye brukt i SRE/DevOps.

Oppdragsgiver vil stille med en test/utviklingsplattform for gjennomføring av oppgaven.



## Oppgavens mål:

Hovedmålet med oppgaven er å lage en integrasjon mellom kantelementene og Prometheus for å kunne lese ut data, via API'et. Videre må det sikres at dette lagres og struktureres slik at det kan benyttes til overvåkning og trendvarsling i en driftsorganisasjon.

- Gruppen må i samarbeid med oppdragsgiver kartlegge og vurdere tilgjengeligheten på data og hva som kan hentes ut fra kantelementene via API'et. API'et er dokumentert fra leverandør.
- De må finne en metode for å lese ut data og få dette tilgjengeliggjort som metrikkinformasjon for Prometheus.
- Løsningen må ha støtte for å konfigurere hvilke elementer som skal leses ut, og utlesing fra flere kilder samtidig.
- Løsningen skal være en exporter for prometheus som bør kunne kjøres i docker.
- Det anbefales at det brukes en CI/CD pipeline, for utviklingen.
- Gruppen må selv etablere opp sitt eget test/utviklingsmiljø på tildelt kapasitet, med Prometheus/Grafana og en CI/CD pipeline.

## Kontaktpersoner i HDO

Stig Atle Haugen, Stig@hdo.no, +47 91 300 260



# Appendix F

## Repository

This appendix includes the repository for the project and its README file.



**Link to the Repository**

[https://github.com/Sonjorg/edge\\_exporter](https://github.com/Sonjorg/edge_exporter)

**README File**

# Readme

---

## Prometheus exporter for Ribbon Communications SBC routers

---

Developed by Sondre Jørgensen in cooperation with Sang Ngoc Nguyen at NTNU: Norwegian University of Science and Technology, [sondre2409@gmail.com](mailto:sondre2409@gmail.com) and [29sangu@gmail.com](mailto:29sangu@gmail.com)

### Configuration of the exporter

The configuration is implemented in config.yml in the root folder of the source code.

```
---
authtimeout: 3 #all hosts will have max 3 sec timeout
hosts:
- hostname: Host1
  ipaddress: 11.111.111.11
  username: Username1
  password: Password1
  routing-database-hours: 24 #For routingentry collector, data is stored
                             #in the database for 24 hours for this host.
- hostname: Host2
  ipaddress: 11.111.111.12
  username: Username2
  password: Password2
  routing-database-hours: 24
- hostname: Host3
  ipaddress: 11.111.111.13
  username: Username3
  password: Password3
  routing-database-hours: 24
  exclude:
    - routingentry
    - system
    - diskpartition
    - systemcallstats
    - linecard
    - ethernetport
#Excluding the above collectors for this host
```

- Above you can see the layout of a config.yml file having 3 hosts with dummy data.
- It is required to use a hostname, ipaddress, username and password.
- You can choose which collectors you want to exclude for each host by adding them to the list "exclude" as shown below the last host. The name of the collectors have to match exactly as spelled in this example.

- "Authtimeout" is the maximum chosen time to attempt authentication to a host. Usually it is not reachable if the duration is more than 1-2 second.
- "routing-database-hours" is the duration of which data related to the routingentry collector is stored within the database. Fetching new data through http takes several extra seconds per scrape. Metrics are never stored, only data such as routing tables and their routing entries.
- It is recommended not to use too many hosts per docker instance because of performance issues; a scrape on 2 hosts with no collectors excluded takes around 13 seconds on the first scrape, and around 10 seconds on the following scrapes.

## Deployment running docker

- Run: `sudo docker build -t edge_exporter .` `sudo docker run -p 5123:5123 edge_exporter`
- Or if you have an external config.yml file: `sudo docker run -v path/to/your/config.yml:/usr/src/exporter/config.yml sondrjor/edge_exporter`
- Metrics can be gathered from `host:5123/metrics`

## Deployment of the SBCexporter on a linux server

The exporter is developed and tested for the official ubuntu server image found at <https://ubuntu.com/download/server>.

- Download golang using the official download page: [install golang](#), and remember to reboot
- To start the exporter and download all necessary packages, navigate to the SBCexporter directory and run `go install`

## To test go exporters:

`go run .` in the SBCexporter directory, then use `curl localhost:9100/metrics` in another windows to view live metrics data that can be collected by prometheus

## To test a specific file, for use

`go run main.go` However this will not make use of dependencies from other files

## Installation of Go on HDO's VMs

As root folders are not accessible on HDO's VMs we need to install Go in home directory if docker is not utilized

- Download last version of Go to home directory, from Go's official website
- Unzip the file with tar
- Execute the commands: `export GOPATH=$HOME/go` `export PATH=$PATH:$GOPATH/bin`
- If starting Go gives a message that its not yet installed, make a startup script that executes: `source .bashrc` from home directory

# Grafana and prometheus setup with docker

---

Choose between grafana local or grafana cloud

## Grafana local

This is a setup with grafana-docker hosted locally, following a similar approach as this tutorial:

<https://www.youtube.com/watch?v=9TJx7QTrTyo&t=712s>

The config for all docker images used, resides in the docker-compose.yml file

## Deployment of grafana with docker

Use `docker compose up -d` in either directory `edge_exporter\Other\Grafana-Prometheus\grafanacloud` or `.../grafanalocal`, respectively

## test docker containers:

---

### get ip address of grafana container

```
sudo docker inspect -f '{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' grafana
```

```
curl ip-address:3000
```

### Restart all containers if changes are made to docker-compose.yml

```
docker-compose up -d --force-recreate
```

### check status in log files for a container

```
sudo docker-compose logs -f container-name
```



## **Appendix G**

# **Minutes of Meeting**

This appendix includes the task minutes of meeting. The language in these is a mixture of mostly English and some Norwegian.



# Meeting 0

søndag 15. januar 2023 02:06

Date	December 7th 2022
Place	Teams
Participants	Johannes, Sondre, Sang and Stig Atle
Agenda	First meeting with HDO
Summary	The group discussed the task with the employer.

# Meeting 1

onsdag 11. januar 2023 16:24

Date	January 11th 2023
Place	On campus
Participants	Johannes, Sondre and Sang
Agenda	First group meeting
Summary	<ul style="list-style-type: none"><li>• Plan the project report</li><li>• Need to contact supervisor</li><li>• Prepare the confidentiality agreement and standard agreement</li><li>• Have 2 physical meetings weekly</li><li>• Talked about grade ambitions</li><li>• Delegated roles:<ul style="list-style-type: none"><li>• Group leader: Johannes</li><li>• Secretary: Sang</li><li>• Contact person: Sondre</li></ul></li><li>• Discussed time tracking</li> <li>• Before next meeting<ul style="list-style-type: none"><li>• Read course description</li><li>• Make a calendar for meetings and other important events</li><li>• Read the project/task description</li></ul></li></ul>

# Meeting 2

fredag 13. januar 2023 14:43

Date	January 13th 2023
Place	On campus
Participants	Johannes, Sondre and Sang
Agenda	Discuss project plan
Summary	<ul style="list-style-type: none"><li>• Discussed how we will do time tracking</li><li>• Delegated writing tasks on the project plan</li><li>• Discussed what to do before next meeting</li><li>• Discussed what programming language to use for creating the exporter</li><li>• Made a Gantt-diagram</li><li>• Discussed development model</li> <li>• To do for before next meeting:<ul style="list-style-type: none"><li>• Research CI/CD pipeline and Prometheus exporters</li><li>• Ask Ernst about how to Gantt-diagram looks</li></ul></li></ul>

# Meeting 3

søndag 15. januar 2023 02:20

Date	January 15th 2023
Place	On campus
Participants	Johannes, Sondre and Sang
Agenda	Discuss what we have worked on What to do next Work on the project plan
Summary	<ul style="list-style-type: none"><li>• Worked a little bit on the project plan</li><li>• Discussed regular meetings with our employer</li></ul>

# Meeting 4

mandag 16. januar 2023 13:13

Date	January 17th 2023
Place	At HDO
Participants	Johannes, Sondre, Sang and Stig Atle
Agenda	<ul style="list-style-type: none"><li>• Discuss the task</li><li>• Discuss agreements</li><li>• Discuss regular meetings with employer</li><li>• Discuss how to access the server</li><li>• Discuss test environment</li> <li>• HDO Agenda<ul style="list-style-type: none"><li>• Signere avtaler</li><li>• Kort omvisning</li><li>• Åpen dialog<ul style="list-style-type: none"><li>◦ Gruppen presenterer tanker rundt prosjektgjennomføring (overordnet)</li><li>◦ Dialog rundt praktisk organisering</li></ul></li><li>• 13.00 - 14.XX gjennomgang av oppgaven / arkitektur</li><li>• 14.30: sjekk på at tilganger til horizon og servere fungerer som de skal</li><li>• 14.50 oppsummering og avslutninger</li></ul></li></ul>
Summary	<ul style="list-style-type: none"><li>• Signed papers with employer (for VPN access, taushetsærklring, standard agreement and confidentiality agreement), will get username and password</li><li>• HDO can allocate more servers if needed</li><li>• To access the ONE linux server we have to vpn and ssh</li><li>• 3 clients for every student is provided</li><li>• Prometheus and Docker on same server to avoid usage of ports between servers</li><li>• Will add Stig Atle to our bachelor Teams channel, stigatle.haugen@hdo.no</li><li>• Looked at the task<ul style="list-style-type: none"><li>• Ribbon communications REST API</li><li>• Find out which metrics are the most important to GET</li><li>• We will use VMware Horizon Client</li></ul></li><li>• Discussed regular meetings every other week, dates are to be determined because of Stig Atle's busy schedule. Meetings will be planned beforehand with prepared questions. The meetings will either be physically or digitally on Teams based on needs of information and discussion.</li></ul>

# Meeting 5

mandag 16. januar 2023 13:13

Date	January 19th 2023
Place	On Campus
Participants	Johannes, Sondre, Sang and Ernst
Agenda	Talk about the project plan Do we have to use LaTeX for the project plan? Can we have meetings later?
Summary	<ul style="list-style-type: none"><li>• Discussed what we discussed on the meeting with Stig Atle on January 17th 2023</li><li>• Asked about the project plan, what is needed on the plan?<ul style="list-style-type: none"><li>• The task</li><li>• How we are going to solve it</li></ul></li><li>• Will make group rules</li><li>• Asked for feedback on the project plan and received advice<ul style="list-style-type: none"><li>• Document while making the product</li><li>• Combine two first rows of the project plan</li><li>• A little low minimum work hours? But choose yourself. (maybe 30?)</li><li>• Gantt-diagram: Research: gain an understanding on the whole picture of the task (technologies, etc.)</li><li>• Proof of concept: have a plan a design of the solution</li><li>• A wish from Ernst: before easter: send in an first draft with the 3 first chapters of the project report</li><li>• Don't spend time on writing the project report in LaTeX/overleaf, because we have come so far already</li></ul></li><li>• For next meeting with Ernst:<ul style="list-style-type: none"><li>• Sign standard agreement</li><li>• Ask about CI/CD pipeline</li></ul></li><li>• Work session<ul style="list-style-type: none"><li>• Worked on the project plan</li></ul></li></ul>



# Meeting 6

mandag 16. januar 2023 13:14

Date	January 20th 2023
Place	Teams
Participants	Johannes, Sondre and Sang
Agenda	Meeting about HDO remote access
Summary	<ul style="list-style-type: none"><li>• Discussed how to gain remote access to the VMs, try to set up Prometheus and testing</li><li>• Discussed test environment</li></ul>

# Meeting 7

torsdag 26. januar 2023 08:35

Date	January 26th 2023
Place	On Campus
Participants	Johannes, Sang and Ernst
Agenda	<ul style="list-style-type: none"><li>• Feedback on the project plan</li></ul>
Summary	<ul style="list-style-type: none"><li>• Will look at the project plan with us in the meeting</li><li>• Told Ernst that we got access to the testing environment, but not to the API yet</li> <li>• Layout comment<ul style="list-style-type: none"><li>• Remove unnecessary titles</li></ul></li><li>• Clearer task description</li><li>• Ta med avgrensninger i scope (ting vi hadde tenkt til å gjøre, men kommer ikke til å gjøre)</li><li>• Last sentence in delimitations, find out if we are going to do that or not</li><li>• Ernst thinks that the minimum hours per week is low, should be 30 hours per week</li><li>• Tools<ul style="list-style-type: none"><li>• Add something to delegate tasks</li></ul></li><li>• Comment about sprints<ul style="list-style-type: none"><li>• Write down the tasks somewhere</li></ul></li><li>• Add more to bibliography, references</li><li>• Add changes to milestones</li><li>• Change HDO's API to SBC Edge REST API</li> <li>• Fix the projectplan and send it to Ernst before next morning in teams (tag the channel in) or e-mail</li> <li>• Future meetings will now be every Thursday at 10:30</li></ul>

# Meeting 8

torsdag 2. februar 2023 10:31

Date	February 2nd 2023
Place	On campus
Participants	Johannes, Sondre, Sang and Ernst
Agenda	Discuss feedback on the project plan
Summary	<ul style="list-style-type: none"><li>• Make the CI/CD pipeline early on to utilize it, but prioritize the project</li><li>• Collect all metrics from the API and then filter or filter from the beginning?<ul style="list-style-type: none"><li>• Write something about the plan for the solution</li></ul></li><li>• Went through comments</li><li>• Effect goals, the result, not making the product, but to have the product and the effect of HDO having the solution</li><li>• Always refer to figures and refer first and then have the figure after the reference</li><li>• Avoid using personal pronouns</li><li>• Add user manual to result goals</li><li>• 5.4 write about what the different measures attempt to solve</li><li>• Begin with the final report as early as we want</li><li>• Discuss if we are going to use AI for the project</li><li>• Avoid references to YouTube videos, have references to Git code, try to refer the original source</li> <li>• Goal for the week<ul style="list-style-type: none"><li>• Fix the most important things in project plan</li><li>• Set up LaTeX for the final report</li><li>• Learn Go</li><li>• Give Ernst access to the meeting notes, send him an e-mail when it is ready</li><li>• REST Authentication</li></ul></li></ul>

# Meeting 9

søndag 5. februar 2023 23:34

Date	February 9th 2023
Place	On campus
Participants	Sondre and Sang and Ernst
Agenda	Regular supervisor meeting
Summary	<ul style="list-style-type: none"><li>• Ernst has received the minutes of meeting from last week</li><li>• REST Authentication</li><li>• Sondre has learned about Go and is making a simple exporter</li><li>• Difficulties with learning Go syntax in the beginning, two variables</li><li>• Haven't started on the bachelor report</li><li>• Don't wait until the end to start on the report</li><li>• Start learning LaTeX/Overleaf in the near future</li><li>• Use the mail from the general chat in the supervisor chat</li><li>• Discuss work with Johannes</li><li>• Plan for the week, learn a little bit of LaTeX and work on the exporter</li></ul>

# Meeting 10

onsdag 15. februar 2023 23:58

Date	February 16th 2023
Place	On Campus
Participants	Sang and Ernst
Agenda	Regular supervisor meeting
Summary	<ul style="list-style-type: none"><li>• Discussed team members</li><li>• Contact Johannes to see if everything is alright</li><li>• Adjust the project plan according to the needs of the group</li><li>• Sondre has worked on the exporter and Sang has worked on the bachelor thesis</li><li>• Plan ahead:<ul style="list-style-type: none"><li>• Work on the project plan, Sang will help Sondre after the proof of concept is finished</li></ul></li><li>• Bachelor report structure:<ul style="list-style-type: none"><li>• Introduction (high level and our task, about 4 pages), the reader should know what the is and how we will solve it</li><li>• Background/teori/liknede arbeid, gi informasjon om teknologier vi bruker, hvordan helsenettet ser ut, gi ekstra informasjon for at leseren skal skjønne løsningen, liknede arbeid (løsninger som likner på vår)</li><li>• Metode, hvordan løsningen er implimentert/test (noen få sider), metodikken</li><li>• Design av løsningen (overordnet), hvordan har vi designet løsningen, hvordan fungerer løsningen, hvilke komontenter består løsningen av?</li><li>• Hvordan har løsningen blitt implimentert, løsningen vi har lagd (kode), hvordan har vi lagd løsningen?</li><li>• Testning av løsningen</li><li>• Utfordringer eller svakheter med løsninger, videre arbeid, refleksjon</li><li>• Vi blir vurdert på rapporten, så alt utenfor (koden) "finnes ikke"</li></ul></li></ul>

# Meeting 11

fredag 17. februar 2023 02:52

Date	February 21st 2023
Place	Teams
Participants	Sondre, Sang and Stig Atle
Agenda	Statusmeeting
Summary	<ul style="list-style-type: none"><li>• Questions:<ul style="list-style-type: none"><li>• What kind of data is relevant to collect?</li><li>• Where to store the data?</li></ul></li><li>• We have created a simple exporter that reads from an XML-file and puts data into Prometheus</li><li>• Does the data change often?<ul style="list-style-type: none"><li>• Depends on the data, will look further into that later on another date</li><li>• Look at what data is relevant to collect</li></ul></li><li>• HDO can pull from GitHub</li><li>• We can get our own GitLab area if we need</li><li>• Check all boxes individually</li><li>• Have labels for each box</li><li>• One metric where we can see each individual box</li><li>• Storing data:<ul style="list-style-type: none"><li>• Store all data or store only selected data</li><li>• One curl query is better than multiple queries</li><li>• Stig Atle said do the easiest one (one query for all data)</li><li>• Query every 15 seconds (standard Prometheus)</li></ul></li><li>• Metrics reset when they reset the system</li><li>• Make a sanity check to see if data has been reset</li><li>• Physical meeting next week where we will discuss collection of data</li></ul>

# Meeting 12

torsdag 23. februar 2023 10:36

Date	February 23rd 2023
Place	On Campus
Participants	Johannes, Sondre, Sang and Ernst
Agenda	Regular meeting with supervisor
Summary	<ul style="list-style-type: none"><li>• Issues with VM and internet access, asked Stig Atle to grant access to internet on those VMs</li><li>• Can't download Golang packs and etc. because of no internet access on the VMs</li><li>• Sang has written a little bit on the introduction on the report, should not be more than 4-5 pages</li><li>• Sondre thinks about changing the exporter to read directly from the API call instead of an XML-files</li><li>• Johannes is back and will help Sang with the writing of the bachelor report</li><li>• About design, explain why we have chose the chosen design, and maybe about why we chose the chosen design over other ones</li><li>• Don't write about small bugs</li><li>• Best practices is relevant for the design chapter, keep security in mind</li><li>• If we find difficulties with the design that we would like to change, but change because of time constraints, we could write about what we would do instead in a "further work" chapter</li><li>• It is important in the report to explain the design and why we chose it</li><li>• The solution can be open source, will discuss with Stig Atle</li></ul>

# Meeting 13

torsdag 23. februar 2023 10:39

Date	February 27th 2023
Place	At HDO
Participants	Johannes, Sondre, Sang and Stig Atle
Agenda	Discuss exporter, metrics and design
Summary	<ul style="list-style-type: none"><li>• Should the solution be open source?<ul style="list-style-type: none"><li>• Yes it can</li></ul></li><li>• Reached milestone during the weekend<ul style="list-style-type: none"><li>• Cookie session</li><li>• API calls with Go</li></ul></li><li>• Stig Atle has uploaded the documentation for login on the SBCs</li><li>• He has made a document with relevant data to collect<ul style="list-style-type: none"><li>• Mostly integer values</li></ul></li><li>• Some resources are more important than others and should be prioritized</li><li>• Runtime instead of historical statistics</li><li>• Have a config file where HDO can put in IP-adresses and configure themselves</li><li>• We need to find identifiers dynamically</li><li>• Example: Prometheus fortigate exporter, config file</li><li>• Make a parameter for scraping intervals</li><li>• Everything may be gauge, Stig Atle has not found anything that are useful for counters because we can't know if anything has been reset or not</li><li>• Use names similar to the parameter names, such as rt_CPUUsage = System CPU Usage</li><li>• edge_{resourceName}_{typeOfData}</li><li>• We don't need to make alerts</li><li>• Open source: Stig Atle: the group decides</li></ul>



# Meeting 14

søndag 5. mars 2023 03:09

Date	March 9th 2023
Place	On Campus
Participants	Johannes, Sang and Ernst
Agenda	Regular meeting with supervisor
Summary	<ul style="list-style-type: none"><li>• Use "we" or "the group", use report/thesis<ul style="list-style-type: none"><li>• Use the group, avoid personal pronouns</li><li>• Avoid story, we did this and then this and then this</li></ul></li><li>• Write in past tense or future tense<ul style="list-style-type: none"><li>• Rule of thumb: when writing about something the product that has developed, use past tense: we have developed..., the report is going to be read after the project has been finished</li></ul></li><li>• Sondre has worked a lot on the exporter</li><li>• Sang has "finished" the introduction on the report</li><li>• Sondre has made a Docker file for testing</li><li>• What are we meant to write on the project scope?/domain/fagområde</li><li>• Johannes: group leader, needs to work, has not logged hours, lack of communication</li><li>• Futher work<ul style="list-style-type: none"><li>• Start on the background chapter</li><li>• Continue working on the exporter</li></ul></li><li>• Johannes will work on writing on the report or program from now on</li><li>• The 3 chapters, be clear on which chapters are finished and the ones that we know we are going to change</li><li>• Read the text that people have written in the report</li></ul>

# Meeting 15

mandag 20. mars 2023 16:45

Date	March 23rd 2023
Place	On Campus
Participants	Sondre, Sang and Ernst
Agenda	Regular meeting with supervisor and discuss situation with Johannes
Summary	<ul style="list-style-type: none"><li>• Work on the Exporter is going well</li><li>• How much code should we comment?<ul style="list-style-type: none"><li>• We should first comment on how the system works overall (overordnet)</li><li>• And then we can comment on the code that makes up each component</li><li>• Write about the technologies we use (SQLite, Golang, HTTP, etc.)</li><li>• Details about the programming language (Golang), routes, channels etc.<ul style="list-style-type: none"><li>◦ Write about them and connect it to what we have made as an example</li></ul></li></ul></li><li>• The product is a prototype and therefore it's not so important to do it in an elegant way, but we can write about other ways we could have done things</li><li>• Deliver some content to Ernst before easter (2 april), write what will be changed and what is finished, the goal is to deliver 3 finished chapters</li><li>• Find out what's going on with Johannes, contact him, call him on Friday</li><li>• If we can't contact him by the end of this week, then Ernst will help</li><li>• Have a chapter about the security about our product towards the end of the report</li><li>• Discuss what we can share and what we can't share with HDO (in terms of sensitive information)</li></ul>

# Meeting 16

tirsdag 28. mars 2023 11:58

Date	March 28th 2023
Place	On Campus
Participants	Johannes, Sang and Ernst
Agenda	Meeting with supervisor about group issues
Summary	<ul style="list-style-type: none"><li>• Discussed the situatuon with Johannes</li><li>• The group should have meetings with each other more often to delegate tasks and have status meetings</li><li>• Make a plan for what he group will do in the future</li><li>• Have a meeting tomorrow where the group discusses the plan for the future</li></ul>

# Meeting 17

onsdag 29. mars 2023 15:25

Date	March 29th 2023
Place	On Discord
Participants	Sondre, Sang and Johannes
Agenda	Discussion about the group issues
Summary	<ul style="list-style-type: none"><li>• Recap of the meeting with Ernst yesterday</li><li>• 2 meetings per week, more towards the end if it is needed<ul style="list-style-type: none"><li>• Smaller meetings when it's needed</li><li>• Mondays and Thursdays</li></ul></li><li>• Showcase of Sondre's code</li><li>• Make a file with tasks</li><li>• Drop pipeline</li><li>• Difference between design and implementation</li><li>• Requirements, as it they were at the start or as they are after consultation</li></ul>

# Meeting 18

onsdag 29. mars 2023 15:25

Date	March 30th 2023
Place	On Campus
Participants	Sondre, Sang, Johannes and Ernst
Agenda	Regular meeting with supervisor and a short work session
Summary	<ul style="list-style-type: none"><li>• More regular meetings</li><li>• We have made a Kanban/to-do list</li><li>• Don't spend too much time on explaining the details about the exporter, it may be time consuming</li><li>• Sondre says that the exporter should be finished one week after easter</li><li>• He finished a collector yesterday, but is stuck on a collector because of missing test data</li><li>• If implementing all of the collectors will take too much time, then the group can pick out a few collectors and make a proof of concept of a few of them</li><li>• Drop the pipeline, but the group can write a couple of sentences of how they would've implemented the pipeline instead of spending time implementing it (videre arbeid/discussion)</li><li>• Avgrensning: si at vi ikke skal ta oss av hva hver metrikk betyr</li><li>• Requirements: requirements shouldn't be changed, but groups may adjust it to better fit with what they have made (shouldn't do this)</li><li>• Background: what the reader needs to know to understand the report/project</li><li>• Eget kapittel: kartlegging av hvilke dataer som kan hente ut</li><li>• Skrive i design at vi diskuterte med oppdragsgiver om hvilke dataer som skal hentes ut</li><li>• After background: kravspesifikasjon</li><li>• The most important thing is to keep writing and get feedback from Ernst</li><li>• We should talk about the security of the product<ul style="list-style-type: none"><li>• Passwords in plaintext: write about different ways we would've done it, we can write it in the design or in a different discussion/security chapter</li></ul></li><li>• Talk to Stig Atle about if it's okay that we write about the plaintext password situation in the report</li><li>• The product is a proof of concept, we don't have to make things perfect for a production environment</li><li>• Security: discuss what the strengths and weaknesses are of our solution</li><li>• Do research about the security of the technologies that we use</li><li>• Use we/the group, do what feels natural, but keep in mind that</li></ul>

"we" is not really scientifically "correct"

- Try to understand Ernst's comments and agree on them before changing things in our report
- Write what we have polished and think is done and what's not finished
- References can be at the start of a section and be enough for the whole section
- Give the reader the option to read further material if they're interested

# Meeting 19

tirsdag 11. april 2023 13:57

Date	April 11th 2023
Place	Teams
Participants	Sondre, Sang, Johannes and Stig Atle
Agenda	Questions about the task
Summary	<ul style="list-style-type: none"><li>• Questions around the signaling groups and other questions about the exporter</li><li>• Next meeting on the 25th of April</li></ul>

# Meeting 20

onsdag 12. april 2023 23:06

Date	April 13th 2023
Place	On Campus
Participants	Sang, Johannes and Ernst
Agenda	Regular meetings with supervisor
Summary	<ul style="list-style-type: none"><li>• Questions<ul style="list-style-type: none"><li>• Find more primary sources? Preferably primary sources over wikipedia</li><li>• What to do if you need to reference the same source more than once? Use a main reference early on, use references on special påstands<ul style="list-style-type: none"><li>◦ Same source but on a different page? Don't do this, just the whole article is enough</li></ul></li><li>• How to reference sources? BibTeX? Not very important, but BibTeX is good</li><li>• How should we add links to the references? Check wikipedia and the websites on how to reference</li></ul></li><li>• Sang and Johannes will work on the comments this week and next week we will work on the design and implementation</li><li>• Sondre will work on the exporter until the 20th of April</li><li>• Discussed what we have time to do and what we might drop with Stig Atle</li></ul>



# Meeting 21

mandag 17. april 2023

15:35

Date	April 20th 2023
Place	On Campus
Participants	Sang, Johannes and Ernst
Agenda	Regular meeting with supervisor
Summary	<ul style="list-style-type: none"><li>• Questions<ul style="list-style-type: none"><li>◦ Copyright rules<ul style="list-style-type: none"><li>◦ Make your own figures if possible</li><li>◦ Check if you are allowed to use it (copyright)</li><li>◦ Check Innsida for copyright rules</li><li>◦ Make it very clear that you have gotten it from elsewhere: "from: …"/"inspired by"</li></ul></li><li>• HDO in english, use norwegian name or translate it?<ul style="list-style-type: none"><li>◦ Make it clear that it's a name, write it in cursive</li><li>◦ Ask HDO about it</li><li>◦ Should have it in english</li></ul></li><li>• Related work, mention fortigate exporter, talk about what makes our solution different?<ul style="list-style-type: none"><li>◦ Fortigate exporter is an implementation, not exactly related work, related work would be more about a general monitoring system (not exactly an exporter)</li><li>◦ Move it to the background/design chapter instead of having a separate related work chapter, write just a few pages (1-3) about the solution</li><li>◦ Not necessary to write about it if we haven't used it that much</li><li>◦ Ernst says it sounds like our solution vs. Fortigate exporter fits better in the design chapter, we can write about our solution and alternatives such as the fortigate exporter, can also write about best practice principles that we have chosen to follow</li></ul></li><li>• First chapter figure<ul style="list-style-type: none"><li>◦ Make a very simple figure that illustrates the concept in a simple way, include the core network and the sbc edges</li><li>◦ Use a map over Norway and include sbc edges?</li></ul></li><li>• Sources, have we done it right? When to use sources and when to use footnotes<ul style="list-style-type: none"><li>◦ If we are unsure, then always use sources</li><li>◦ Use footnotes for things that are not exactly statement, for example if we just mention something</li><li>◦ If something is obvious, then we don't have to include a source</li></ul></li><li>• Include readme file in the report?</li></ul></li></ul>

- Installation, VM/HDO can be included in a chapter called something like "use"
  - Add the code as a zip with the report, but write the report well so that the reader doesn't have to open up the code to understand the project
  - Also ask HDO if they think it's okay to share information about the code and their infrastructure
  - Packs implementasjon, separate the implementation and write about each function?
    - Sondre will not write about each function, but about the most important ones
    - Make it readable and easy for the reader to read
    - Move the section about Golang into chapter 2
  
  - A common worry is to have too few pages, but it usually ends up with being too much and difficult for the reader to read. So write short and simple text.
  - Don't make drastic changes to our solution (database)
  - Further plan: finish writing chapters and send it to Ernst for review
- Move the 1.1 figure to design?

# Meeting 22

torsdag 20. april 2023 11:56

Date	April 26th 2023
Place	At HDO and Teams
Participants	Sondre, Sang, Johannes and Stig Atle
Agenda	Meeting with employer
	<ul style="list-style-type: none"><li>• Ask about the chapter 1 simple figure</li><li>• Ask about HDO name in english<ul style="list-style-type: none"><li>• Stig Atle will check some english documents for an official english name for HDO</li><li>• Use the norwegian name, they don't use an english name, it's the juridical name for HDO</li></ul></li><li>• Security concerns<ul style="list-style-type: none"><li>• Passord i klartekst<ul style="list-style-type: none"><li>◦ Not optimal, could make a hash<ul style="list-style-type: none"><li>▪ The API does not support any other way of authentication</li><li>▪ It is atleast transfered over cryptated channels</li><li>▪ The reward is small if they find something</li><li>▪ The only thing they can do is to destroy one AMK</li><li>▪ Don't have the same password on the same users</li></ul></li><li>◦ The SBC needs the password to work</li><li>◦ It is possible for someone to destroy stuff, but it is not very likely and there are other things that are more likely to be interesting for the attackers</li><li>◦ Only thing they can do is to delete the config file, the user is a local user on one SBC</li></ul></li><li>• Exporter kjører i HDO sine vm'er, er det trygt?</li><li>• Kan exporterne spamme/overarbeide SBC-ene/infrastrukturen?<ul style="list-style-type: none"><li>◦ Yes there is a possibility, Stig Atle has made the SBCs crash before, they might crash with the exporter because of very many HTTP calls</li></ul></li><li>• Hvordan ser infrastrukturen ut igjen? Hvordan henger SBC sammen med kjernenettverket?</li><li>• Noen tips på hva vi kan skive om med tanke på sikkerhet?<ul style="list-style-type: none"><li>◦ Vurdere sikkerheten rundt koden vår, Docker (vanlige sårbarheter), vudere at eksporteren skal kjøre i et lukket miljø (mikrosegmentert), alt er stengt by default, brannmur, mye sikkerhet i nettverket, zero trust nettverk, database?</li></ul></li></ul></li></ul>

- Status update on the exporter, stopped working on the exporter to write on the project, might have time to finish the exporter later
- Make a document that describes which datas that are retrieved, why they are retrieved and what value they bring
  - What data is retrieved from what URL
- Write about the already existing monitoring system and how the exporter will be integrated with the current solution
  - Stig Atle will make a simple sketch about the infrastructure
- Stig Atle is not allowed to tell us where the SBC's are located in Norway
- Write about the YAML file in the report as an attachment
- Publish image and code on Docker Hub for easier deployment for HDO
- Send image to Stig Atle for testing, might need to fix tweak some things (labels, etc.), he will give feedback
- Make a gauge about success and fails when connecting to the API
- Make a check if the host is reached, if it fails then cancel the rest of the requests
- Drop the boot partition collector

# Meeting 23

onsdag 26. april 2023 19:55

Date	April 26th 2023
Place	Discord
Participants	Sondre, Sang and Johannes
Agenda	Status meeting
Summary	<ul style="list-style-type: none"><li>• Discusses content that Johannes could work on</li></ul>

# Meeting 24

torsdag 27. april 2023 01:31

Date	April 27th 2023
Place	On Campus
Participants	Sondre, Sang and Johannes
Agenda	<ul style="list-style-type: none"><li>• Discuss situation with Johannes</li></ul>
Summary	<ul style="list-style-type: none"><li>• Made a list of what Johannes can work on</li><li>• Group has made a deadline for Johannes for Monday to prove that he can contribute in a significant way</li> <li>• Ernst says that communication is very important now<ul style="list-style-type: none"><li>• Have meetings often (physical if possible) to discuss and work together</li></ul></li></ul>

# Meeting 25

torsdag 27. april 2023 20:09

Date	April 27th 2023
Place	Discord
Participants	Sondre, Sang and Johannes
Agenda	<ul style="list-style-type: none"><li>• Group meeting</li><li>• Discuss work</li></ul>
Summary	<ul style="list-style-type: none"><li>• Make a list of metrics that are collected</li></ul>

# Meeting 26

fredag 28. april 2023 18:49

Date	May 2nd 2023
Place	Discord
Participants	Sondre, Sang and Johannes
Agenda	<ul style="list-style-type: none"><li>• Group meeting</li><li>• Discuss work</li></ul>
Summary	<ul style="list-style-type: none"><li>• Discussed work</li><li>• Try to read implementation and send it to Ernst</li></ul>



# Meeting 27

tirsdag 2. mai 2023 20:08

Date	May 3rd 2023
Place	Discord
Participants	Sondre, Sang and Johannes
Agenda	Regular group meeting
Summary	<ul style="list-style-type: none"><li>• Discussed API Tokens and hashing</li><li>• Discussed questions about some metrics</li><li>• Discussed how to divide the sections in chapter 4</li><li>• More psdeuocode in chapter 7?, it's very long and difficult to understand</li></ul>

# Meeting 28

tirsdag 2. mai 2023 20:12

Date	May 4th 2023
Place	On Campus
Participants	Sondre, Sang, Johannes and Ernst
Agenda	Regular meeting with supervisor
Summary	<ul style="list-style-type: none"><li>• List of data, put it in the report or as vedlegg?</li><li>• List of data what format? List/etc.?</li><li>• How far have you come with the report?<ul style="list-style-type: none"><li>• About 70% finished</li></ul></li><li>• Read each others text</li><li>• Make more figures in implementation for easier understanding</li><li>• Exchange reports with other groups to get some useful feedback</li><li>• What can we say about the availability of the data?</li><li>• Have the list of data as an attachment OR have a chapter about data of each collector in terms of<ul style="list-style-type: none"><li>• Availability</li><li>• Why they are collected</li><li>• What value they bring</li></ul></li><li>• Write about the metrics in terms of what type they are, try to group them together?</li><li>• Flytte "Data" til etter "Implementation"</li><li>• Skrive om hvilken rolle Edge boksene har, hvorfor er de koblet til core, hvorfor går ikke samtale rett fra core til AMK-ene?, hva skjer hvis en ambulanse kommuniserer med en AMK?, bruker de også Telenor?</li><li>• Spørre Stig Atle om et møte for oppklaring om arkitekturen, hva gjør SBC Core og hva gjør SBC Edge, hvorfor trenger vi begge?</li></ul>

# Meeting 29

torsdag 4. mai 2023 15:40

Date	May 5th 2023
Place	Teams
Participants	Sang, Sondre, Johannes and Stig Atle
Agenda	Questions about the architecture and exporter
Summary	<ul style="list-style-type: none"><li>• What can we say about the availability of the data?<ul style="list-style-type: none"><li>• Hensive til dokumentasjonen av hvilke data som kan hentes ut</li></ul></li><li>• Skrive om hvilken rolle Edge boksene har, hvorfor er de koblet til core, hvorfor går ikke samtalene rett fra core til AMK-ene?, hva skjer hvis en ambulanse kommuniserer med en AMK?, bruker de også Telenor, spørre Stig Atle om et møte for oppklaring om arkitekturen, hva gjør SBC Core og hva gjør SBC Edge, hvorfor trenger vi begge?<ul style="list-style-type: none"><li>• SBC Core (6 stk totalt, interconnect, NNI, kobling til andre nettverk Telia, Telenor, osv.) sender til riktig SBC Edge/lokasjon</li><li>• AMK snakker ikke SIP/IP, SBC Edge konverterer fra SIP/IP til ISDN</li><li>• Ambulanse (AMK) bruker radio, figuren Stig Atle ga oss snakker i hovedsak om 113 og 116117</li><li>• Sykehus ringer mellom seg og HDO er ikke involvert</li></ul></li><li>• What value do these metrics bring?<ul style="list-style-type: none"><li>• CPU, memory is monitoring of hardware<ul style="list-style-type: none"><li>◦ The point is to check if the hardware is okay, or what happend when an incident occurred</li></ul></li><li>• Monitoring of other data is to find trends and acting before an incident occurs</li></ul></li><li>• Has HDO tried the newest version of the exporter?<ul style="list-style-type: none"><li>• No, will test soon</li></ul></li><li>• Write about what we can be done better in the further work chapter</li><li>• Problems with the CPU with the SBC Edges</li><li>• Fix introduction according to the new information and send it to Ernst</li><li>• SBC Core har en Prometheus exporter/script</li></ul>

# Meeting 30

mandag 8. mai 2023 16:26

Date	May 8th 2023
Place	Discord
Participants	Sondre, Sang and Johannes
Agenda	Regular group meeting
Summary	<ul style="list-style-type: none"><li>• Questions<ul style="list-style-type: none"><li>• DDoS?<ul style="list-style-type: none"><li>◦ It's possible to connect to the SBC's and spam them with traffic</li></ul></li><li>• Security of code?<ul style="list-style-type: none"><li>◦ Port numbers may be hacked in some way</li></ul></li><li>• SBC does not tolerate much load<ul style="list-style-type: none"><li>◦ Check development process</li></ul></li><li>• Check package exploits</li></ul></li><li>• Write about the two versions of the implementation</li></ul>

# Meeting 31

torsdag 11. mai 2023 10:33

Date	May 11th 2023
Place	On Campus and Teams
Participants	Sondre, Sang, Johannes and Ernst
Agenda	Regular meeting with supervisor
Summary	<ul style="list-style-type: none"><li>• Will read our report today after 3pm</li><li>• Can't read the report as detailed as back in easter</li><li>• The group will send Ernst and updated version before 3pm today, for review</li><li>• Sondre is worried about repeating himself in the report (3 times)<ul style="list-style-type: none"><li>• Do this? As described in x.x, y is z.</li></ul></li><li>• Don't make the reader read the same thing again unnecessarily</li><li>• Misplaced figures: try [htbp]<ul style="list-style-type: none"><li>• Or move figures in the text so that they get placed where we want</li></ul></li><li>• Johannes thinks that the code listings in the implementation are too long<ul style="list-style-type: none"><li>• Sondre thinks that they are important for understanding how the product works</li><li>• Ernst suggests to break the code up and write about sections of the code to avoid the reader losing the red thread</li><li>• Avoid too much code so that the reader questions why they are there and what function they have</li><li>• Avoid explaining every line in the code listings</li><li>• Add comments in the code to explain the main points</li></ul></li><li>• The report should not be longer than 80 pages</li><li>• Try to make the report shorter instead of writing for the sake of adding unneeded content</li></ul>

# Meeting 32

mandag 15. mai 2023 15:33

Date	May 15th 2023
Place	Discord
Participants	Sang and Johannes
Agenda	Regular group meetings
Summary	<ul style="list-style-type: none"><li>• Gotten feedback from Ernst</li><li>• This week we will review the comments and finish the report.</li><li>• Prepare questions for Ernst on Thursday</li><li>• Change the sequence in the discussion chapter</li></ul>

# Meeting 33

mandag 15. mai 2023 16:30

Date	May 16th 2023
Place	Discord
Participants	Sang, Sondre and Johannes
Agenda	Regular group meetings
Summary	<ul style="list-style-type: none"><li>• Divided workload</li></ul>

# Meeting 34

tirsdag 16. mai 2023 16:58

Date	May 18th 2023
Place	Discord
Participants	Sondre, Sang and Johannes
Agenda	Group meeting
Summary	<ul style="list-style-type: none"><li>• Discussed how are we doing?</li><li>• Divided workload</li><li>• Fått svar fra Stig Atle om Grafana screenshots?<ul style="list-style-type: none"><li>• Not yet</li></ul></li><li>• Sondre will fix Appendix B</li><li>• Fixed readme file</li></ul>



# Meeting 35

fredag 19. mai 2023 05:16

Date	May 19th 2023
Place	Discord
Participants	Sang, Sondre and Johannes
Agenda	Regular group meeting
Summary	<ul style="list-style-type: none"><li>• Ask about some comments</li><li>• Finish the report today, read through everything on Saturday and deliver</li><li>• Answer from Stig Atle?<ul style="list-style-type: none"><li>• Yes, will add screenshots to the thesis</li></ul></li></ul>

# Meeting 36

lørdag 20. mai 2023 18:36

Date	20th May 2023
Place	Discord
Participants	Sang, Sondre and Johannes
Agenda	Group meeting
Summary	<ul style="list-style-type: none"><li>• Discussed what is left to finish</li><li>• Finishing up the thesis</li></ul>

# Meeting 37

søndag 21. mai 2023 00:04

Date	21th May 2023
Place	Discord
Participants	Sang, Sondre and Johannes
Agenda	Group meeting
Summary	<ul style="list-style-type: none"><li>• Finishing up the thesis</li></ul>



## **Appendix H**

# **Time Tracking**

This appendix includes the time tracking for the project.



Time Tracking in Hours			
Month	Johannes	Sang	Sondre
January	36	59	45,5
February	14	69	75,5
March	27	83	111
April	80	107	108
May	81	96	57,5
Total	238	414	397,5

**Johannes**

<b>Date</b>	<b>Hours</b>	<b>Details</b>
13.jan	5	Meeting and research
15.jan	3	Writing project plan
16.jan	4	Writing project plan
17.jan	3	Meeting with employer
19.jan	4	Meeting with supervisor and work session
20.jan	4	Working on project plan
26.jan	6	Meeting with HDO + research
27.jan	3	Research
30.jan	4	Research
02.feb	2	Meeting with Ernst + bachelor report
23.feb	2	Meeting with Ernst + group meeting
24.feb	4	Research + learning docker
27.feb	3	Meeting with Stig Atle
28.feb	3	learning Docker
02.mar	1	Meeting with Ernst
07.mar	3	Learning Go
08.mar	3	Learning and setting up Go
09.mar	3	Meeting with Ernst + research
10.mar	3	Setting up vm, testing exporter
11.mar	4	Research + Bachelor report
28.mar	2	Meeting with Ernst + group meeting
29.mar	1	Group meeting
30.mar	3	2 meetings
31.mar	4	Bachelor report
02.apr	3	Bachelor report
03.apr	2	Bachelor report
06.apr	4	Bachelor report
11.apr	4	Meeting with HDO and bachelor report
13.apr	6	Meeting with Ernst and bachelor report
14.apr	5	Bachelor report
15.apr	5	Bachelor report
17.apr	3	Group meeting
18.apr	5	Bachelor report
19.apr	1	Bachelor report
20.apr	4	Meeting with Ernst + bachelor report
25.apr	6	Bachelor report
26.apr	2	Meeting with HDO + group meeting
27.apr	10	Meeting with Ernst + group meeting + bachelor report
28.apr	1	Bachelor report
29.apr	9	Bachelor report
30.apr	10	Bachelor report
01.mai	2	Bachelor report
02.mai	7	Group meeting + bachelor report
03.mai	7	Group meeting + bachelor report
04.mai	6	Meeting with supervisor + bachelor report
05.mai	7	Meeting with employer + bachelor report
08.mai	4	Meeting + bachelor report



11.mai	5 Meeting with supervisor + bachelor report
15.mai	4 Group meeting + bachelor report
16.mai	8 Group meeting + correcting the report
18.mai	9 Group meeting + correcting the report
19.mai	8 Group meeting + correcting the report
20.mai	8 Finishing up the thesis
21.mai	6 Finishing up the thesis

**Sang**

<b>Date</b>	<b>Hours</b>	<b>Details</b>
13.jan	5	Meeting and reading previous thesises
14.jan	4	Reading and writing projectplan
15.jan	3	Writing project plan and group meeting
16.jan	4	Writing project plan
17.jan	3	Meeting with employer
18.jan	2	Writing project plan
19.jan	4	Meeting with supervisor and work session
20.jan	3	Reading through the project plan for quality assurance
23.jan	2	Research
24.jan	2	Research and trying to access VMware
25.jan	6	Meeting with HDO and Prometheus
26.jan	6	Meeting with Ernst, project plan and Docker
27.jan	4	Docker, prometheus and dialogue with HDO
29.jan	2	Docker
30.jan	4	Docker, prometheus and dialogue with HDO
31.jan	5	Docker, prometheus, dialogue with HDO and REST API authentication
01.feb	5	REST API Authentication
02.feb	5	Meeting with Ernst, project plan and REST API Authentication
03.feb	3	Project Plan and REST API Authentication
04.feb	4	REST API Authentication and dialogue with HDO
05.feb	2	Testing with REST API
06.feb	5	Researching exporters
09.feb	3	Meeting with Ernst and studying LaTeX/Overleaf
10.feb	2	Starting on the bachelor report in Overleaf
12.feb	2	Bachelor report
13.feb	3	Studying LaTeX
14.feb	2	Bachelor report
15.feb	2	Research exporters and Go
16.feb	4	Meeting with Ernst and Bachelor report
17.feb	2	Bachelor report
19.feb	5	Bachelor report
20.feb	4	Bachelor report
21.feb	2	Meeting with Stig Atle and bachelor report
22.feb	3	Bachelor report
23.feb	2	Meeting with Ernst and group meeting
24.feb	2	Bachelor report
27.feb	3	Meeting with Stig Atle and bachelor report
28.feb	4	Bachelor report
01.mar	4	Bachelor report
03.mar	1	Bachelor report
05.mar	3	Bachelor report
06.mar	2	Bachelor report
07.mar	4	Bachelor report
08.mar	3	Bachelor report
09.mar	4	Meeting with Ernst and reading background material
10.mar	4	Bachelor report
14.mar	2	Reading background material

15.mar	2 Dialogue with Ernst and bachelor report
16.mar	6 Bachelor report
19.mar	4 Bachelor report
20.mar	3 Bachelor report
21.mar	3 Bachelor report
22.mar	5 Bachelor report
23.mar	2 Meeting with Ernst
24.mar	5 Bachelor report
26.mar	6 Bachelor report
27.mar	4 Bachelor report
28.mar	2 Meeting with Ernst
29.mar	2 Group meeting and bachelor report
30.mar	6 2 meetings, writing background
31.mar	6 Bachelor report
02.apr	3 Bachelor report
03.apr	8 Bachelor report
06.apr	1 Bachelor report
11.apr	5 Meeting with HDO and bachelor report
12.apr	3 Bachelor report
13.apr	8 Meeting with Ernst and Bachelor report
14.apr	7 Bachelor report
15.apr	1 Bachelor report
16.apr	7 Bachelor report
17.apr	4 Meeting and bachelor report
18.apr	4 Bachelor report (adding figures)
19.apr	4 Bachelor report (adding figures)
20.apr	4 Meeting with Ernst and bachelor report
21.apr	6 Bachelor report
23.apr	7 Bachelor report
24.apr	6 Bachelor report
25.apr	8 Bachelor report
26.apr	6 Meeting with HDO and bachelor report
27.apr	4 Meeting with Ernst, group meeting, bachelor report
28.apr	6 Bachelor report
30.apr	5 Bachelor report
01.mai	6 Bachelor report
02.mai	7 Group meeting and bachelor report
03.mai	7 Group meeting and bachelor report
04.mai	6 Meeting with supervisor and bachelor report
05.mai	7 Meeting with employer and bachelor report
07.mai	2 Bachelor report
08.mai	4 Group meeting and bachelor report
09.mai	2 Bachelor report
10.mai	3 Bachelor report
11.mai	3 Meeting with supervisor and bachelor report
15.mai	4 Group meeting and bachelor report
16.mai	8 Group meeting and correcting the report
17.mai	6 Correcting the report
18.mai	10 Correcting the report and group meeting
19.mai	8 Correcting bachelor report and group meeting

20.mai

7 Finishing up the thesis and group meeting

21.mai

6 Finishing up the thesis and group meeting

**Sondre**

<b>Date</b>	<b>Hours</b>	<b>Details</b>
12.jan	2	Research and some writing
13.jan	5	Meeting and research
15.jan	2	Writing project plan
16.jan	4	Writing project plan
17.jan	3	Meeting with employer
18.jan	1	Project plan
19.jan	4	Meeting with supervisor and work session
25.jan	5,5	Meeting with HDO and research
26.jan	5	Project plan and research
27.jan	3	Research and test of a simple prometheus setup
30.jan	6	Writing a new grafana template and research
31.jan	5	Debugging prometheus/docker, dialogue with HDO
01.feb	3,5	Studying go and api testing
02.feb	5	Meeting with Ernst, studying go and a tutorial
03.feb	1,5	Fixing a model
04.feb	4	Writing on project plan, testing API and studying exporters
06.feb	2,5	Studying go
07.feb	6,5	Studying go and exporters
08.feb	5	Scraping one metric from file to variables with regex
09.feb	4,5	Meeting with Ernst, writing on thesis and studying docker
10.feb	2	Made go file read from XML
12.feb	3	Authentication issues gitlab, migrated to github
15.feb	3	Setting up ubuntu vm on personal desktop for easier workflow
16.feb	7	Configuring ubuntu vm, working on exporter, dialogue with hdo
20.feb	6	Small exporter ready, studying prometheus and design options
21.feb	2	Meeting with Stig Atle and studying metric types
22.feb	4	Working on exporter, writing a test, awaiting response from atle
23.feb	3	Meeting with Ernst, working on exporter
25.feb	5	Wrote code making API call with golang
27.feb	7	Meeting with Stig Atle, studying config/yml package, working on systemexporter
28.feb	1	Writing lecture with Frode
01.mar	8,5	Completing first exporter and api call functions
03.mar	7	systemExporter, functions for designproblems, bachlor's thesis
04.mar	4,5	Writing on thesis, building a chart
05.mar	8	Minimum viable product ready, systemExporter collecting multiple hosts
06.mar	3	editing chart,coding
07.mar	4	Studying and adding config package, dockerfile
08.mar	4	Config, cleaning git, dialogue with hdo
09.mar	6	Errorhandling, testing, errhandling as gauge value
10.mar	3	Testing routingentry api, dialogue with stig atle and coding
11.mar	1	Successful test with multiple working ip
13.mar	6	Successful use of config file for auth and getip, started on new exporter
14.mar	5	Testing routingentry api, dialogue with stig atle and coding
15.mar	2	Working on routingentry exporter
16.mar	4,5	Routingentryexporter, studying docs
17.mar	4	coding and studying, improving functions
19.mar	2	coding and studying, improving functions

20.mar	10 Routing collector
21.mar	6 Successful sqlite use to compare time of auth and retrieve last cookie if in time
22.mar	8 Sqlite done, last exporters to go and cleaning up the code layout
23.mar	1 Meeting with Ernst,
27.mar	2,5 Writing a bit on background and design
29.mar	5 Group meeting, made a new collector, need help from atle for next collector
30.mar	4 2 meetings, working on requirements
31.mar	2 research security and docker
01.apr	7 hardware collector, trying to store more sqlite data
02.apr	6 Bachelor report
03.apr	6 Exporter work
04.apr	2 Exporter work
08.apr	6 Cleaned repository and implemented packages for cleaner structure
09.apr	1 Preparing for the meeting with stig
10.apr	7,5 Made database for routingentries and tables, additional 2 sek improvement
11.apr	2 Meeting with Stig Atle and working on exporter
12.apr	8 Chassis collector and dialogue with stig atle and fixing a stubborn bug
13.apr	3 New collector and drawing
15.apr	7 Ethernetport collector and fixing some bugs
16.apr	5 Improving exporter and finishing dockerfile
17.apr	7 Meeting, drawing, new collector, improving code
18.apr	2,5 Bachelor report
19.apr	3 Bachelor report and studying go
20.apr	5,5 Meeting and bachelorreport
21.apr	2 Bachelor report
24.apr	2 Bachelor report
25.apr	2,5 Bachelor report
26.apr	5 Meeting, developing, docker upload
27.apr	8,5 Meeting with Ernst and exporter, docker
28.apr	5,5 Exporter and bachelor report
30.apr	4 Bachelor report
02.mai	2 Group meeting
03.mai	2 Group meeting
04.mai	6 Meeting with Ernst and Bachelor report
05.mai	7,5 Meeting with HDO, bachelor report and improved exporter design
08.mai	4 Meeting and bachelor report
09.mai	3,5 Bachelor report
10.mai	2 Bachelor report
11.mai	1 Meeting with Ernst
12.mai	1,5 Tidying exporter
15.mai	5 Bachelor report
16.mai	Group meeting
17.mai	6 Bachelor report
18.mai	2,5 Group meeting and bachelor report
19.mai	5 Bachelor report and group meeting
20.mai	5,5 Bachelor report and group meeting
21.mai	4 Bachelor report and group meeting



 **NTNU**

Norwegian University of  
Science and Technology