INS | Institute for Network and Security

OST Ostschweizer Fachhochschule

Secure Device Provisioning Using SZTP

Department of Computer Science

OST – University of Applied Sciences

Campus Rapperswil-Jona

Autumn Term 2023

Authors	Vanessa Gyger | Patrick Lenherr
Advisor	Urs Baumann
Co-Advisor	Yannick Zwicker
Project Partner	INS

Abstract

Initial Situation

Managing many network devices takes a lot of effort and poses risks of inconsistency in configuration. Furthermore, time is needed to plug the device in, attach your computer and connect to the console for configuration. With automation, this process can be made much more efficient and reliable. The classical approach for this is called Zero Touch Provisioning (ZTP), meaning the device doesn't have to be touched to configure it. Instead, it is registered in an inventory and can load a predefined configuration automatically.

Objective

The objective of this term project is to lay the groundworks for a network controller that handles zero touch provisioning of newly installed devices, as well as transferring configuration in case of 1-to-1 device replacement. The scope of the controller is limited to Cisco devices for this project, but it should be extendable to support various other devices. Furthermore, the controller should also be open for future extension, providing more functionality like ongoing configuration after the provisioning process.

Result

The authors developed a controller that bridges between inventory management and network device. The application is able to provision devices even in insecure environments. On one end Netbox is used to manage device parameters and context-dependent configuration. Both are rendered into the target devices configuration using a template. On the other end the devices use DHCP to get SZTP (Secure ZTP) redirect information for the controller. The provisioning controller provides an endpoint for SZTP-compliant devices to securely get the data needed to bootstrap themselves via HTTPS. The bootstrapping data includes target firmware version, download source and integrity hash as well as the configuration itself. During the devices lifetime our backup controller is used to automatically retrieve configuration backups. Nornir and Napalm are used to run the backup task on all devices registered in Netbox. In the event of a hardware failure, a replacement device can quickly be set up. By simply setting the configuration source device in Netbox the provisioning controller will automatically load the backup configuration.

Acknowledgments

We would like to thank our project advisors Urs Baumann and Yannick Zwicker, for their continued support and guidance.

We are also appreciative of Jonas Meier for his assistance by proofreading our work.

Lastly, we are grateful for our patient friends, colleagues, family members and supporting employers.

Contents

1. Management Summary

Baseline

In the field of network infrastructure managing devices an always present task. The management during the devices normal operation is automated by many organizations using either a vendor specific automation solution or one of the many available open-source tools supporting a variety of different vendors. One time-consuming aspect of device management is often overlooked, however: The devices have to be configured initially so that the automation tools can connect to them. Moreover, to enable more efficient deployment of devices, it should be possible to connect them in their final location without having to touch their configuration interface beforehand. Since this location is not always guaranteed to be in a trusted environment, both the device itself and the system doing the provisioning should be able to authenticate one another.

Approach / Technologies

In a first step, the available systems and protocols for Zero Touch Provisioning were compared. After initially planning to work with Cisco's PnP (Plug-and-Play), the more standardized SZTP (Secure Zero Touch Provisioning) was chosen. The choice aims at being extendable to work with devices from other vendors as well. Due to the lack of well working complete systems, the decision was made to develop an own application following the SZTP standard. The application is written in Python, using Flask as a web server for the SZTP endpoints. For the inventory Netbox is used. It also provides the application with the function of rendering configuration templates directly from the stored configuration settings. The backup controller which creates configuration backups from the physical devices is written in Python and composed of the automation framework Nornir with an inventory plugin to access Netbox and the Napalm library for device connections.

Results

The result of this project consists of a Python application with provisioning controller and backup controller. While the provisioning controller takes care of providing the devices with an extendable initial configuration, the backup controller can be used to automatically create configuration backups from all the devices. These backups can in turn be automatically restored on a replacement device using the provisioning controller. Both controllers interface with Netbox as the inventory solution. A user guide not only shows how to setup and the application but also describes how to configure Netbox to automatically consolidate organization-wide configuration settings as well as render cisco-style configurations based on these settings and device specific values.

Forecast

While this project is focused on Cisco devices, the implemented SZTP standard is supported by a variety of vendors. With the increasing demand for more security in every aspect of networking, the organizations' growing awareness of the problematic vendor lock-in and the expected growth in the vendor support of standardized management solutions, the use of SZTP will likely grow in the future.

2. Assignment

Introduction

Network infrastructure consists of routers and switches. The larger the network the more of these devices are present within the network. The number of such devices can thus become quickly so large, that managing them manually is not only tedious but also poses significant risks of inconsistent configuration and extended downtimes in case of hardware failure. Automation mechanisms for device and configuration management are therefore a necessity as soon as the networks surpass a certain size. Especially the initial device onboarding can be automated efficiently.

Status quo

Many large network device vendors have realized the demand for automation and are offering their own automation solutions, like Cisco's Catalyst Center [2]. Most of these solutions are however not only restricted to the vendor's own devices but are also closed source and often too expensive for small to medium network operators. Some of them have come together with their idea of a standardized way to securely provision new devices without manual intervention on the device itself (zero-touch) [3].

Goals of the Project

The objective of our term project is to lay the groundwork for a network controller that handles zero-touch provisioning of newly installed devices as well as transferring configuration in case of 1-to-1 device replacement. The scope of the controller is limited to Cisco devices for this project, but it should be extendable to support various devices that implement SZTP according to RFC 8572. Furthermore, the controller should also be open for future extension providing more functionality like ongoing configuration after the provisioning process.

3. Requirements

3.1 Use Cases

Once the team had decided what the topic of the semester term should be about, they thought about what use cases should be covered. This step is dedicated to defining the scenarios, in which a system receives an external request and how it should respond to it.

The fundamentals of this project are the Use Cases and the NFR (Non-Functional Requirements).

The Use Cases were assigned to the following three categories: required, optional and out-of-scope.

3.1.1 Overview

The following Use Case diagram shows an overview of the defined use cases. More details can be found in the later sections.

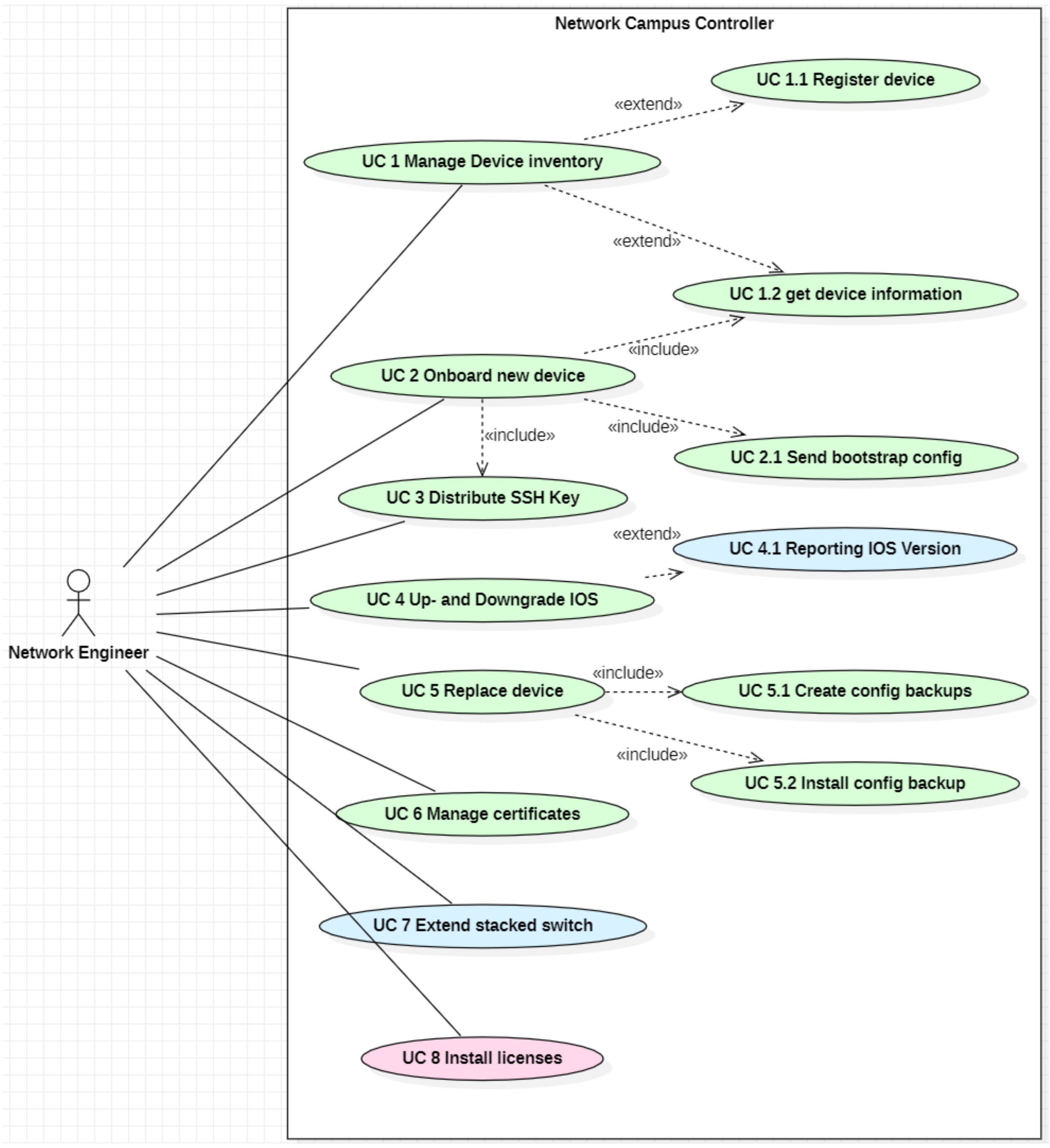

Network Campus Controller
UC 1.1 Register device
«extend»
UC 1 Manage Device inventory
«extend»
UC 1.2 get device information
«include»
UC 2 Onboard new device
«include»
«include»
UC 2.1 Send bootstrap config
UC 3 Distribute SSH Key
«extend»
UC 4.1 Reporting IOS Version
UC 4 Up- and Downgrade IOS
Network Engineer
«include»
UC 5 Replace device
UC 5.1 Create config backups
«include»
UC 5.2 Install config backup
UC 6 Manage certificates
UC 7 Extend stacked switch
UC 8 Install licenses

Figure 1 Use Cases

Legend:

- Green: Are required. They must be implemented.
- Blue: Optional goal. If times allows, the team will start with these.
- Rose: Is out-of-scope but would be a nice feature to have.

3.1.2 Actors

Our system has two actors: the Network Engineer and the Licensing Server. The Network Engineers goal is to manage the devices, set those up and replace it. The Network Engineer will be the only Actor that will interact with the devices.

Optionally, there is also the Licensing Server. Its function is to provide the correct licenses and evaluating.

3.1.3 Use Case description

3.1.3.1 UC 1 Manage device inventory

- As a Network Engineer, I want to have an inventory of all devices, which should be managed by the controller.

The Controllers inventory should be managed via a SSOT.

3.1.3.2 UC 1.1 Register device

- As a Network Engineer, I want to add a new device to the device inventory.

As a precondition, Use Case 1 must be fulfilled. To add a device to the inventory, some basic information (e.g. MAC-Address, Serial number or Certificates) are needed.

3.1.3.3 UC 1.2 Get device information

- As a Network Engineer, I want to gather all my managed devices with their additional information from the database.

As a precondition, Use Case 1 must be fulfilled. When a new device is connected for the first time, the controller has to know which configuration is needed for this specific device.

3.1.3.4 UC 2 Onboard new device

- As a Network Engineer, I want to onboard the new device with the controller.

As a precondition, Use Case 1.2 must be fulfilled. It is required to define the device in the inventory before it boots. When a new device boots up and has configured the DHCP with the correct information for the controller, it sends a request message to the controller. The controller matches the device with its specific information from the inventory. The state of the device should be updated in the inventory.

UC 2.1 Send bootstrap configuration

- As a Network Engineer, I want to send a bootstrap configuration to a cisco device.

As a precondition, Use Case 2 must be fulfilled. The main part of zero touch is to not connect to the console of the device, it should be integrated automatically in the existing network.

3.1.3.5 UC 3 Distribute SSH Key

- As a Network Engineer, I want to distribute SSH Keys to the device.

As a precondition, Use Case 1.2 must be fulfilled. This is used for faster, simpler and safer login to the devices.

3.1.3.6 UC 4 Up- and Downgrade IOS

- As a Network Engineer, I want to upgrade and downgrade the IOS Firmware within the controller,

The image should be saved, and accessible from the network. The Up- and Downgrade should be done at the onboarding process. The device receives a link to the desired image version.

UC 4.1 Reporting IOS Version

- As a Network Engineer, I want to get a report, if a firmware of a cisco device isn't up to date with the configuration on the controller.

Today a common scenario is that a crucial security issue was identified. A patch from the vendor is already available for installation. We want to handle the scenario fast without much effort. In this case we want to get a report, which devices aren't up to date.

3.1.3.7 UC 5 Replace device

- As Network Engineer, I want to replace a failed device with a new one.

As a condition, it only should be replaced with the same model. To onboard the new device, only a backup from the old one is needed.

3.1.3.8 UC 5.1 Create config backup

- As Network Engineer, I want to create a backup from a device within the controller.

The device has to be in the inventory to create backups. The backup has to be stored in a location, which is managed by the controller.

3.1.3.9 UC 5.2 Install config backup

- As Network Engineer, I want to install a backup from a previous device.

As a precondition, Use Case 5.1 must be fulfilled. To replace a device, a backup is first needed. The new device has to be configured with the backup.

3.1.3.10 UC 6 Manage Certificates

- As Network Engineer, I want to import new certificates to the device.

To meet the proper standard today, the provisioning controller has to be able to install the company CA certificate on the device at the time of bootstrapping.

3.1.3.11 UC 7 Extend stacked switch

- As Network Engineer, I want to manage stacked switches.

As stacks are used to increase the capacity, this feature is often used in practice. This feature is not vital however: Even if not used, the switches do still work, but more configuration is needed. We can implement this feature if time allows.

3.1.3.12 UC 8 Install licenses

- As a Network Engineer, I want to install licenses on the devices.

This Use Cases is out-of-scope because the expenditure of time exceeds our time budget.

The licenses, which come from cisco have to be implemented on the correct cisco device. With those licenses, different functionalities can be unlocked.

3.2 Non-Functional Requirements

3.2.1 NFR1 Reliability

Our term project aims to lay the groundwork for an application that automates staging and configuration of network devices. The main goals are to reduce the risk of errors by manual configuration and importantly to save time. The same steps should end in the same configuration.

Acceptance criteria: The configuration sent to the device is the same as if the router were configured manually. The device can process the pre-defined configuration without a problem.

Verification process: The generated configuration from Netbox is the same as, when configured on the device itself.

3.2.2 NFR2 Maintainability / Reusability

During this term project an application with basic use cases will be developed. It is expected that in the future i.e., during a bachelor thesis or other projects, the application will be extended with further functionality. In order to make the reuse of our code possible and easy, we aim at writing modular code.

Acceptance criteria: Each task can be executed separate from the others. The corresponding code is also divided in a modular fashion.

Verification process: Every change to the code will be checked by the other developer before merging takes place.

3.2.3 NFR3 Maintainability / Changeability

Our application should be extendable and changeable in the future. To enable this, we need to avoid redundancy in the code, which complicates changes and reuse of code.

Acceptance criteria: The code contains only well justified code duplicates.

Verification process: The code is checked by the other developer. This happens before the branch will be merged.

3.2.4 NFR4 Maintainability / Analyzability

Since our application will work – in large parts – automatic, we need to make sure that we can track, what actions were taken by the application. This helps not only to control the work progress of the zero-touch configuration process but also to enable easy debugging in case of failures.

Acceptance criteria: Each communication between the controller and the network device to be configured is logged. The log allows to determine which device was involved and which action was taken.

Verification process: The SZTP Controller will be started and a test request for get-bootstrapping-data is sent to the controller. After the request, the log contains the specific device and what action has been done.

3.2.5 NFR5 Transferability / Installability

We want to make sure, that the installation and provisioning of the service in the future should be easy. It is common that hardware fails, or human errors happens therefore it should be easy to recover from these mistakes.

Acceptance criteria: The installation can be done in an acceptable time frame. A cloud native architecture isn't a prerequisite.

Verification process: The configuration on the provided lab setup with the user guide on the mkdocs site can be done in an hour.

3.3 Verification of NFRs

3.3.1 NFR 1 Reliability

Detailed verification process: The configuration from the test with the mocked device is taken as reference. Based on the generated configuration each step was done manually "by hand" directly on the device. There were no issues regarding the generated configuration and the manually typed in commands worked successfully.

Protocol: No issues found.

3.3.2 NFR2 Maintainability / Reusability

Detailed: verification process: Every change to the application branch will be reviewed by the other developer. Only after this, the branch can be merged to the main branch.

Protocol: Was done regularly.

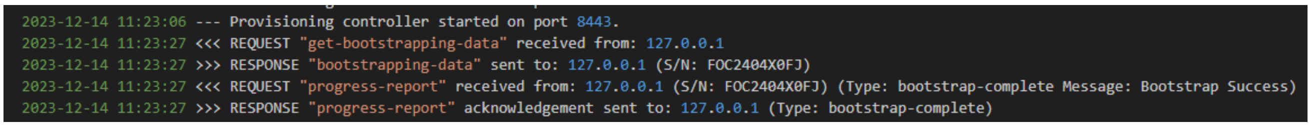
3.3.3 NFR3 Maintainability / Changeability

Detailed verification process: As the code will be checked by the opposite before merging – the code was analyzed and verified that there are no duplicates in the code. But there is always room for improvement and it's a consistent learning curve how the code can be made better readable and faster.

Protocol: Was done regularly.

3.3.4 NFR4 Maintainability / Analyzability

Detailed: verification process: The SZTP Controller is running and can accept requests. The mock "test_controller_with_mock_device" was started and starts the get-bootstrapping-data request and the "report-progress" request. The request was successfully logged. Below is an excerpt from the log.



```
2023-12-14 11:23:06 --- Provisioning controller started on port 8443.
2023-12-14 11:23:27 <<< REQUEST "get-bootstrapping-data" received from: 127.0.0.1
2023-12-14 11:23:27 >>> RESPONSE "bootstrapping-data" sent to: 127.0.0.1 (S/N: FOC2404X0FJ)
2023-12-14 11:23:27 <<< REQUEST "progress-report" received from: 127.0.0.1 (S/N: FOC2404X0FJ) (Type: bootstrap-complete Message: Bootstrap Success)
2023-12-14 11:23:27 >>> RESPONSE "progress-report" acknowledgement sent to: 127.0.0.1 (Type: bootstrap-complete)
```

Figure 2 Log SZTP Controller

Protocol: The log contains the information about the requests and responses.

3.3.5 NFR5 Transferability / Installability

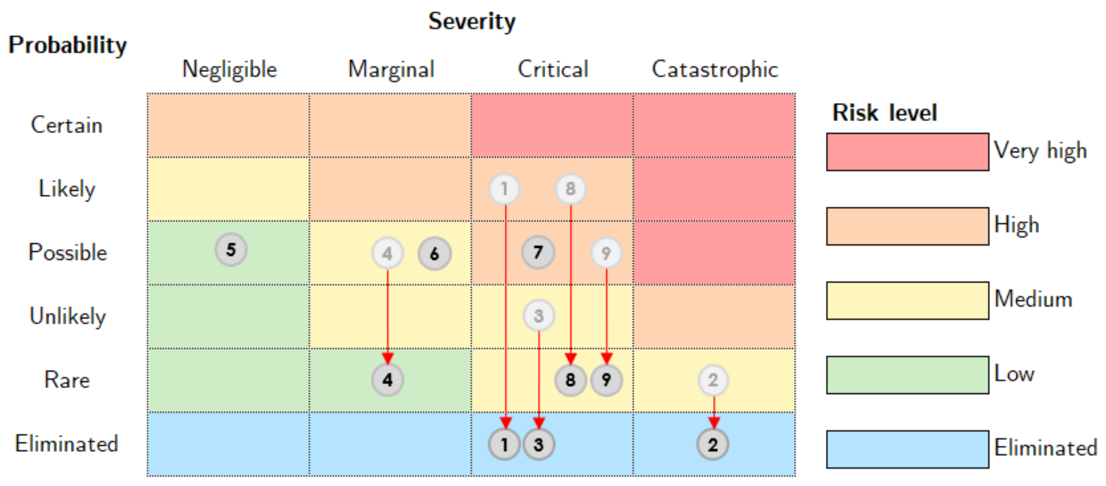
Detailed verification process: The setup can be accomplished within an hour. First the git repository was locally cloned to the server. After this, a new CA certificate was created or if this the network campus controller will be integrated to an existing working environment, there should be already the CA certificate available. The Docker Netbox image was pulled, and the override file was configured according to the environment. In Netbox the regions, the API token and other required information must be defined. Next, at the existing DHCP Server, the Option 143 SZTP has to be defined. Finally, the provisioning server can be started successfully.

Protocol: The installation was completed successfully in about 40 minutes.

4. Risk analysis

Issues can always arise, when working on a project. In preparation we have identified potential risks we could face along the way. The risks have been assessed according to their possibility and impact. Mitigation strategies have been developed in order to reduce or eliminate them.

4.1 Overview



Risks

1 Not enough information on Ciscos PnP protocol
2 Used hardware has no support for PnP
3 Implementing PnP requires too much time
4 Not enough expertise in programming language
5 Illness of a team member
6 Not enough information on SZTP available
7 Problems in implementing SZTP
8 SZTP in virtual devices
9 Chain of trust for SZTP controller

Figure 3 Risk overview

4.2 Details

1. Not enough information on Cisco's PnP Protocol

In this project we will use Cisco's proprietary protocol PnP (Plug-and-Play) to setup and configure the network devices. Since Cisco uses this protocol in their paid solution DNA Center, official resources about the workings of PnP are scarce to non-existent. This means we must rely on unofficial sources and former work of the INS. Additional features/functions will have to be reverse engineered by the authors. Depending on the number and complexity of functions, this will need a considerable amount of time and effort.

Mitigation: To reduce this risk, we planned for an extended elaboration phase to make time for our research.

2. Used hardware has no support for PnP

   There exist many Cisco devices. Since the PnP protocol is proprietary there's no simple way of knowing if all the devices, we want to support with our controller are compatible with all necessary PnP functions. Missing functions would have to be implemented manually, therefore consuming more time, and adding complexity.

   Mitigation: To avoid this risk, we plan to start out with a single network switch model. We will add support for more models if time permits.

3. Implementing PnP requires too much time

   Neither of the authors had any experience with PnP protocol prior to this project. This circumstance along with the fact, that information on the PnP protocol is scarce, makes estimating the complexity and time required to implement the protocol rather difficult. Therefore, implementing PnP could turn out to be more difficult and time-consuming than expected.

   Mitigation: To avoid ending up with a not finished product, we start with a limited amount of mandatory use cases and consider additional use cases optional.

4. Not enough expertise in programming language

   There are programming languages that are more or less suited in the area of network automation. The most represented language in our study is Java which unfortunately is not necessarily well suited for network automation. Python on the other hand is often used for such tasks. Since the authors do not have very much experience in Python, there will be additional time needed to fill these knowledge gaps.

   Mitigation: Since we will already need programming for the prototype, we expect to need time in the elaboration to brush up on our Python skills. This phase is therefore planned with extra time.

5. Illness of a team member

   Considering that only two students work in this project, the prolonged absence of one of them poses a risk to the project's success i.e., delivering a finished product on time.

   Mitigation: We will make use of available collaboration tools to ensure that both team members always have the latest version of the documentation and code base. We furthermore keep track of all short-term scheduled an in-progress tasks as well as the assigned member. This will help the continuation of work in case a team member gets ill.

6. Availability of SZTP implementation in virtual devices

   Since SZTP is a relatively new RFC, chances are that not all devices implement it. High risk is especially present for virtual devices, since a key requirement for the security lies in a hardware based tamper-proof authentication module (at least for Cisco).

   Mitigation: We consider using physical devices which feature the hardware TAM.

7. Chain of trust for SZTP controller

   An obstacle could be acquiring the ownership vouchers for the devices (in the scope of Cisco devices).

   Mitigation: We consider using the traditional ZTP approach to avoid the certificate issues.

4.3 Risk development

4.3.1 12.10.2023

Risk 1, 2, 3

To circumvent the risks concerning PnP and to lower dependency on a single vendor, we decided to switch from the proprietary PnP and instead plan to base our controller on the SZTP approach described in RFC8572. All risks concerning PnP are therefore eliminated.

Risk 6

Since SZTP is a rather new standard, and neither of the team members has any prior experience with it, a lack of information in form of documentation and examples could hinder the development of the application.

Risk 7

Especially since SZTP is a new approach which seems to no be widely used at all, the degree of which SZTP is implemented by vendors, in this case Cisco, as well as their adherence to the standard is unknown. So any problems encountered during the development could impact the success of this project.

Risk 8

After elaborating the necessity of a physical device to be able to use SZTP with our advisors, they committed to provide us with a suitable switch or router. The risk probability is therefore lowered to rare for the moment.

Risk 9

After elaborating the necessity of the ownership voucher for using SZTP with our advisors, they committed to make the necessary inquiries to their supplier for us. The risk is therefore lowered to rare for the moment.

4.3.2 02.11.2023

Risk 4

After taking some time to read into a Python tutorial, Patrick feels more comfortable with the Language. The Flask Framework is still new to both team members, but they are confident, that the programming language won't pose unmanageable hurdles. This risk is therefore lowered to rare.

5. Evaluation

5.1 Provisioning Protocol

The first decision is the fundament of the project. The goal is to create a Campus Network Controller, there are a few different protocols which can be used. The question is which protocol best suited to use to realize the Campus Network Controller. Every protocol has their own up- and downsides. The evaluation is focused on Cisco PnP and the RFC 8572 SZTP.

5.1.1 Cisco Plug and Play PnP

First released was Cisco PnP in 2016.[4] This protocol has already a few years on it, nevertheless this means that the protocol is mainly bug-free and was well-updated over the years. Nowadays Cisco PnP isn't open source anymore, it is integrated in the Cisco controller Cisco Catalyst and DNA Essentials[1] or Cisco Catalyst and DNA Advantage.

5.1.1.1 How Cisco PnP works

PnP is the zero-touch solution by Cisco for its devices. First there are the Cisco devices, which embed the PnP agent. The PnP Server also known as DNA Center, is the core of Cisco's PnP solution. There happens all the magic for the configuration of the devices. The DNA Center works with PnP Protocol, which is no longer open source. It works with HTTPs and XML protocols. The problem here is that it is unknown how the product will develop in the future. Cisco might change the available features or protocol syntax without public notice. The same is true for successor devices to those that reach their end of life. That means, continuous checks and adaption to the new syntax are needed. If this project should be maintained after the initial term, it needs more care afterwards.

The last part is PnP Connect, it's a cloud link-up, the on-prem server to enable cloud access.

This won't be a part of the Network Campus Controller, the team would focus on the local server for now.

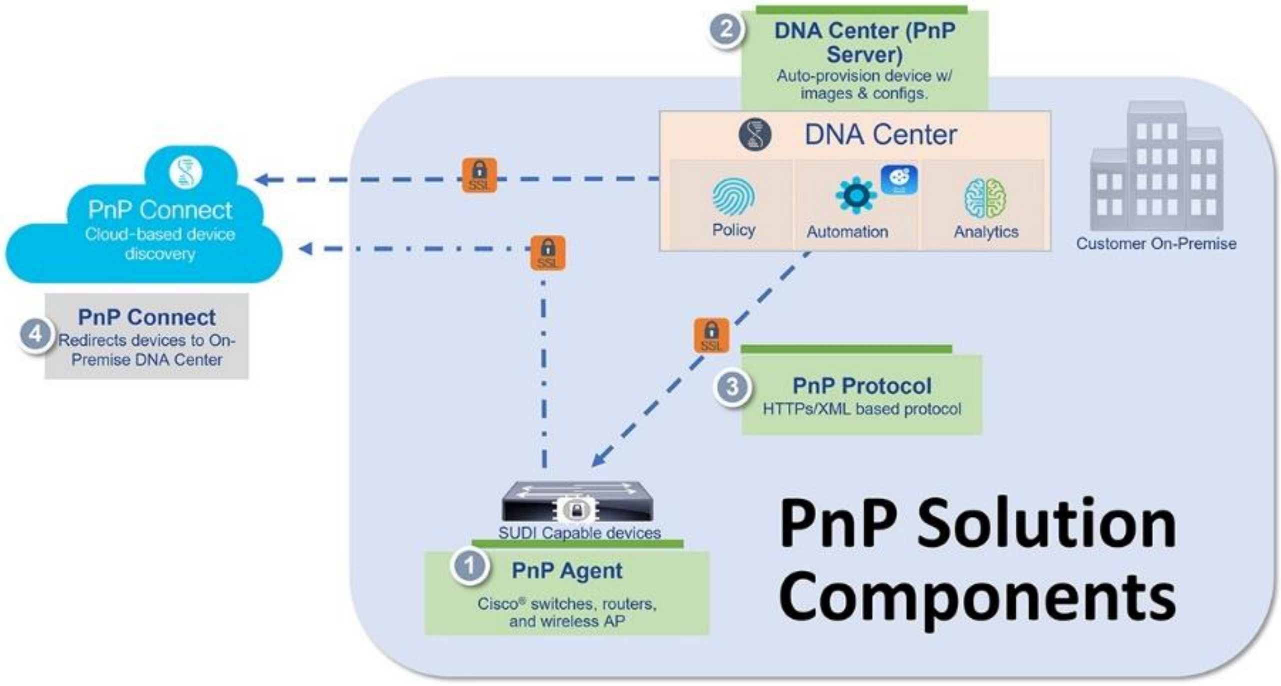


Figure 4 Cisco PnP [1]

Advantages

- It is widely used.
- Much information was available online when it was free. Therefore, this information is not quite up to date.
- Protocols which are used are a good start.
- When you have a problem and use the licensed version you receive support.

Disadvantages

- The Cisco PnP protocol isn't open source anymore.
- For PnP you have to pay now (and not a little).
- More time is used to find out, how the protocol works. And even at with time it can happen that you can't reverse engineer everything.

5.1.2 Secure Zero Touch Provisioning SZTP

Secure Zero Touch Provisioning is an RFC standard since 2019, the relatively new protocol has adopted the secure component to ZTP. ZTP helps to quickly deploy new devices with pre-configured scripts or commands. The protocol has really similar functions and methods, the devices also work the same so there also has to be same configuration at the end. SZTP is supported by the vendors Cisco, Juniper, Huawei and Nokia, among others.

In addition to ZTP, the new standard adds a chain of trust to the vendor. The issuing of certificates is controlled via the vendors servers, the API is called MASA.

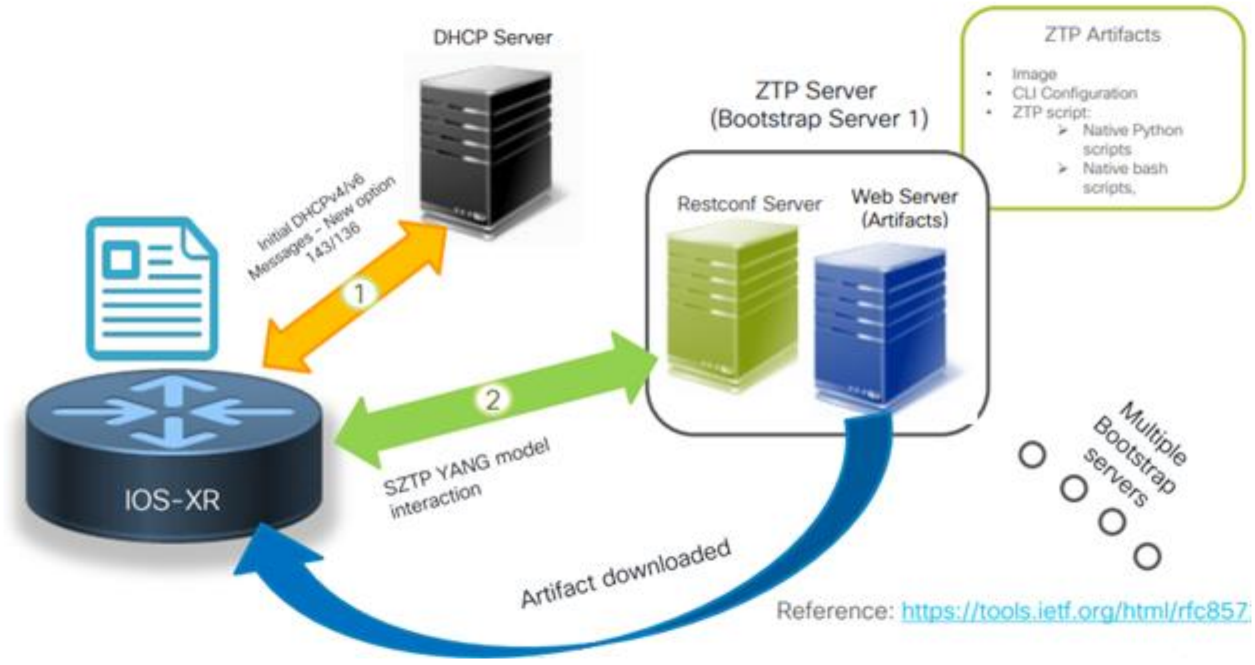

DHCP Server
ZTP Artifacts
Image
CLI Configuration
ZTP script:
Native Python scripts
Native bash scripts,
ZTP Server (Bootstrap Server 1)
Restconf Server
Web Server (Artifacts)
Initial DHCPv4/v6 Messages – New option 143/136
1
2
IOS-XR
SZTP YANG model interaction
Multiple Bootstrap servers
Artifact downloaded
Reference: https://tools.ietf.org/html/rfc857

Figure 5 SZTP on Cisco devices [5]

In the picture above the workflow of SZTP is displayed. The protocols used within SZTP are YANG, RESTCONF or HTTP(-S).

Advantages:

- Standard by IETF, this means there are no possibilities that this feature would become closed source at some time.
- Better, well formatted protocols are supported (e.g No XML)
- It's free to use, only the device costs.

Disadvantages:

- There is no support, if something doesn't work, you have to do the troubleshooting by yourself.
- The standard is relatively new, so on the internet there is not so much information to find about it.

5.1.3 Suitability of SZTP as a solution for the assignment

Since the scope of this project has been limited to Cisco devices, this chapter focuses on a current version of Cisco IOS XE.

Prerequisites for using SZTP on Cisco IOS XE Dublin 17.11.x[6]

- The management system must validate that it is provisioning a valid device.
- The device must validate that it is being deployed in the correct network.
- The device must validate that the provisioning data has not been tampered with.
- Provisioning must use a secure transport protocol for data communication.

The mutual validation of device and server is accomplished by certificates secured with a trust anchor. This means, the device provides a signed version of its identifier, the Secure Unique Device Identifier (SUDI) which signature the server can check against a Cisco root certificate. The secret keys used for signing the SUDI reside in a proprietary tamper-resistant Trust Anchor module (TAM) on the device.

The device on the other hand, validates the server by an ownership voucher - containing its serial number – signed by Cisco. This ownership voucher is normally generated by an API service by supplying proof of purchase.

This validation method leads to the following questions:

1. Are there a SUDI and ownership voucher for virtual devices (no hardware TAM)?
2. Is there a way to use SZTP without ownership voucher (in case there will be none available for our project)?

After bringing up this issue at the weekly advisor meeting, the advisors committed to supply us with a physical device along with its corresponding ownership voucher.

Another remaining question is whether SZTP provides any function to act as a client for management of the device after the provisioning process. The team could find no indication, that such a function exists. SZTP includes pre- and post-configuration scripts that run before and after the provisioning process, but once they are finished, no further functionality is provided by SZTP.

This means that we need another mechanism to manage the device after staging.

5.1.4 Decision on Provisioning Protocol

PnP provides a bad starting point for our project, mainly because of two reasons: 1. It's proprietary, meaning there will be little information publicly accessible on how it works. Which is bad because we will rely on detailed information in order to implement our solution. 2. Its compatible only with Cisco devices. Which will limit the scope of the controller's usefulness for future extensions beyond the timeframe of out semester project to a single vendor.

SZTP on the other hand is an IETF standard in the making (proposed standard). It is already supported by different vendors and by being a standard, the protocols working mechanisms are publicly available. The team therefore decided to go with SZTP as provisioning protocol for this project.

5.2 Programming language

The programming language is the base of this project and is an important part of it because different languages have different support for tools.

5.2.1 Golang

Golang is an open-source language, which is supported by Google. The first stable version Golang was released in 2012.[7]

Advantages

- It's a modern programming language.
- It's often used for cloud-services.

Disadvantages

- Patrick and Vanessa do not have any previous knowledge of this language.

5.2.2 Python

Same as Go, Python is also an open-source programming language, but it was first released in 1991.

Advantages

- It's a modern programming language with a good ecosystem.
- It's the most used language.
- Vanessa has attended the module at OST "Automation with Python".
- Patrick would like to learn Python.

Disadvantages

- Patrick does not have any previous knowledge about Python.

5.2.3 Decision

Because of the highest common denominator, the team decided to use python as the programming language in the SA. The team has more interest in learning and deepening their knowledge of Python.

5.3 Device Inventory

The application needs an inventory to record the devices which are present. Additionally, their base configuration values, ownership vouchers and so on are needed. Furthermore, the application needs to store some operational data like target firmware version.

The team evaluated three possible solutions to address the needs concerning device inventory: Ansible, Netbox and a custom implementation.

5.3.1 Ansible

Advantages:

- The tool is already known by the team members from earlier semesters (Network Automation).
- Also provides easy interaction with devices in the form of tasks.

Disadvantages:

- The inventory is stored in flat files: no GUI, complicated process to write variables like device status.
- Ansible focuses mainly on automation action rather than on device inventory.

5.3.2 Netbox

Advantages:

- Netbox includes a WebGUI.
- Netbox provides database storage.
- Netbox provides read and write API access.
- Netbox includes configuration template rendering, which might be usable for the needs of this project.
- Netbox can be used as SSoT to provide all configuration information in future extensions of the controller.

Disadvantages:

- Netbox is new to the team members.
- Netbox provides much more functionality than needed for this project.

5.3.3 Custom implementation

Advantages:

- A custom implementation provides maximum flexibility in regards of data storage structure and adaptability in interfacing with the provisioning and operation controller as well as possible other systems.

Disadvantages:

- A custom implementation needs much more work to develop, thus reducing time from our time budget.
- A custom implementation also needs development of additional elements like GUI.
- A custom implementation poses additional work for maintenance.

5.3.4 Decision on device inventory

The team decided to use Netbox for this project, as it fulfils all the functional needs, while not requiring much additional development effort. Additionally, it is future-proof, meaning it can be used as SSoT for all configuration when the provisioning and operation controller will be extended. Furthermore, it is open-source and allows to be extended with (custom) plugins, should the need ever arise.

5.4 SZTP Provisioning Controller

The SZTP Server is the central piece of this project, and the operation controller consists of different parts. The Controller is responsible for the different SZTP tasks, such as managing the secure part: Certificates, vouchers, onboard devices or upgrade the firmware. There are more segments, which are going to be implemented and need different interfaces to the outside. Here only cover the most important sections of the SZTP controller are covered and their advantages and disadvantages described.

5.4.1 Python Framework

Python Frameworks are here for the communication to devices, the inventory and to the MASA Server for obtaining ownership vouchers and licenses.

5.4.1.1 SZTPD Python Framework

This Python Framework was developed by Watson, a member, who also wrote the RFC standard. The SZTPD library is now in the Alpha release and as of today (26.10.2023) the current version is 0.0.11.[8].

Advantages:

- Already a usable Framework for SZTP.
- With this Framework a lot of time could be saved – don't reinvent the wheel.
- Is written in Python, our preferred language.

Disadvantages:

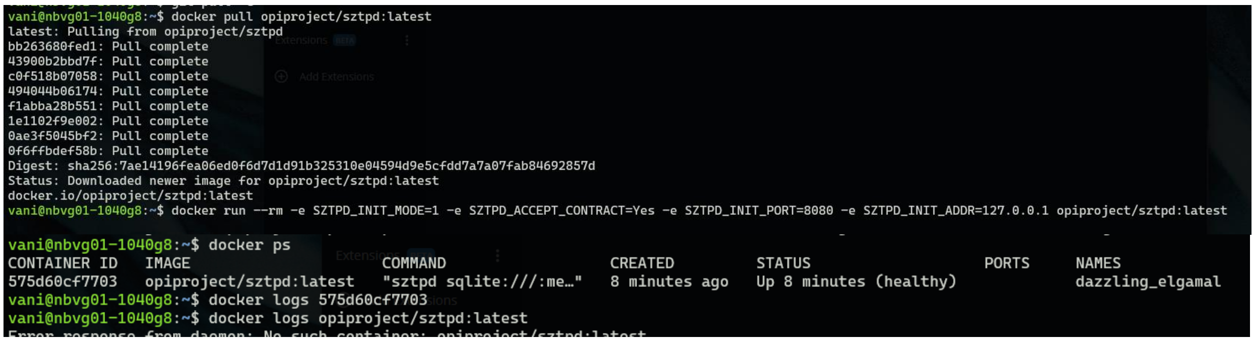
- This tool is in Alpha release and could have a lot of bugs.
- When something does not work, figuring out the problem could be time consuming.

5.4.1.2 Research on SZTPD Framework

During the research, we tried to install and configure the SZTPD Bootstrapping Server[9]. The Repo was last updated early February with the second release of this repo (Checked on November).This is repo has a Dockerfile included, which uses the Python library[8] from Watson, who is the main-contributor of the SZTP Standard.

The team tried to install SZTPD – but unfortunately without success.

The run command was successful, but the container didn't have any logs and the SZTPD_INIT_PORT=8080 wasn't even assigned.



```
vani@nbvg01-1040g8:~$ docker pull opiproject/sztpd:latest
latest: Pulling from opiproject/sztpd
bb263680fed1: Pull complete
43900b2bbd7f: Pull complete
c0f518b07058: Pull complete
494044b06174: Pull complete
f1abba28b551: Pull complete
1e1102f9e002: Pull complete
0ae3f5045bf2: Pull complete
0f6ffbdef58b: Pull complete
Digest: sha256:7ae14196fea06ed0f6d7d1d91b325310e04594d9e5cfdd7a7a07fab84692857d
Status: Downloaded newer image for opiproject/sztpd:latest
docker.io/opiproject/sztpd:latest
vani@nbvg01-1040g8:~$ docker run --rm -e SZTPD_INIT_MODE=1 -e SZTPD_ACCEPT_CONTRACT=Yes -e SZTPD_INIT_PORT=8080 -e SZTPD_INIT_ADDR=127.0.0.1 opiproject/sztpd:latest

vani@nbvg01-1040g8:~$ docker ps
CONTAINER ID   IMAGE                     COMMAND                  CREATED         STATUS                    PORTS     NAMES
575d60cf7703   opiproject/sztpd:latest   "sztpd sqlite:///:me…"   8 minutes ago   Up 8 minutes (healthy)              dazzling_elgamal
vani@nbvg01-1040g8:~$ docker logs 575d60cf7703
vani@nbvg01-1040g8:~$ docker logs opiproject/sztpd:latest
Error response from daemon: No such container: opiproject/sztpd:latest
```

Figure 6 No logs from SZTPD

In this case, SZTPD Bootstrapping Server can't be used, since it's difficult to debug something without any logs.

5.4.2 Web-API

For the communication with the devices and other services a web-server-framework is needed. Here are two frameworks described, which can both be used with Python.

5.4.2.1 Flask

Advantages

- It's simple to learn and build up.
- It has a large community for help.

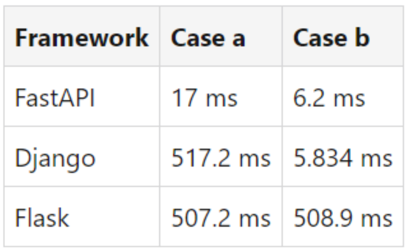
Disadvantages

- It doesn't scale/needs more time for requests, because flask is a single source[10].
- Flask is new to the team members.

5.4.2.2 FastAPI

Advantages

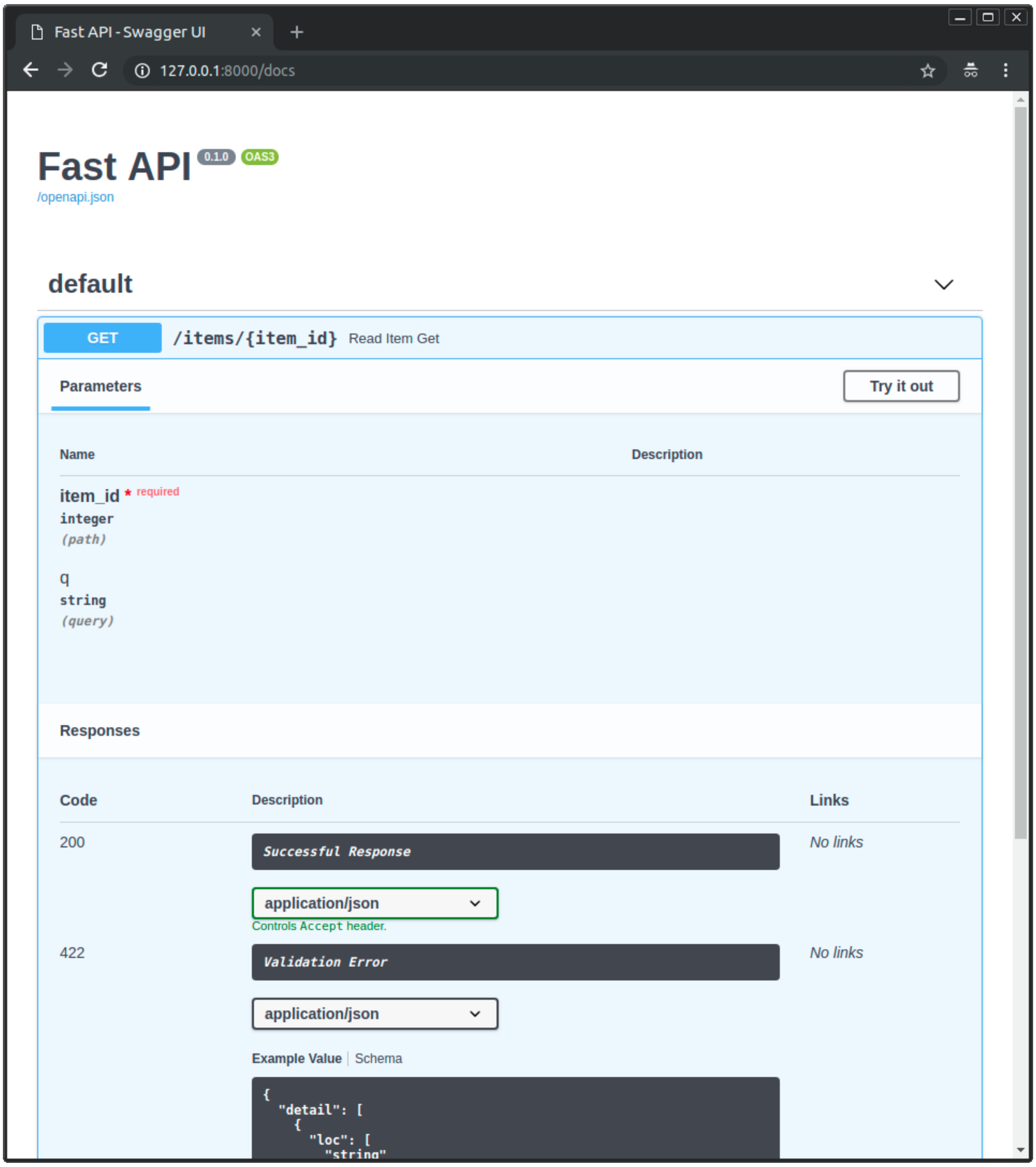
- It's faster than Django and Flask to load a page. Below is a comparison[11]:



Framework	Case a	Case b
FastAPI	17 ms	6.2 ms
Django	517.2 ms	5.834 ms
Flask	507.2 ms	508.9 ms

Figure 7 Comparison web frameworks

- It has a built-in inspector for queries, with "/docs" added at the end of the URI. Here is an example[12]:



Name	Description
item_id * required integer (path)	
q string (query)	

Responses

Code	Description	Links
200	Successful Response; application/json; Controls Accept header.	No links
422	Validation Error; application/json	No links

Example Value | Schema

```
{
  "detail": [
    {
      "loc": [
        "string"
```

Figure 8 Screenshot Fast API

Disadvantages

- It's more complicated than flask.
- It has a smaller community for help.
- It's also new to the team members.

5.4.3 Evaluation on SZTP Provisioning Controller

As shown in the research it's a challenge to implement an already available SZTP Controller. The documentation is often lacking information, or it isn't up to date anymore. If the team were to work with an existing SZTPD controller, it could quickly become confusing, and the troubleshooting would take more time than if everything was set up by the team itself. From the teams experience it's advisable to build an own controller.

Because both members are beginners in Pythons Web APIs, they prefer a Framework which can offer much community-help. FastAPi has great features, but a smaller community as Flask. However, the most impact from the advantages is the huge community. For these reasons the Flask framework was chosen.

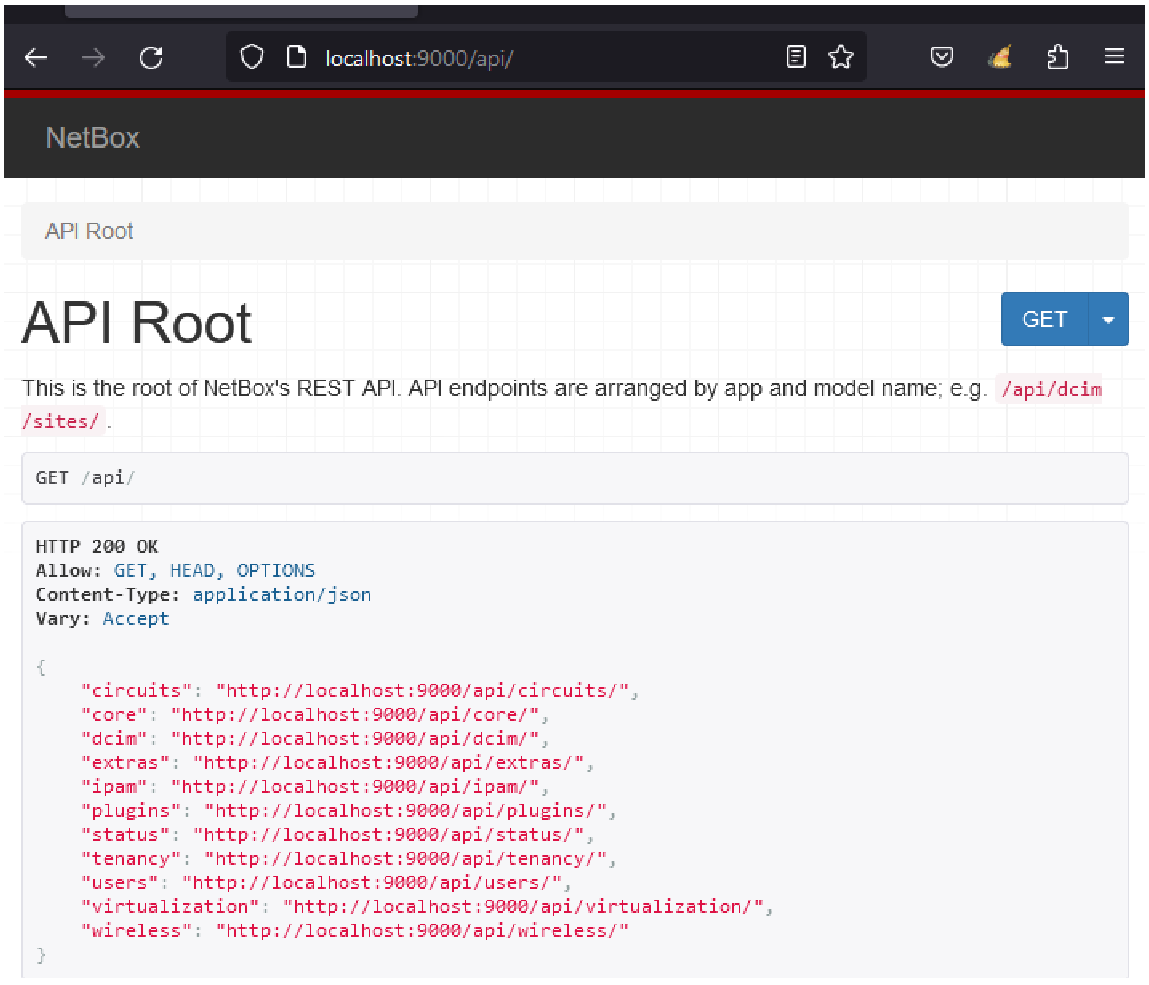
5.5 Netbox Interface to SZTP Controller

As Netbox is used for the inventory, a programable interface for the SZTP provisioning controller and the operation controller is needed. Available for selection is the pure API access or the pynetbox library. Both use a token created from within Netbox to authenticate the requests.

5.5.1 Netbox API

The Netbox API is at first sight well structured. It looks just like any other applications API.

Below is an excerpt of the API.



API Root

This is the root of NetBox's REST API. API endpoints are arranged by app and model name; e.g. `/api/dcim/sites/`.

```
GET /api/
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "circuits": "http://localhost:9000/api/circuits/",
    "core": "http://localhost:9000/api/core/",
    "dcim": "http://localhost:9000/api/dcim/",
    "extras": "http://localhost:9000/api/extras/",
    "ipam": "http://localhost:9000/api/ipam/",
    "plugins": "http://localhost:9000/api/plugins/",
    "status": "http://localhost:9000/api/status/",
    "tenancy": "http://localhost:9000/api/tenancy/",
    "users": "http://localhost:9000/api/users/",
    "virtualization": "http://localhost:9000/api/virtualization/",
    "wireless": "http://localhost:9000/api/wireless/"
}
```

Figure 9 Netbox API

Advantages

- It's structure is the same as any other API interface, therefore only little knowledge is required.
- It has no dependencies.

Disadvantages

- More parsing of the answer is needed.

5.5.1.1 Pynetbox

Pynetbox is a Python library. As mentioned on the Github repo of this library, it provides more features than Netbox-python. Maybe not all features are needed, but there would be the possibility for it. The repo was last updated in September with the Version 7.2.0 (as of late November).

Git-Repo: Releases · netbox-community/pynetbox (github.com)

Documentation: https://pynetbox.readthedocs.io/en/latest/

Advantages

- It regularly receives updates.
- It's well documented.
- Less code is need for the same result as with the pure API.

Disadvantages

- Its usability depends on the further development, whether the library receives updates and is well maintained.

5.6 Evaluation on Netbox Interface to SZTP Controller

Because less and better structured code and the all-in-one documentation is preferred, the team decided to use the pynetbox library.

5.7 User interface

Like every other application, this Network Controller also needs a UI. The team thought about different options, one of which was develop a complete graphical web-based interface. On the other hand, the application could have also been configured and controlled completely via text files and CLI.

For the inventory part Netbox was chosen, which already provides an organized and easy to use web interface. Since most user interaction will be happening with the inventory management, a large part of the UI is already covered.

As for the provisioning and backup controller: They both only need starting via CLI and no other user interaction. The provisioning controller can thus be easily auto started at boot time via cron or could also be enveloped in a service. The backup controller is supposed to be run frequently automated via cron.

Therefore, no additional development time is needed for the user interface.

5.8 Operation Controller

The operation controller is responsible for the devices after the bootstrapping process. The operation controller should send a new configuration to the device and must create backups from the devices, which are in the inventory.

5.8.1 Creating Backups

The process of creating backups of the devices include, that the backups will be saved at a location, which can be also accessed by the provisioning controller. The provisioning controller needs the backup for the device replacement.

5.8.1.1.1 Netbox Backup Plugin

Advantages

- One less additional tool, which has to be taken care of and get to know means less effort.

Disadvantages

- Netbox isn't made for this aspect, so there could be functionality missing or bugs, that have not yet been fixed.
- Not many people use such a plugin – not much additional information is available on the internet.

5.8.1.1.2 Netbox Scripts

Advantages

- One less additional tool, which has to be taken care of and get to know means less effiort.

Disadvantages

- Support for additional features is limited or needs much more effort.
- Netbox isn't made for this aspect, so there could be bugs.

5.8.1.1.3 Nornir mit Napalm

Advantages

- The team is already a little familiar with Nornir from the Network Automation module.
- There are already guides available online for backing up devices.

Disadvantages

- One additional tool, which has to be taken care of.

5.8.1.1.4 Rancid[13]

Advantages

- This tool has been available for more than 10 years. It is not new on the market and is certainly stable.
- The team knows a user who has successfully deployed it and could provide help.

Disadvantages

- The website looks a little old-fashioned. The copyright stamp was from 1996-2016. Maybe it is not well maintained anymore.
- One additional tool, which has to be taken care of.

5.8.1.1.5 Ansible

Advantages

- The team is already familiar with Ansible from the Network Automation module.
- There is much information available about Ansible. Ansible is known and used by a lot of people.

Disadvantages

- Ansible has many features to offer – maybe already a little too much. Ansible is more heavy weight and sluggish than Nornir.

5.8.2 Decision on creating backups

The Netbox tools and script are behind Ansible, Rancid and Nornir with Napalm. Because the others aren't just little used plugins and have a lot more downloads. Rancid's website does not look trustworthy – it looks like a website which was made ten years ago and didn't receive many updates since its creation. The project however should live on.

Because Nornir and Napalm is more lightweight than Ansible and an interesting guide was already found, the team decided to use Nornir with Napalm for the backup.

5.8.3 Operation Controller: Reporting and configuration

In addition to creating backups the operation controller should also be able to report the device IOS version and configure the device. The device should also be able to be configured as a stack with another device.

5.8.3.1 Ansible

Advantages

- It uses structured data for configuration: YAML.
- It has many modules: open for further development.

Disadvantages

- Maybe it's too slow: Tasks are only processed in a serial way.
- Ansible has many features to offer – already a little too many?

5.8.3.2 RestConf

Configuring a Cisco device using RESTCONF (RESTful Network Configuration Protocol) involves REST-like methods as GET, POST, PUT, PATCH, and DELETE.

Advantages

- REST-full architecture: Already known processes.
- It's human readable, what will be sent: easier for debugging.

Disadvantages

- Older devices maybe do not implement Restconf.
- (Restconf has to be enabled on the device first.)
- Limited operations: the operations are bound to the HTTP methods, which may not cover all the uses cases or further development. A better option would be to use Netconf with more options.

5.8.3.3 gRPC

gRPC(gRPC Remote Procedure Call)[14] was developed by Google, but is now maintained by CNCF. It's implemented with HTTP/2.

Advantages

- It has high efficiency by using HTTP/2.

Disadvantages

- Not all devices may have a good support for gRPC.
- It must be configured on the device.
- It has a steep learning curve.

5.8.3.4 Netmiko

Netmiko is a CLI based automation tool and implements Paramiko. Python is the only language which is supported.[15]

Advantages

- It's easy to use: Could be implemented with Netbox and modified for every different OS.
- It's supported by many vendors.
- It supports Python: The chosen language for this project.

Disadvantages

- It relies on SSH only.
- It's limited to CLI Automation: Might be unstable when more advanced interactions are needed. A better option would be using an API.

5.8.4 Decision on Operation Controller: Reporting and configuration

There are many nice tools to get into und learn how they work. All are a little bit different and have their own advantages and disadvantages.

Unfortunately, there wasn't enough time at the end of the project to implement this part. Therefore, a decision a tool for the reporting and configuration operations was not made.

6. Problems encountered

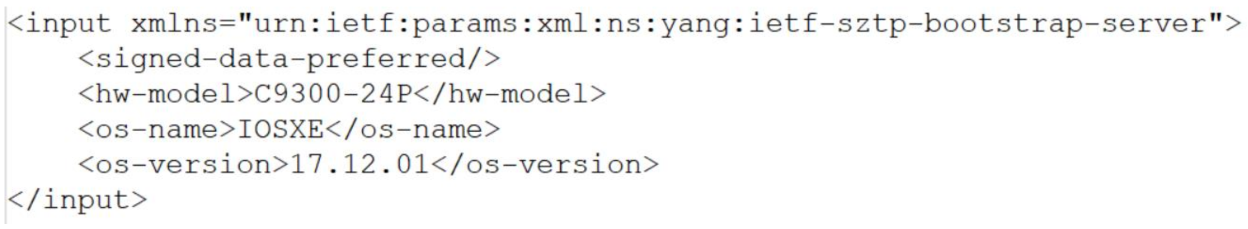
6.1.1 DHCP-Option 143

The RFC 8572 on SZTP[3] references RFC 7227 on DHCP Options Guidelines[16] for information on options with URI information. This RFC presents two possibilities. One, include a single URI of variable length in the options data field. Since there is only one URI, it's length can be inferred from the options length contained in the option-len field which is set automatically by the DHCP server software. Possibility two is including a list of URIs in the option data field. To separate the URIs on the client side, each URI needs to be preposed by a 2-Byte number representing its length.

Since there is only one bootstrap server, the team went with possibility one. After many tries, they found out, that the Cisco device in use, always needs a list (meaning with the preposed 2-Byte length of the URI) even when only one URI is used.

6.1.2 Little information in Request for Bootstrap data

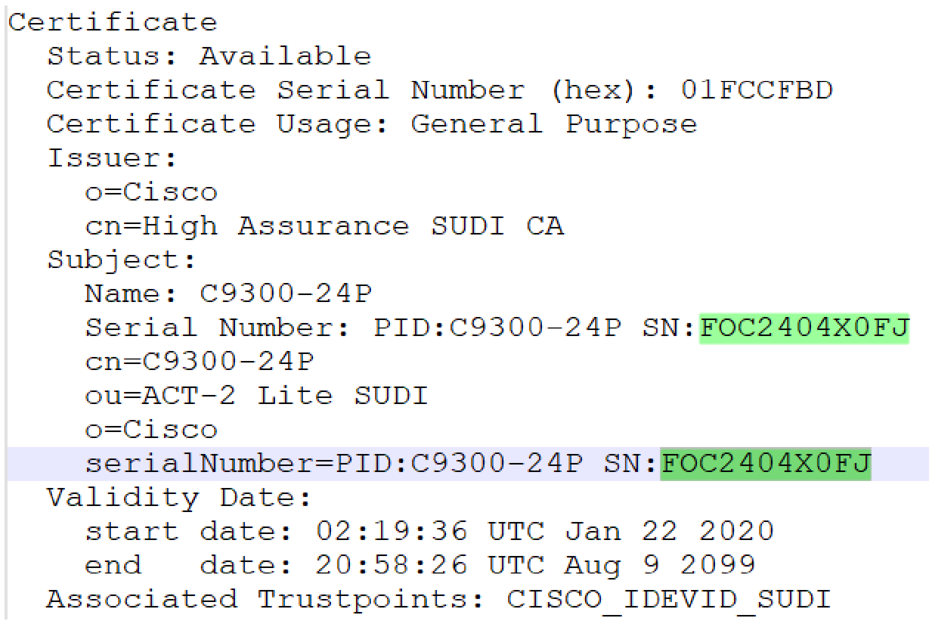
The controller needs to be able to identify the device making the request to query the corresponding values for the configuration from our inventory. Unfortunately, the request only contains information about the device model and its OS.



```
<input xmlns="urn:ietf:params:xml:ns:yang:ietf-sztp-bootstrap-server">
    <signed-data-preferred/>
    <hw-model>C9300-24P</hw-model>
    <os-name>IOSXE</os-name>
    <os-version>17.12.01</os-version>
</input>
```

Figure 10 Data contained in request to /restconf/operations/ietf-sztp-bootstrap-server:get-bootstrapping-data

The team knows that the client certificate used by the device to authenticate itself to the server includes the serial number. If the certificate data can be passed from the web server to the application, the team would be able to extract the device id from there.



```
Certificate
  Status: Available
  Certificate Serial Number (hex): 01FCCFBD
  Certificate Usage: General Purpose
  Issuer:
    o=Cisco
    cn=High Assurance SUDI CA
  Subject:
    Name: C9300-24P
    Serial Number: PID:C9300-24P SN:FOC2404X0FJ
    cn=C9300-24P
    ou=ACT-2 Lite SUDI
    o=Cisco
    serialNumber=PID:C9300-24P SN:FOC2404X0FJ
  Validity Date:
    start date: 02:19:36 UTC Jan 22 2020
    end   date: 20:58:26 UTC Aug 9 2099
  Associated Trustpoints: CISCO_IDEVID_SUDI
```

Figure 11 Command output extract of 'show crypto pki certificate'

Another less elegant option would be to only provide a generic script via SZTP to the device that then could read the serial number from the device locally and request the specific configuration in another request to the controller.

6.1.3 Change in Provisioning technology

In this project zero-touch configuration for network devices should be realized. The base of the zero-touch is the protocol. There are a few available, but the challenge here is to implement it as open source. In this case, the Cisco DNA Center can't be used, but the PnP protocol from the DNA center could be reverse engineered.

First in the term project the team started with Cisco PnP, collected information and tried to understand how the protocol works. Cisco PnP was some time ago documented in their resources. In this time PnP could be implemented with Cisco devices, now PnP was added to the DNA Center and isn't documented in their resources anymore. The team got to know that it could be a huge challenge to implement Cisco PnP because of the lacking information or that there aren't any more updates for the PnP Protocol. More information can be found in the evaluation of Cisco PnP or SZTP.

After the evaluation phase, the team progressed with the RFC standard SZTP. A new research process began with SZTP, again collecting information and try to find some examples of other network engineers.

Now in the eighth week of the term project, the team had made some progress for the prototype, but the most important thing was missing: the voucher for the Cisco devices. The voucher is issued by Cisco to complete the chain of trust. In the meeting of this week, it was decided to proceed with ZTP – without the secure part. This would cost us an additional amount of time – ZTP would use some basics of SZTP. Nevertheless, it again meant repeating the research about another technology.

Fortunately, at the evening of the meeting, Yannick gained access rights with the correct permissions. He handed the vouchers to the team so that the project could now proceed with SZTP.

6.1.4 Obtaining Device Certificate

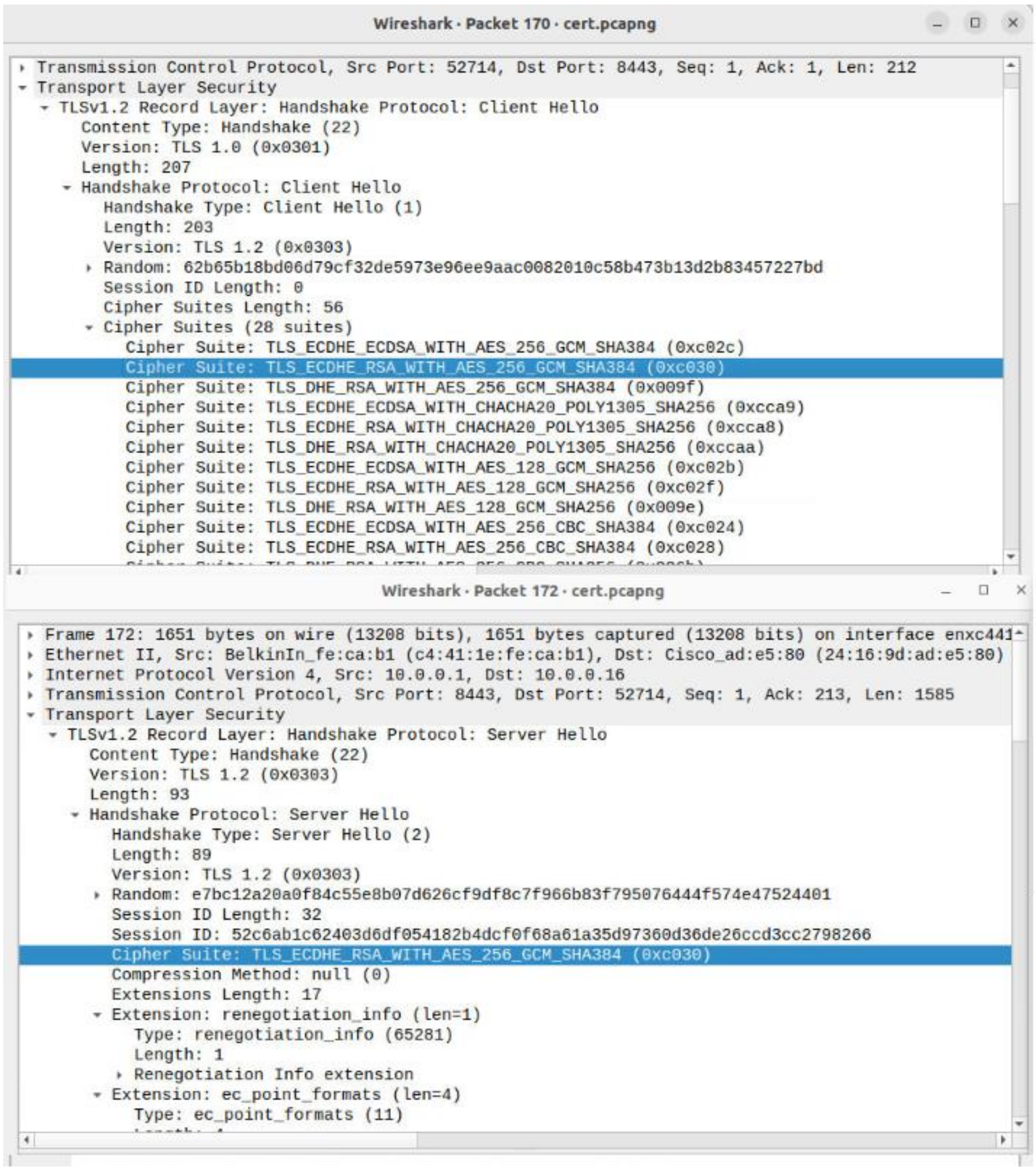
The device uses its private key to initiate the TLS session with the SZTP Server and therefore sends the corresponding certificate in the TLS request. On the server side the team needs to extract this certificate from the TLS session, in particular the devices serial number from the subject field. Unfortunately, this proved to be much harder, than anticipated.

The early tries with just the built-in development webserver of flask couldn't get the team the client certificate content. They then tried to use uWSGI to serve the app on a socket and use Nginx as a proxy to forward the certificate content to the flask application. Unfortunately, this didn't bring any more success.

From our advisors, the team got the idea of using the werkzeug library and a custom request handler to serve the app without Nginx. With this solution they finally could get the client certificate retrieval to work. However, this only applied to their own test client certificates signed by their own test CA.

With client authentication enabled the Cisco device could not establish a TLS session correctly. Since cisco uses a large variety of CAs, some research and a lot of testing were necessary to get the chain of root CA and intermediate CA right. With the devices client certificate extracted from a TCP dump and the correct chain of CA certificates, the team managed to verify the client certificate manually using OpenSSL commands. However, mTLS in their application still wasn't working.

The team tested again some other theories. The first of which was that maybe client and server can't find a common cipher suite. An analysis of the TLS handshake in Wireshark proved this theory wrong. The client advertised a bunch of cipher suites, and the server chose one of them, specifically `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`.


Wireshark · Packet 170 · cert.pcapng

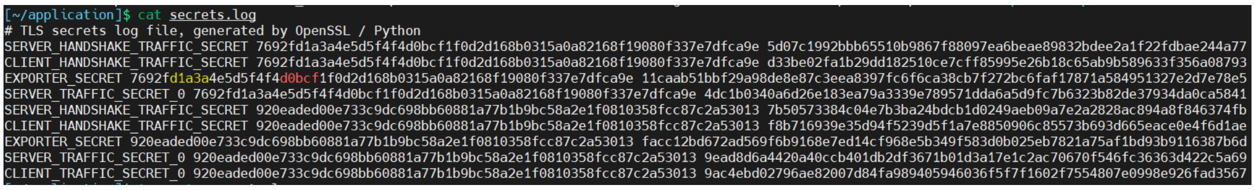
```
▸ Transmission Control Protocol, Src Port: 52714, Dst Port: 8443, Seq: 1, Ack: 1, Len: 212
▾ Transport Layer Security
  ▾ TLSv1.2 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 207
    ▾ Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 203
        Version: TLS 1.2 (0x0303)
      ▸ Random: 62b65b18bd06d79cf32de5973e96ee9aac0082010c58b473b13d2b83457227bd
        Session ID Length: 0
        Cipher Suites Length: 56
      ▾ Cipher Suites (28 suites)
          Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
          Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
          Cipher Suite: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x009f)
          Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
          Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)
          Cipher Suite: TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xccaa)
          Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
          Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
          Cipher Suite: TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x009e)
          Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xc024)
          Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)
```

Wireshark · Packet 172 · cert.pcapng

```
▸ Frame 172: 1651 bytes on wire (13208 bits), 1651 bytes captured (13208 bits) on interface enxc441
▸ Ethernet II, Src: BelkinIn_fe:ca:b1 (c4:41:1e:fe:ca:b1), Dst: Cisco_ad:e5:80 (24:16:9d:ad:e5:80)
▸ Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.16
▸ Transmission Control Protocol, Src Port: 8443, Dst Port: 52714, Seq: 1, Ack: 213, Len: 1585
▾ Transport Layer Security
  ▾ TLSv1.2 Record Layer: Handshake Protocol: Server Hello
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 93
    ▾ Handshake Protocol: Server Hello
        Handshake Type: Server Hello (2)
        Length: 89
        Version: TLS 1.2 (0x0303)
      ▸ Random: e7bc12a20a0f84c55e8b07d626cf9df8c7f966b83f795076444f574e47524401
        Session ID Length: 32
        Session ID: 52c6ab1c62403d6df054182b4dcf0f68a61a35d97360d36de26ccd3cc2798266
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
        Compression Method: null (0)
        Extensions Length: 17
      ▾ Extension: renegotiation_info (len=1)
          Type: renegotiation_info (65281)
          Length: 1
        ▸ Renegotiation Info extension
      ▾ Extension: ec_point_formats (len=4)
          Type: ec_point_formats (11)
```

Figure 12 Comparison of supported TLS Cipher Suites

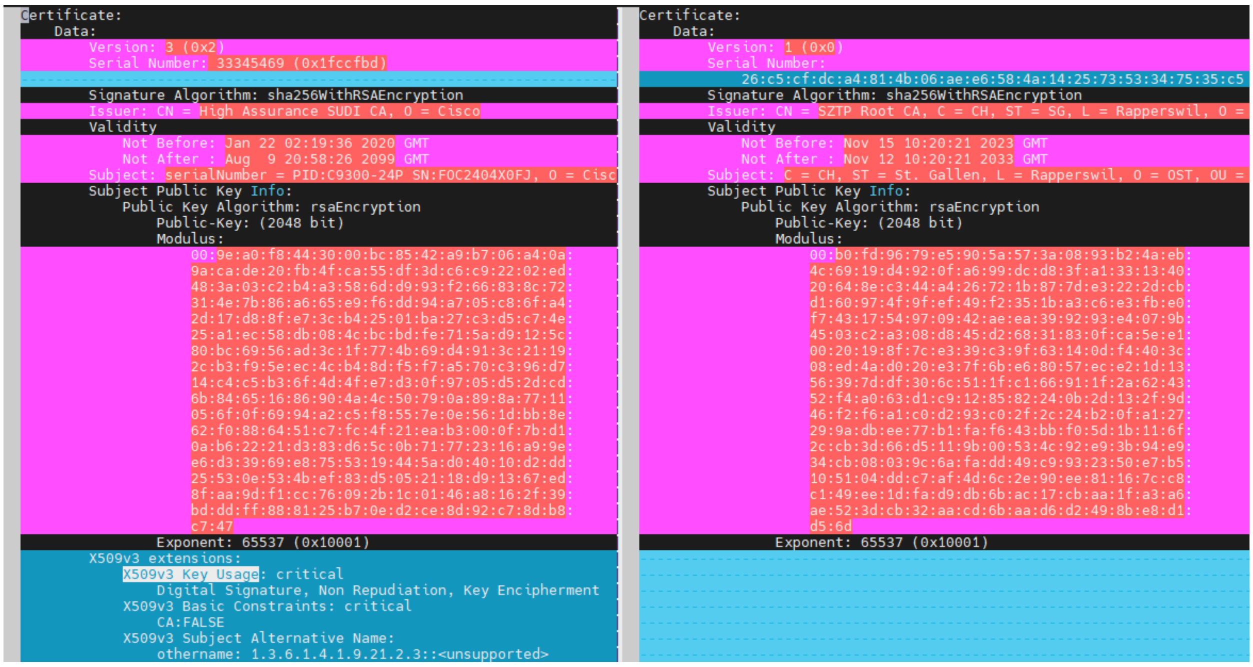
The team received another hint from our advisors on a setting in Python's SSL library that lets the developer log information on the secret keys in a TLS session. In their tests however, they could find no benefit of it: With their own CA and client certificate, they got the log messages. If they switched to the cisco device as client and the corresponding CA chain, the log file stayed empty. Thus proving, that there is a problem, but not bringing them any closer to the solution.



```
[~/application]$ cat secrets.log
# TLS secrets log file, generated by OpenSSL / Python
SERVER_HANDSHAKE_TRAFFIC_SECRET 7692fd1a3a4e5d5f4f4d0bcf1f0d2d168b0315a0a82168f19080f337e7dfca9e 5d07c1992bbb65510b9867f88097ea6beae89832bdee2a1f22fdbae244a77
CLIENT_HANDSHAKE_TRAFFIC_SECRET 7692fd1a3a4e5d5f4f4d0bcf1f0d2d168b0315a0a82168f19080f337e7dfca9e d33be02fa1b29dd182510ce7cff85995e26b18c65ab9b589633f356a08793
EXPORTER_SECRET 7692fd1a3a4e5d5f4f4d0bcf1f0d2d168b0315a0a82168f19080f337e7dfca9e 11caab51bbf29a98de8e87c3eea8397fc6f6ca38cb7f272bc6faf17871a584951327e2d7e78e5
SERVER_TRAFFIC_SECRET_0 7692fd1a3a4e5d5f4f4d0bcf1f0d2d168b0315a0a82168f19080f337e7dfca9e 4dc1b0340a6d26e183ea79a3339e789571dda6a5d9fc7b6323b82de37934da0ca5841
SERVER_HANDSHAKE_TRAFFIC_SECRET 920eaded00e733c9dc698bb60881a77b1b9bc58a2e1f0810358fcc87c2a53013 7b50573384c04e7b3ba24bdcb1d0249aeb09a7e2a2828ac894a8f846374fb
CLIENT_HANDSHAKE_TRAFFIC_SECRET 920eaded00e733c9dc698bb60881a77b1b9bc58a2e1f0810358fcc87c2a53013 f8b716939e35d94f5239d5f1a7e8850906c85573b693d665eace0e4f6d1ae
EXPORTER_SECRET 920eaded00e733c9dc698bb60881a77b1b9bc58a2e1f0810358fcc87c2a53013 facc12bd672ad569f6b9168e7ed14cf968e5b349f583d0b025eb7821a75af1bd93b9116387b6d
SERVER_TRAFFIC_SECRET_0 920eaded00e733c9dc698bb60881a77b1b9bc58a2e1f0810358fcc87c2a53013 9ead8d6a4420a40ccb401db2df3671b01d3a17e1c2ac70670f546fc36363d422c5a69
CLIENT_TRAFFIC_SECRET_0 920eaded00e733c9dc698bb60881a77b1b9bc58a2e1f0810358fcc87c2a53013 9ac4ebd02796ae82007d84fa989405946036f5f7f1602f7554807e0998e926fad3567
```

Figure 13 SSL library secret log for client test certificate

The final idea was to compare the devices client certificate with their own test certificate in terms of structure and values. The comparison showed that the structure and chosen algorithms were mostly identical. However, the Cisco certificate contained some parameters of the X509v3 extensions. Of particular interest to the team was the Key Usage parameter. They then created a new client test certificate with their own CA that contained the mentioned X509v3 extension parameters and tested this one, expecting the test to fail. Unfortunately, the test succeeded, proving that the certificate's structure is not the problem.



```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 33345469 (0x1fccfbd)

        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN = High Assurance SUDI CA, O = Cisco
        Validity
            Not Before: Jan 22 02:19:36 2020 GMT
            Not After : Aug  9 20:58:26 2099 GMT
        Subject: serialNumber = PID:C9300-24P SN:FOC2404X0FJ, O = Cisc
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:9e:a0:f8:44:30:00:bc:85:42:a9:b7:06:a4:0a:
                    9a:ca:de:20:fb:4f:ca:55:df:3d:c6:c9:22:02:ed:
                    48:3a:03:c2:b4:a3:58:6d:d9:93:f2:66:83:8c:72:
                    31:4e:7b:86:a6:65:e9:f6:dd:94:a7:05:c8:6f:a4:
                    2d:17:d8:8f:e7:3c:b4:25:01:ba:27:c3:d5:c7:4e:
                    25:a1:ec:58:db:08:4c:bc:bd:fe:71:5a:d9:12:5c:
                    80:bc:69:56:ad:3c:1f:77:4b:69:d4:91:3c:21:19:
                    2c:b3:f9:5e:ec:4c:b4:8d:f5:f7:a5:70:c3:96:d7:
                    14:c4:c5:b3:6f:4d:4f:e7:d3:0f:97:05:d5:2d:cd:
                    6b:84:65:16:86:90:4a:4c:50:79:0a:89:8a:77:11:
                    05:6f:0f:69:94:a2:c5:f8:55:7e:0e:56:1d:bb:8e:
                    62:f0:88:64:51:c7:fc:4f:21:ea:b3:00:0f:7b:d1:
                    0a:b6:22:21:d3:83:d6:5c:0b:71:77:23:16:a9:9e:
                    e6:d3:39:69:e8:75:53:19:44:5a:d0:40:10:d2:dd:
                    25:53:0e:53:4b:ef:83:d5:05:21:18:d9:13:67:ed:
                    8f:aa:9d:f1:cc:76:09:2b:1c:01:46:a8:16:2f:39:
                    bd:dd:ff:88:81:25:b7:0e:d2:ce:8d:92:c7:8d:b8:
                    c7:47
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Non Repudiation, Key Encipherment
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Subject Alternative Name:
                othername: 1.3.6.1.4.1.9.21.2.3::<unsupported>
```

```
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number:
            26:c5:cf:dc:a4:81:4b:06:ae:e6:58:4a:14:25:73:53:34:75:35:c5
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN = SZTP Root CA, C = CH, ST = SG, L = Rapperswil, O =
        Validity
            Not Before: Nov 15 10:20:21 2023 GMT
            Not After : Nov 12 10:20:21 2033 GMT
        Subject: C = CH, ST = St. Gallen, L = Rapperswil, O = OST, OU =
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:b0:fd:96:79:e5:90:5a:57:3a:08:93:b2:4a:eb:
                    4c:69:19:d4:92:0f:a6:99:dc:d8:3f:a1:33:13:40:
                    20:64:8e:c3:44:a4:26:72:1b:87:7d:e3:22:2d:cb:
                    d1:60:97:4f:9f:ef:49:f2:35:1b:a3:c6:e3:fb:e0:
                    f7:43:17:54:97:09:42:ae:ea:39:92:93:e4:07:9b:
                    45:03:c2:a3:08:d8:45:d2:68:31:83:0f:ca:5e:e1:
                    00:20:19:8f:7c:e3:39:c3:9f:63:14:0d:f4:40:3c:
                    08:ed:4a:d0:20:e3:7f:6b:e6:80:57:ec:e2:1d:13:
                    56:39:7d:df:30:6c:51:1f:c1:66:91:1f:2a:62:43:
                    52:f4:a0:63:d1:c9:12:85:82:24:0b:2d:13:2f:9d:
                    46:f2:f6:a1:c0:d2:93:c0:2f:2c:24:b2:0f:a1:27:
                    29:9a:db:ee:77:b1:fa:f6:43:bb:f0:5d:1b:11:6f:
                    2c:cb:3d:66:d5:11:9b:00:53:4c:92:e9:3b:94:e9:
                    34:cb:08:03:9c:6a:fa:dd:49:c9:93:23:50:e7:b5:
                    10:51:04:dd:c7:af:4d:6c:2e:90:ee:81:16:7c:c8:
                    c1:49:ee:1d:fa:d9:db:6b:ac:17:cb:aa:1f:a3:a6:
                    ae:52:3d:cb:32:aa:cd:6b:aa:d6:d2:49:8b:e8:d1:
                    d5:6d
                Exponent: 65537 (0x10001)
```

Figure 14 Cisco device certificate (left) and original test client certificate (right)

This problem has already cost the team a lot of time, so they finally decided to stop investigating it for now. Instead, they focused on other parts of the project, like completing the workflow of the application and integrating the Netbox API, leave the device communication with only a server-side certificate and no serial number extraction of the device certificate.

The code providing the functionality for the mTLS authentication and device certificate parsing is left in the source code file with corresponding comments for further development.

6.1.5 Getting Docker Containers to the lab

To set up the inventory, Netbox, can be configure and customize from scratch. Fortunately, there is a Docker Netbox image, so a lot of effort can be saved.

However, it is not as easy as thought by the team. They have tried to install the Docker Netbox image in a lab environment. The lab computers have two namespaces, one of which is connected to the Internet and the other to the test switch.

The first suggested solution was to attach a router (with internet access) between the test switch and the lab computer. The second one would be to download the Netbox images manually via the other browser on the lab computer. The team chose the second option because it's less complex and they would only need a docker image this one time.

The necessary docker images were transferred to the server manually. Luckily docker provides the commands "docker save" and "docker load" to export/import the images into a tar file. The team created the tar files on their development notebook, transferred it to the server and imported them again.

6.1.6 Testing the controller with a Cisco Switch

After the very time-consuming problems with the devices certificate, the decision was made not to use it for authentication. Since this certificate is also the only way to get the devices serial number at the time of bootstrapping, the decision was made to set it manually in the Provisioning Controller.

Having combined or mocked all other data sources the team tried to do a real-world test of the Provisioning Controller in combination with the physical switch. The results were that the device requests the bootstrapping data five times in a row but does not do anything else. Specifically, it neither tries to load the firmware image nor applies the supplied configuration. This behavior lead them to the conclusion, that there must be an error somewhere in the structure of our bootstrapping data.

Unfortunately, the device itself does not report any errors. Neither to its internal log nor via the "bootstrap-error" progress report specified in the SZTP RFC. This is likely due to the RFC allowing error progress reports only to trusted servers and would imply the error lying somewhere in the server authentication.

Since this is accomplished via the ownership voucher from Cisco, which is not human readable, and without any further error details available from logs, chances of resolving this problem in any reasonable amount of time are practically nonexistent. It was therefore decided, that for this project the device and its requests will be replaced by a mock.

7. Outlook

This project provides the basis of a Network Controller. While it contains the most basic functions such as initial provisioning and automatic configuration backup, it also lacks some functionality such as configuring devices in their operation phase and collecting performance statistics. In a future project, extensions like these could be developed.

On the other hand, there is the mutual TLS authentication, which we couldn't implement successfully due to time constraints. Also, there's the interaction with the real physical device which is still to be implemented. As a continuation of this project, these aspects would need to be addressed as well.

This application relies heavily on Netbox as a data source and is written in the same language (Python). So, there would also be the possibility to transform its logic into a plugin, that could be integrated in Netbox and used from there.

8. Project Plan

8.1 Roles and obligations

8.2 Process Model

It was decided to use in this project the Kanban-Method for the short-term planning and the RUP Process model for the long-term planning. The decision is based on the fact, that the Kanban method is easy to use, the team is small so there is no need for extra complexity, and Kanban is already known from the SEP project. The RUP Process Model is also already known and found to be self-comprehending by the team.

8.3 Time Management

8.3.1 Long-Term Planning

The Long-Term Planning is for the whole Semester. The project phases Inception, Elaboration, Construction and Transition are mapped in the diagram. It also includes all the milestones. The plan is separated per week. Below is the RUP plan.

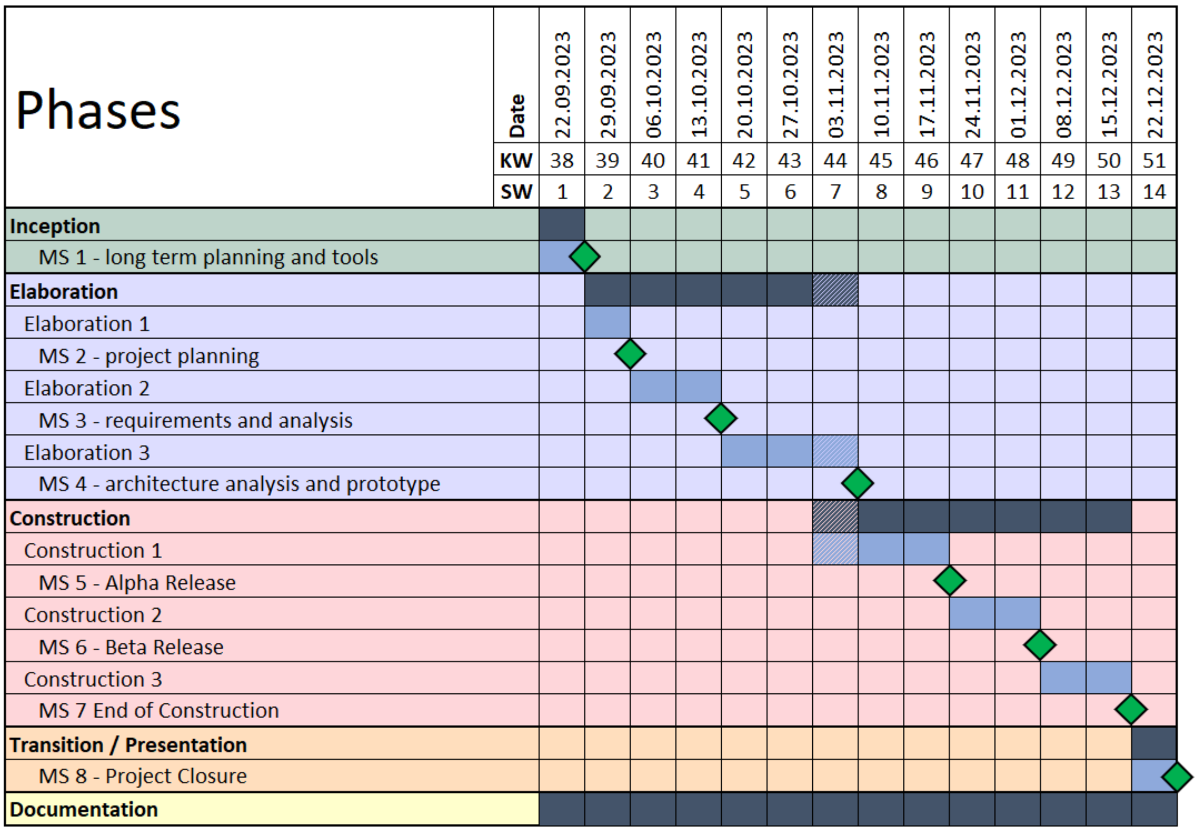


Figure 15 Long-Term Plan

Below are some important symbols according to Figure 15 Long-Term Plan.

- ▨ In the diagram the striped boxes are optional and can be used if the time is needed. In case the team doesn't need the time, they can procced with the next phase.
- ◆ The green squares are milestones.

The project delivery is in KW 51, on Friday at 5:00pm.

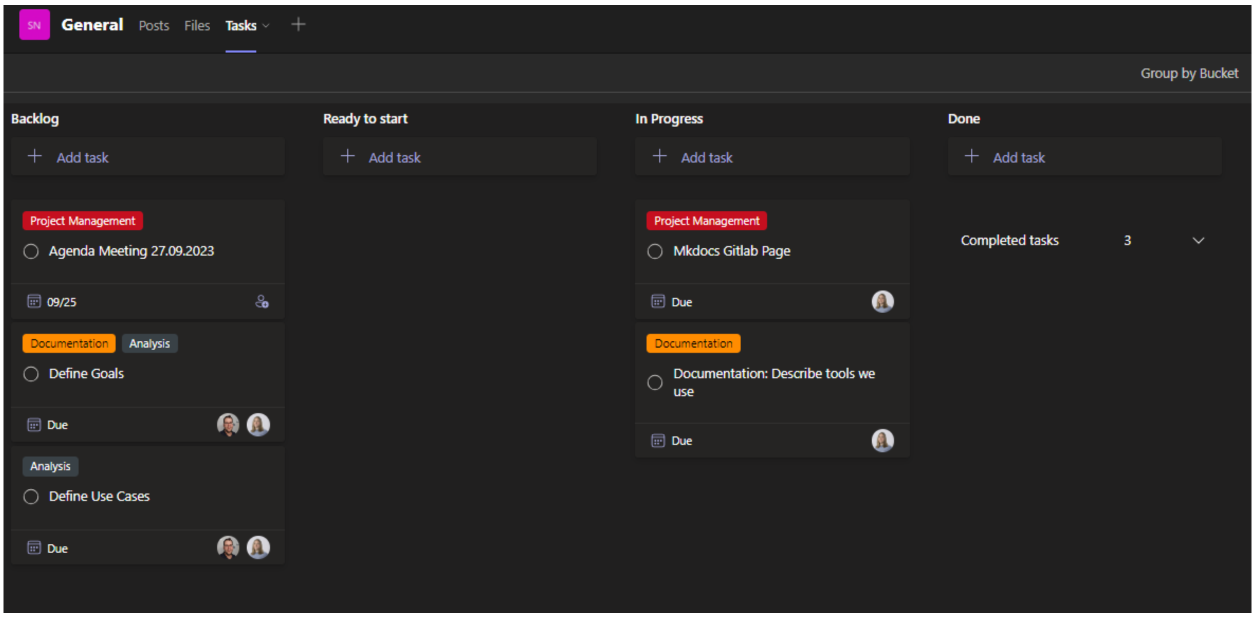
8.3.2 Short-Term Planning

For the short-term planning the Kanban-Method is used. It was decided to use the tool in Teams, "Tasks". It is best suited for the team, because it's already known from work and the team didn't want to use many different tools, keeping it simple and clean.

8.4 Collaboration

8.4.1 Communication

For the communication between the two developers, MS Teams is used.



Backlog

- Project Management — Agenda Meeting 27.09.2023 — 09/25
- Documentation, Analysis — Define Goals — Due
- Analysis — Define Use Cases — Due

Ready to start

In Progress

- Project Management — Mkdocs Gitlab Page — Due
- Documentation — Documentation: Describe tools we use — Due

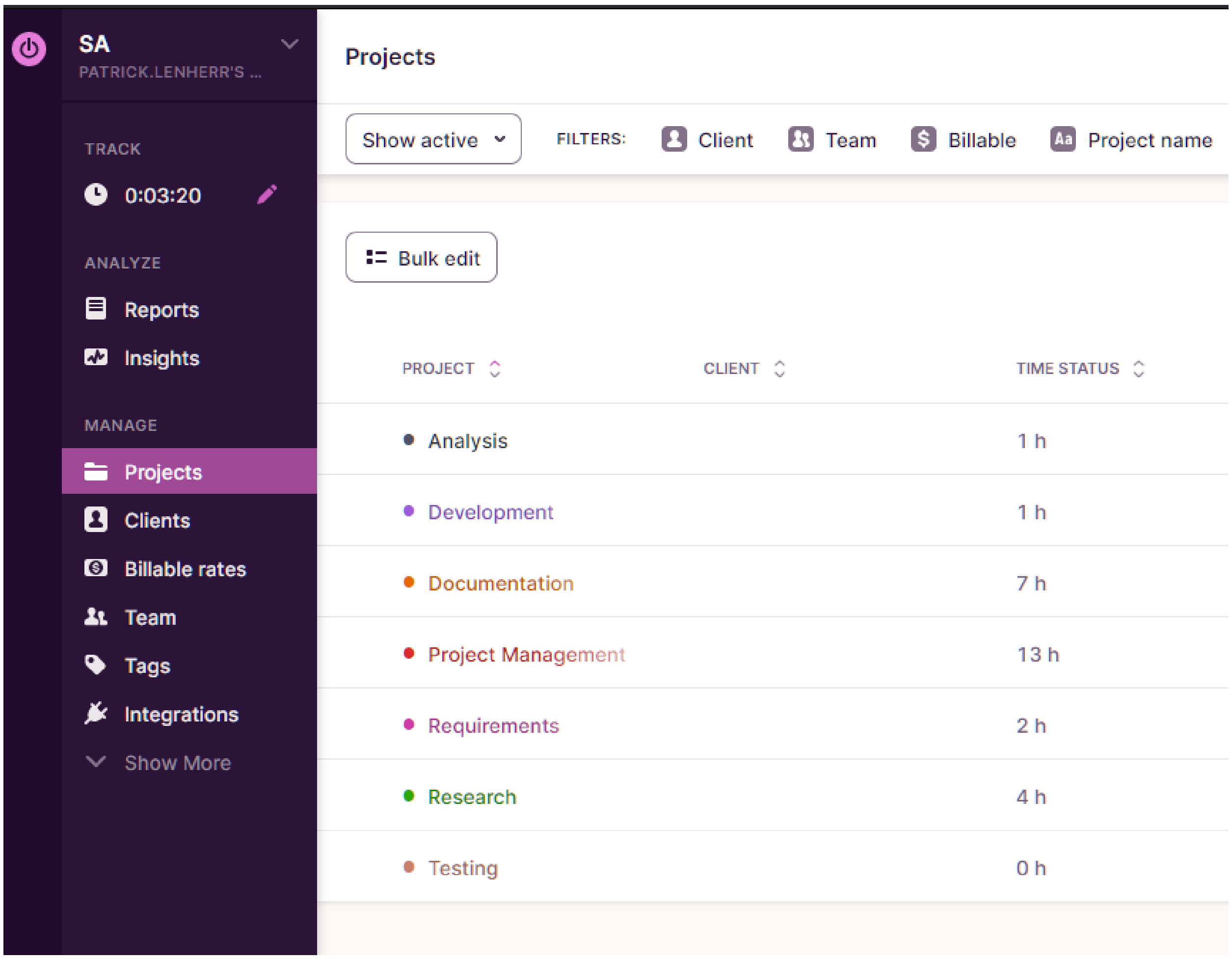
Done

Figure 16 Short-Term Planning with MS Teams

In MS Teams the team has defined the Buckets Backlog, Ready to start, In Progress and Done. Team members and due dates can be assigned to tasks.

8.4.2 Time tracking

To track the time, Toggl is used. In Toggl various projects were created to represent the different disciplines. Toggl allows the team to automatically create reports grouping time by discipline and team member as well as filter by time range. Below is a screenshot of out Toggl time tracking.



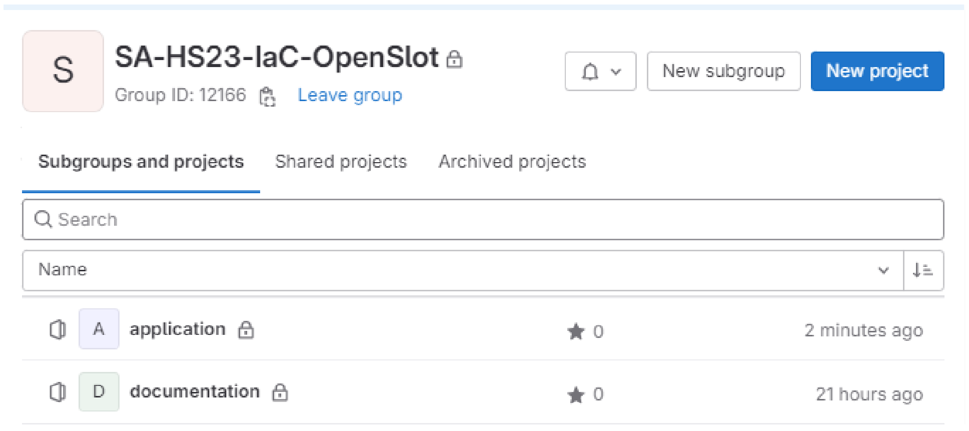
PROJECT	CLIENT	TIME STATUS
Analysis		1 h
Development		1 h
Documentation		7 h
Project Management		13 h
Requirements		2 h
Research		4 h
Testing		0 h

Figure 17 Toggl Track for time tracking

8.4.3 File storge, versioning and review

For this is the OST Gitlab used. To ensure the expected code quality, each merge request is assigned to the other team member for review. This review ensures also that the code to be merged is structured in a comprehensible fashion. In addition, Mkdocs is used for the technical documentation.

In Gitlab, the team used a group with two different repositories "application" and "documentation".



A	application	0	2 minutes ago
D	documentation	0	21 hours ago

Figure 18 Gitlab repositories

8.5 Meetings

Meetings are set up on Wednesday, 11:00am with the supervisors Urs Baumann and Yannick Zwicker and the developers Patrick Lenherr and Vanessa Gyger.

9. Appendix

9.1 Glossary

Abbreviation	Name	Description
CA	Certification Authority	Service to sign certificates
DNA Center	Digital Network Architecture Center	Centralized network management solution by Cisco
MASA Server	Manufacturer Authorized Signing Authority	Service to generate Ownership Vouchers
OC	Owner Certificate	This is an X.509 certificate. This certificate is used for identifying the organization
OV	Ownership Voucher	Used to securely identify the devices associated owner organization.
PDC	pinned-domain-certificate	The PDC is used as a trust anchor in a certificate chain that does not lead up to a root CA.
PK	Private Key	Is used with the PDC to generate the Ownership certificate.
PnP	Plug-and-Play	Cisco Service to configure network devices
RestConf	RESTful Network Configuration Protocol	HTTP-based protocol providing an interface to access YANG data
SSoT	Single Source of Truth	Concept of keeping any and all configuration data in one location
SUDI	Secure Unique Device Identifier	Device identity (product identifier and serial number) in a certificate that is chained to a root CA
SZTP	Secure Zero Touch Provisioning	Protocol standardized in RFC8572 to Provision network devices
TAM	Trust Anchor Module	Hardware module containing the cryptographic secrets
YANG	Yet Another Next Generation	Data modeling language for network management protocols

Figure 19 Glossary

9.2 List of Figures

9.3 Bibliography

[1] Cisco. "Cisco Catalyst and DNA Software Subscription Matrix for Switching." [Online]. Available: https://www.cisco.com/c/m/en_us/products/software/dna-subscription-switching/en-sw-sub-matrix-switching.html

[2] Cisco. "Cisco Catalyst Center Network Managment." [Online]. Available: https://www.cisco.com/site/us/en/products/networking/dna-center-platform/index.html?dtid=osscdc000283

[3] K. Watsen, M. Abrahamsson, and Farrer Ian. "Secure Zero Touch Provisioning (SZTP)." [Online]. Available: https://datatracker.ietf.org/doc/rfc8572/

[4] Ford Arad. "Network Automation with Plug and Play (PnP) – Part 1." [Online]. Available: https://community.cisco.com/t5/networking-blogs/network-automation-with-plug-and-play-pnp-part-1/ba-p/3658231

[5] Cisco. "CiscoLive Innovations SZTP." [Online]. Available: https://www.ciscolive.com/c/dam/r/ciscolive/us/docs/2021/pdf/BRKSPG-2024.pdf

[6] Cisco. "Programmability Configuration Guide, Cisco IOS XE Dublin 17.11.x." Accessed: Oct. 19, 2023. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1711/b_1711_programmability_cg/m_1711_prog_ztp.html#secure-ztp

[7] Wikipedia. "Golang." [Online]. Available: https://de.wikipedia.org/wiki/Go_(Programmiersprache)

[8] Watson. "SZTP Python Framework." [Online]. Available: https://pypi.org/project/sztpd/

[9] Boris Glimcher. "opiproject/sztp." [Online]. Available: https://github.com/opiproject/sztp

[10] STEPHEN CASS. "Top Programming Languages 2022." [Online]. Available: https://spectrum.ieee.org/top-programming-languages-2022

[11] Muhtasim Fuad Rafid. "FastAPI Pros and Cons." [Online]. Available: https://dev.to/fuadrafid/fastapi-the-good-the-bad-and-the-ugly-20ob

[12] tiangolo. "FastAPI." [Online]. Available: https://fastapi.tiangolo.com/

[13] shrubbery. "RANCID Backup." [Online]. Available: https://shrubbery.net/rancid/

[14] IONOS. "gRPC." [Online]. Available: https://www.ionos.de/digitalguide/server/knowhow/grpc-vorgestellt/

[15] pyneng. "Netmiko." [Online]. Available: https://pyneng.readthedocs.io/en/latest/book/18_ssh_telnet/netmiko.html

[16] Hankins, D., Mrugalski, T., Jiang, S., and S. Krishnan. "Guidelines for Creating New DHCPv6 Options." [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7227

9.4 Assignment-Document

INS | Institute for Network and Security

Semester Project Assignment

Secure Device Provisioning Using SZTP

Version 1.0
December 20, 2023
Institute for Network and Security

1 Assignment

1.1 Supervisor

This student project will be supervised by Yannick Zwicker (yannick.zwicker@ost.ch) and Urs Baumann (urs.baumann@ost.ch), OST.

1.2 Students

This project is conducted in the context of the module "Studienarbeit" in the department "Informatik" by:

- Vanessa Gyger
- Patrick Lenherr

1.3 Introduction

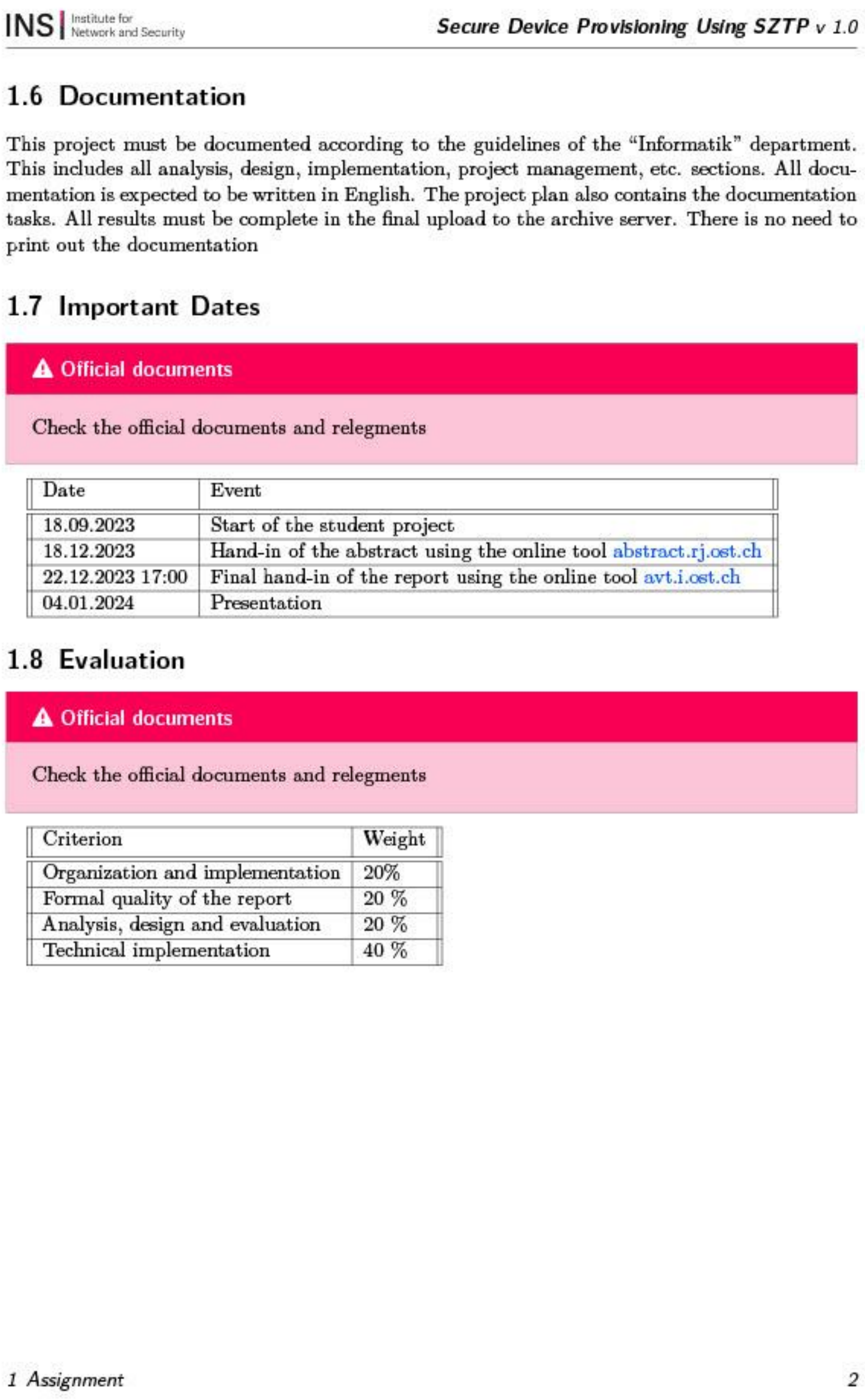
Network infrastructure consists of routers and switches. The larger the network the more of these devices are present within the network. The number of such devices can thus become quickly so large, that managing them manually is not only tedious but also poses significant risks of inconsistent configuration and extended downtimes in case of hardware failure. Automation mechanisms for device and configuration management are therefore a necessity as soon as the networks surpass a certain size. Especially the initial device onboarding can be automated efficiently.

1.4 Status quo

Many large network device vendors have realized the demand for automation and are offering their own automation solutions, like Cisco's Catalyst Center. Most of these solutions are however not only restricted to the vendor's own devices but are also closed source and often too expensive for small to medium network operators. Some of them have come together with their idea of a standardized way to securely provision new devices without manual intervention on the device itself (zero-touch).

1.5 Goals of the Project

The objective of our term project is to lay the groundwork for a network controller that handles zero-touch provisioning of newly installed devices as well as transferring configuration in case of 1-to-1 device replacement. The scope of the controller is limited to Cisco devices for this project, but it should be extendable to support various devices that implement SZTP according to RFC 8572. Furthermore, the controller should also be open for future extension providing more functionality like ongoing configuration after the provisioning process.



1.6 Documentation

This project must be documented according to the guidelines of the "Informatik" department. This includes all analysis, design, implementation, project management, etc. sections. All documentation is expected to be written in English. The project plan also contains the documentation tasks. All results must be complete in the final upload to the archive server. There is no need to print out the documentation

1.7 Important Dates

⚠ Official documents

Check the official documents and relegments

Date	Event
18.09.2023	Start of the student project
18.12.2023	Hand-in of the abstract using the online tool abstract.rj.ost.ch
22.12.2023 17:00	Final hand-in of the report using the online tool avt.i.ost.ch
04.01.2024	Presentation

1.8 Evaluation

⚠ Official documents

Check the official documents and relegments

Criterion	Weight
Organization and implementation	20%
Formal quality of the report	20 %
Analysis, design and evaluation	20 %
Technical implementation	40 %

9.5 Implementation & User Guide from mkdocs

Online version: http://ins-stud.pages.gitlab.ost.ch/sa-ba/sa-hs23-iac-openslot/documentation/

Secure Device Provisioning Using SZTP

Description	Secure Device Provisioning Using SZTP
Author(s)	Patrick Lenherr, Vanessa Gyger
Copyright	Copyright © Patrick Lenherr, Vanessa Gyger

Table of Contents

1 Secure Device Provisioning Using SZTP

The objective of this term project is to lay the groundworks for a network controller that handles zero touch provisioning of newly installed devices as well as transferring configuration in case of 1-to-1 device replacement.

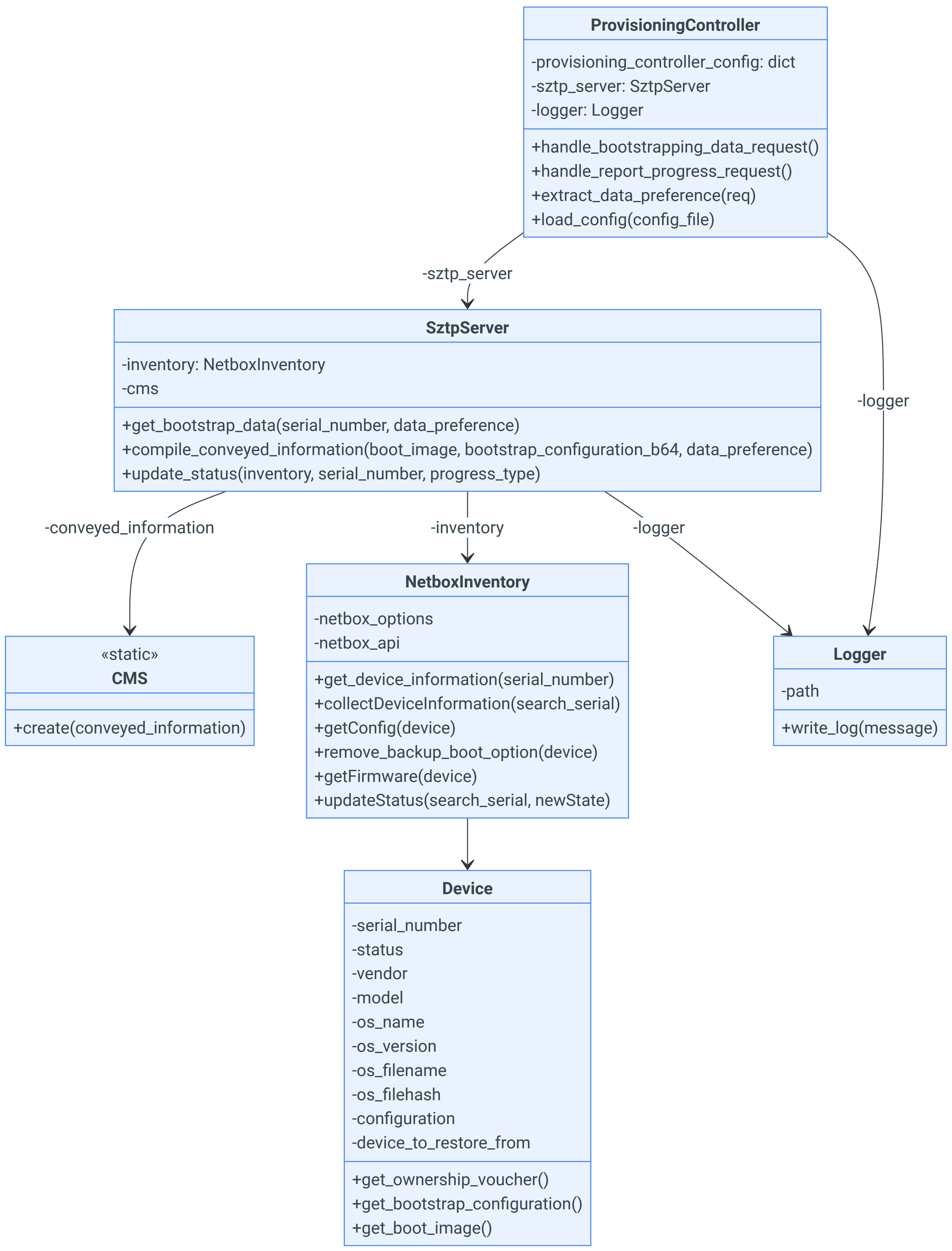
The scope of the controller is limited to Cisco devices for this project, but it should be extendable to support various devices which implement SZTP according to RFC 8572.

I. Architecture & Design

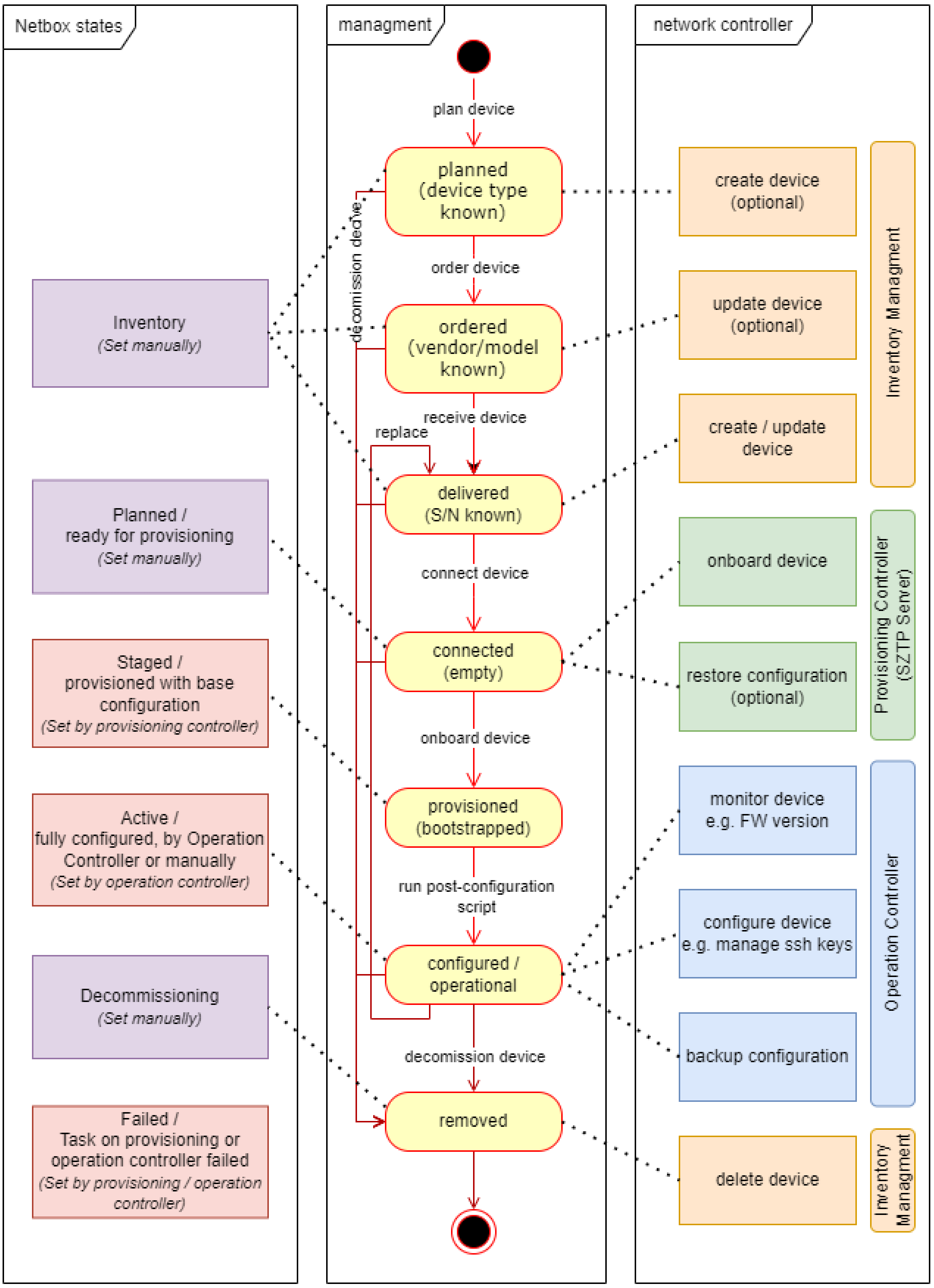
2 SZTP Controller diagram

2.1 Class Diagram

The following figure focuses on the classes which are implemented for the network device controller component. Interfaces (or interface-like constructs for Python) are used to allow for easy extension of the application. Contrary to the philosophy of SZTP – which is to request an ownership voucher from the vendor, this application treats it as an attribute to the device since it is provided as file for the scope of this term project.


ProvisioningController
-provisioning_controller_config: dict
-sztp_server: SztpServer
-logger: Logger
+handle_bootstrapping_data_request()
+handle_report_progress_request()
+extract_data_preference(req)
+load_config(config_file)
-sztp_server
-logger
SztpServer
-inventory: NetboxInventory
-cms
+get_bootstrap_data(serial_number, data_preference)
+compile_conveyed_information(boot_image, bootstrap_configuration_b64, data_preference)
+update_status(inventory, serial_number, progress_type)
-conveyed_information
-inventory
-logger
«static»
CMS
+create(conveyed_information)
NetboxInventory
-netbox_options
-netbox_api
+get_device_information(serial_number)
+collectDeviceInformation(search_serial)
+getConfig(device)
+remove_backup_boot_option(device)
+getFirmware(device)
+updateStatus(search_serial, newState)
Logger
-path
+write_log(message)
Device
-serial_number
-status
-vendor
-model
-os_name
-os_version
-os_filename
-os_filehash
-configuration
-device_to_restore_from
+get_ownership_voucher()
+get_bootstrap_configuration()
+get_boot_image()

2.2 Device Lifecycle



Netbox states
managment
network controller
plan device
planned (device type known)
order device
ordered (vendor/model known)
receive device
replace
delivered (S/N known)
connect device
connected (empty)
onboard device
provisioned (bootstrapped)
run post-configuration script
configured / operational
decomission device
removed
decomission device
Inventory (Set manually)
Planned / ready for provisioning (Set manually)
Staged / provisioned with base configuration (Set by provisioning controller)
Active / fully configured, by Operation Controller or manually (Set by operation controller)
Decommissioning (Set manually)
Failed / Task on provisioning or operation controller failed (Set by provisioning / operation controller)
create device (optional)
update device (optional)
create / update device
Inventory Managment
onboard device
restore configuration (optional)
Provisioning Controller (SZTP Server)
monitor device e.g. FW version
configure device e.g. manage ssh keys
backup configuration
Operation Controller
delete device
Inventory Managment

2.3 Sequence Diagram

The following diagrams shows how the different components interact together.

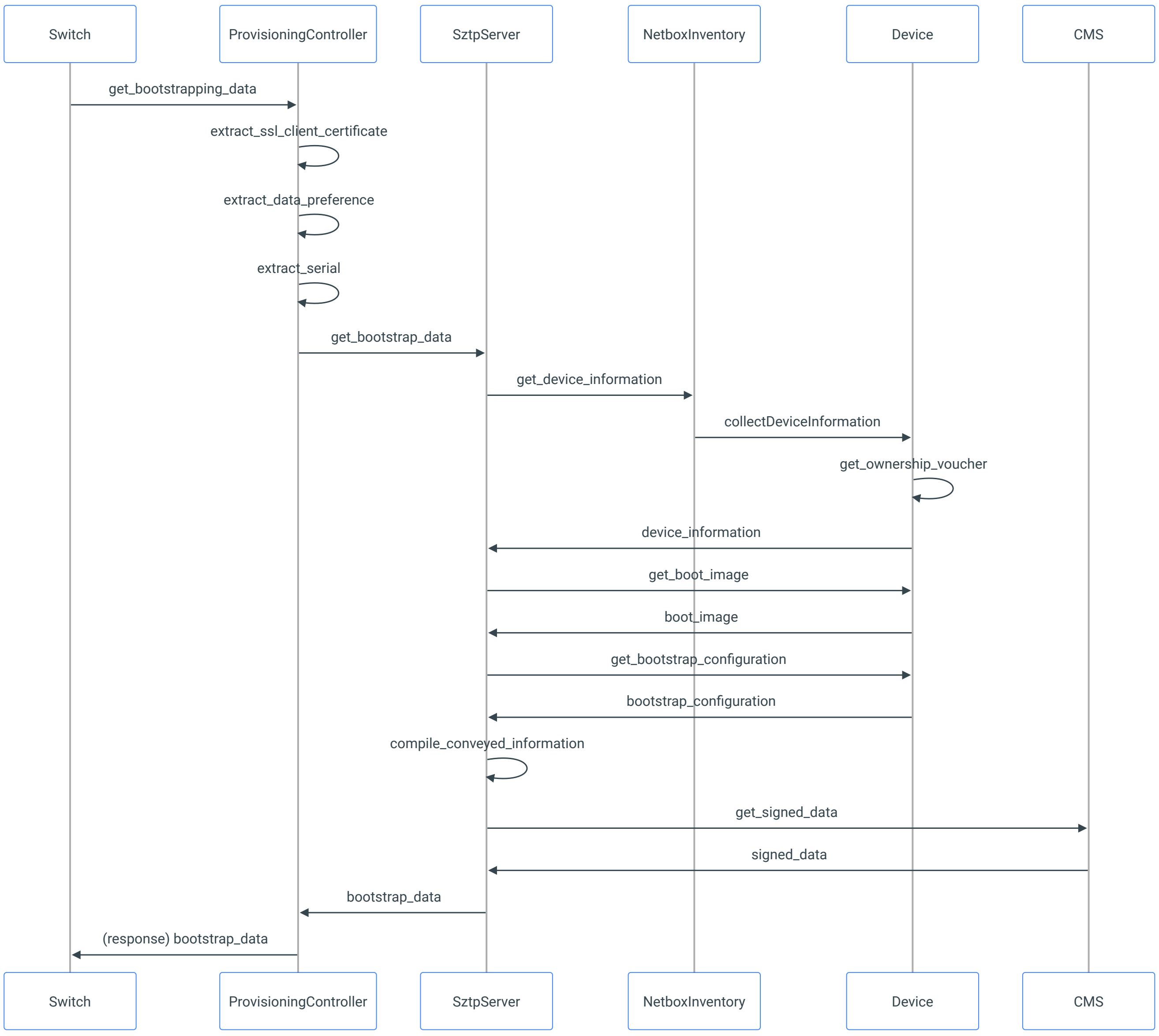
There are two main sequences, which are triggered by the device:

- Get bootstrapping data
- Report progress

For comprehensibility, the logger is not added in the following diagrams.

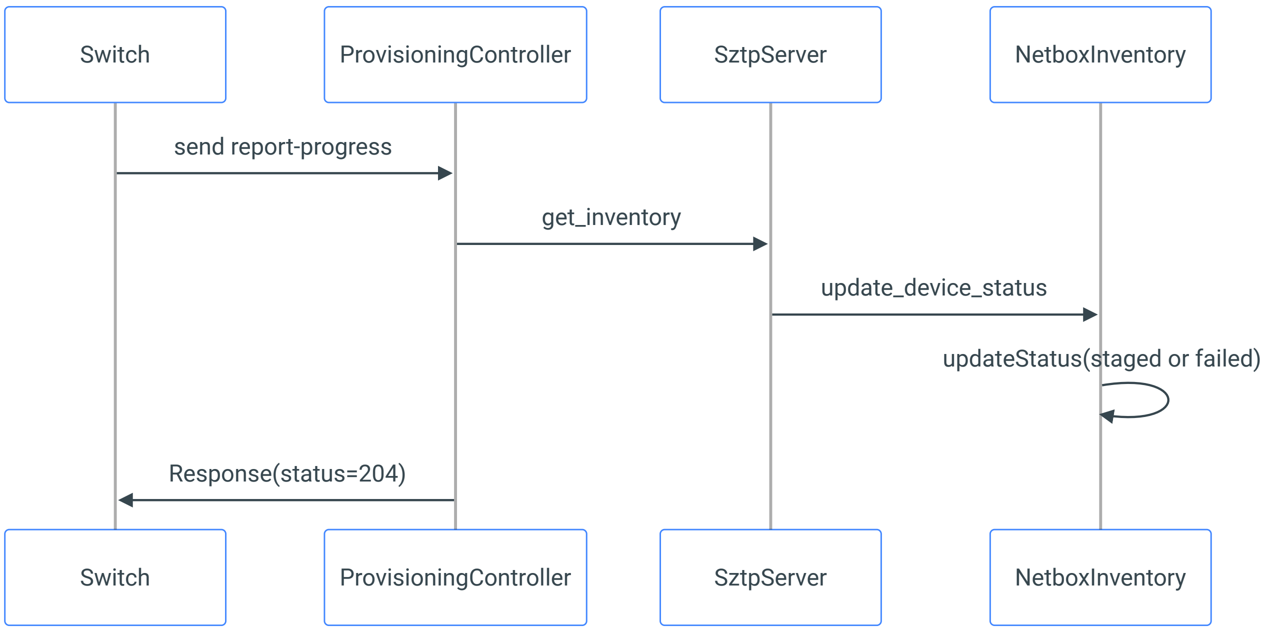
2.3.1 Get bootstrapping data

Below is the sequence diagram of how the bootstrapping data is collected, generated and sent to the device.



2.3.2 Report progress

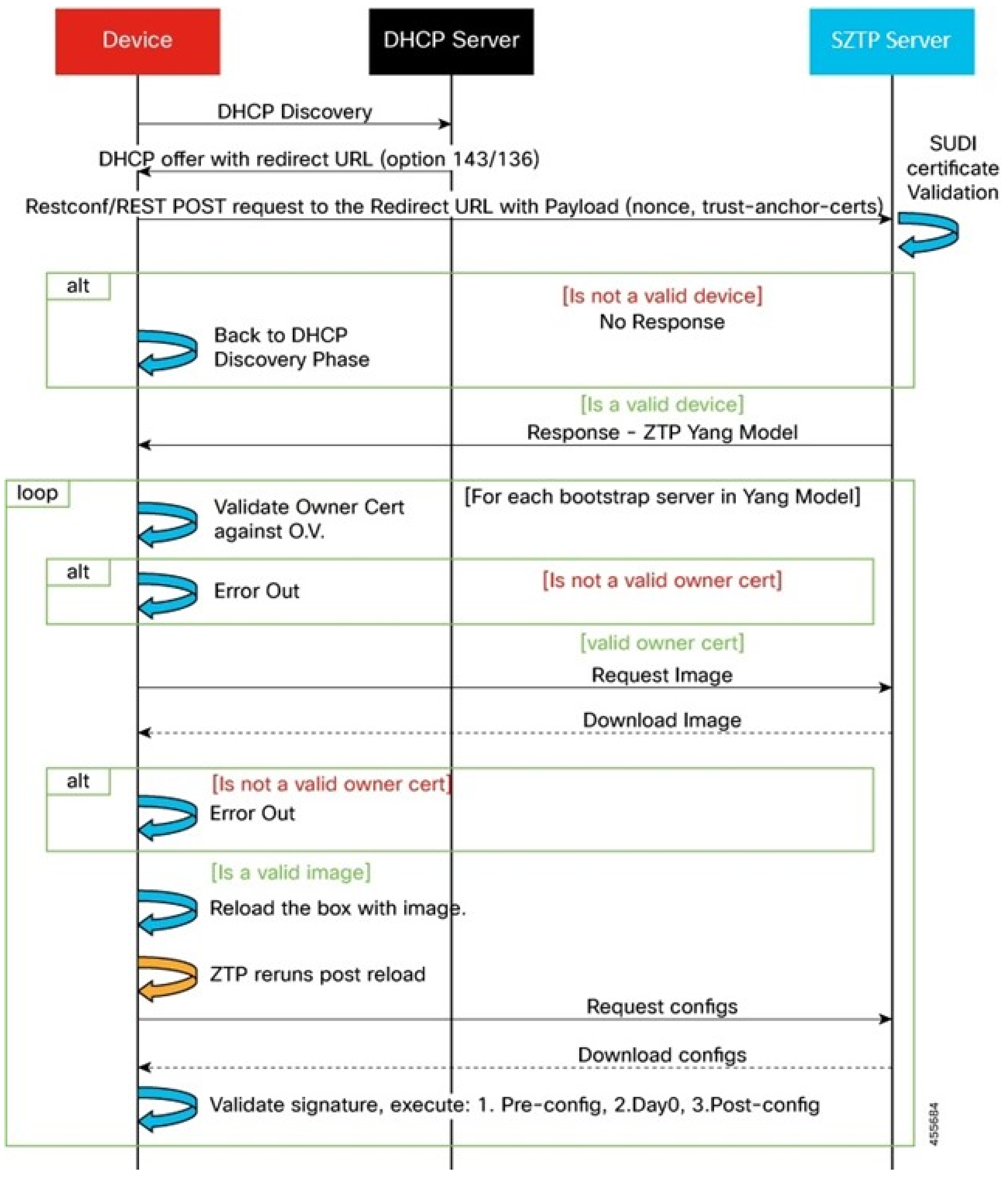
The report progress is sent from the device to the provisioning server. With this request, the status of the device is updated. If the provisioning was successful, the updated status in Netbox is set to "staged", otherwise to "failed".



2.4 SZTP-Workflow

2.4.1 General SZTP-Workflow

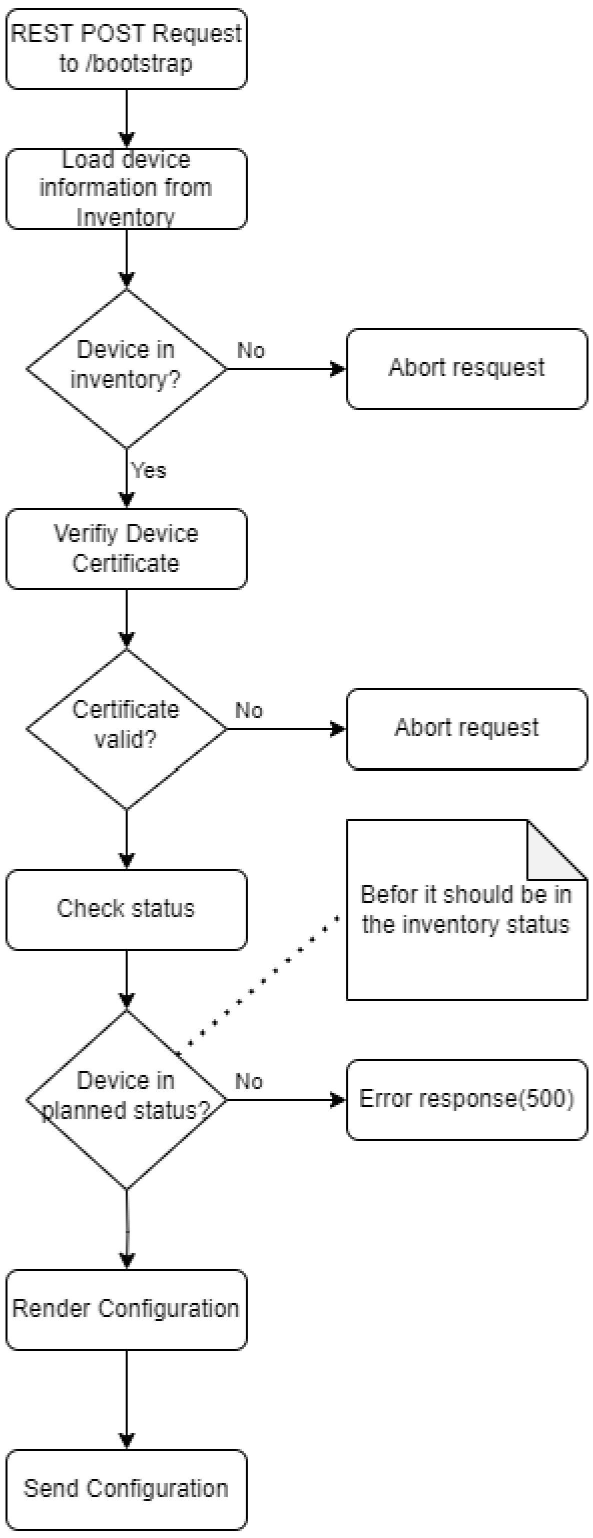
This is the device lifecycle is from Ciscos SZTP implementation. This is how they would implement the SZTP feature.


Device
DHCP Server
SZTP Server
DHCP Discovery
DHCP offer with redirect URL (option 143/136)
Restconf/REST POST request to the Redirect URL with Payload (nonce, trust-anchor-certs)
SUDI certificate Validation
alt
[Is not a valid device]
No Response
Back to DHCP Discovery Phase
[Is a valid device]
Response - ZTP Yang Model
loop
Validate Owner Cert against O.V.
[For each bootstrap server in Yang Model]
alt
Error Out
[Is not a valid owner cert]
[valid owner cert]
Request Image
Download Image
alt
[Is not a valid owner cert]
Error Out
[Is a valid image]
Reload the box with image.
ZTP reruns post reload
Request configs
Download configs
Validate signature, execute: 1. Pre-config, 2.Day0, 3.Post-config
455684

SZTP Workflow from Cisco

2.4.2 Adoption of SZTP-Workflow

This is the adopted finished workflow of how the device gets its bootstrapping data.

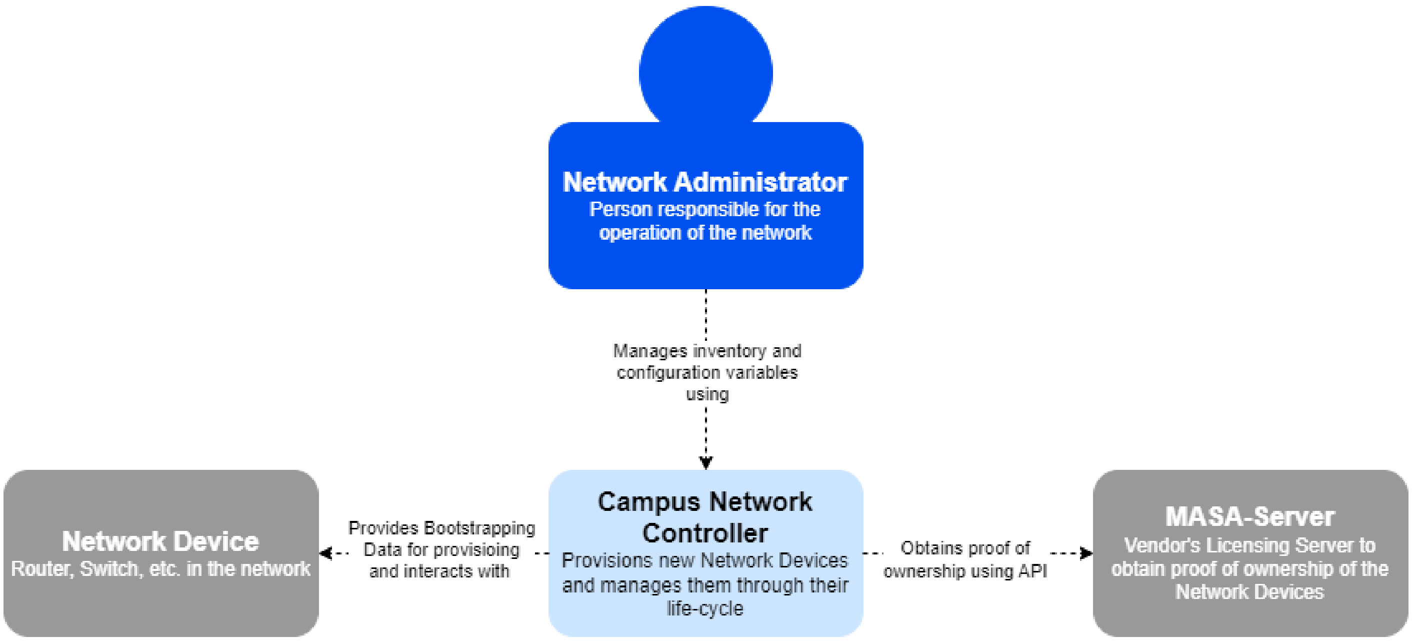

REST POST Request to /bootstrap
Load device information from Inventory
Device in inventory?
No
Abort resquest
Yes
Verifiy Device Certificate
Certificate valid?
No
Abort request
Check status
Befor it should be in the inventory status
Device in planned status?
No
Error response(500)
Render Configuration
Send Configuration

3 Application Architecture C4 Model

3.1 System Context

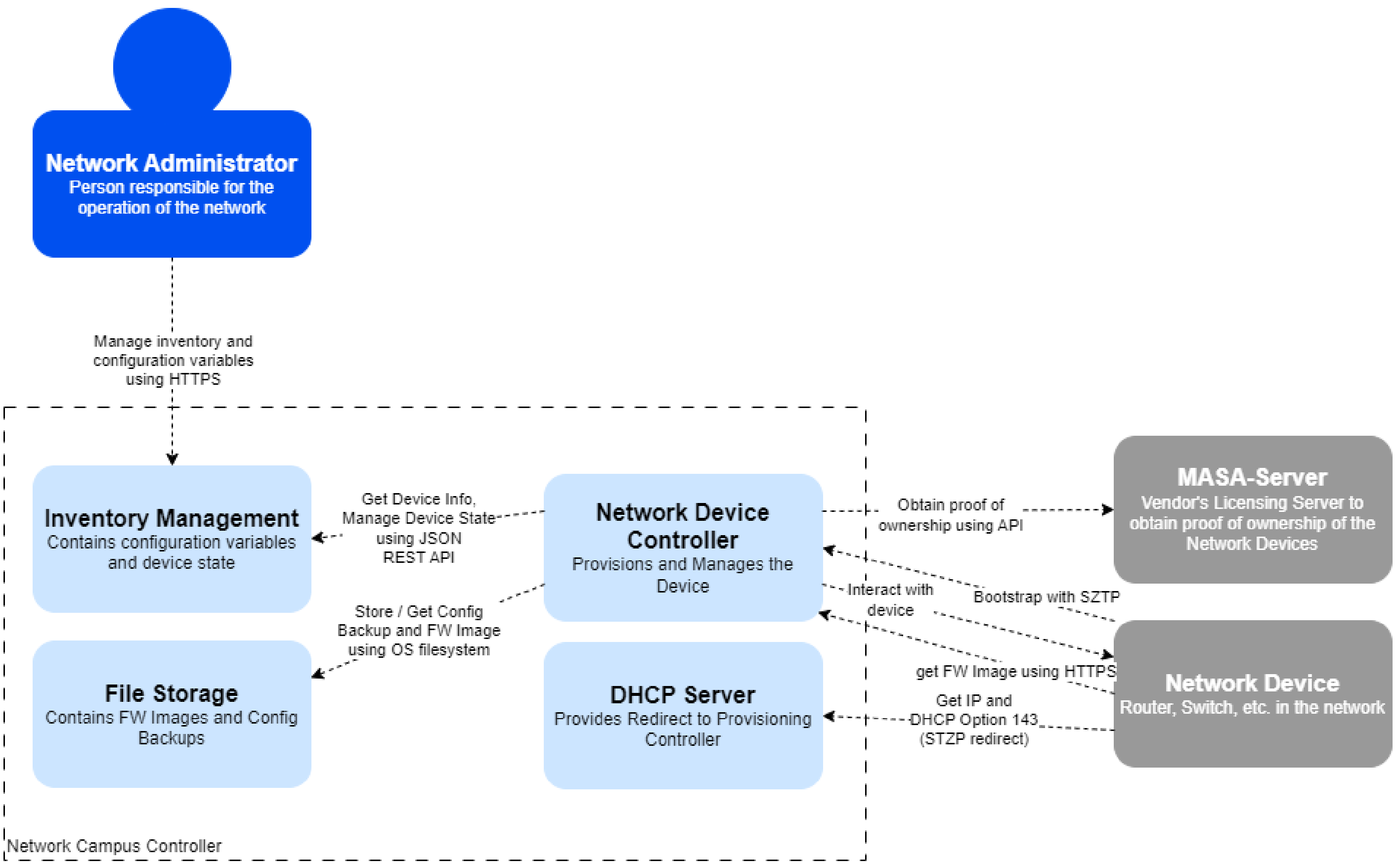
The following figure shows the context of the Campus Network Controller. The controller is used by the Network administrator. The controller itself in turn uses a Licensing Server to get the Ownership Voucher needed to authenticate itself to the device which should be provisioned.

Additional Information: In the diagrams below, the connection via the API to the MASA-Server is not part of the project – this was done manually. More information on this decision can be found in the project documentation, in the evaluation section. This applies to the further diagrams in this chapter as well.


Network Administrator
Person responsible for the operation of the network
Manages inventory and configuration variables using
Network Device
Router, Switch, etc. in the network
Provides Bootstrapping Data for provisioing and interacts with
Campus Network Controller
Provisions new Network Devices and manages them through their life-cycle
Obtains proof of ownership using API
MASA-Server
Vendor's Licensing Server to obtain proof of ownership of the Network Devices

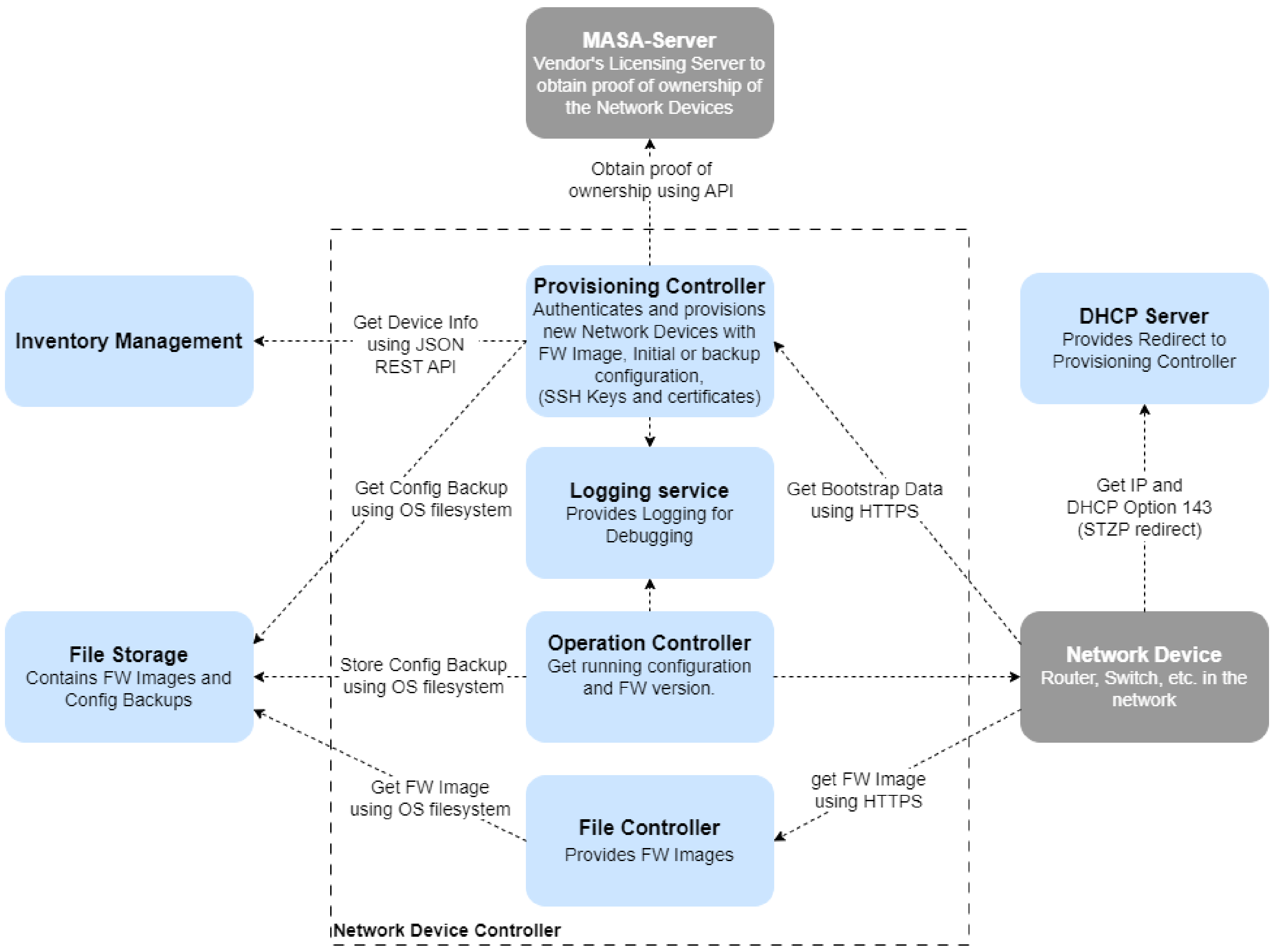
3.2 Container

The following figure shows the application with its core piece – the Network Device Controller – and its additional services for DHCP, Inventory, File Storage and Logging.


Network Administrator
Person responsible for the operation of the network
Manage inventory and configuration variables using HTTPS
Inventory Management
Contains configuration variables and device state
Get Device Info, Manage Device State using JSON REST API
Network Device Controller
Provisions and Manages the Device
Store / Get Config Backup and FW Image using OS filesystem
File Storage
Contains FW Images and Config Backups
DHCP Server
Provides Redirect to Provisioning Controller
Obtain proof of ownership using API
MASA-Server
Vendor's Licensing Server to obtain proof of ownership of the Network Devices
Interact with device
Bootstrap with SZTP
get FW Image using HTTPS
Get IP and DHCP Option 143 (STZP redirect)
Network Device
Router, Switch, etc. in the network
Network Campus Controller

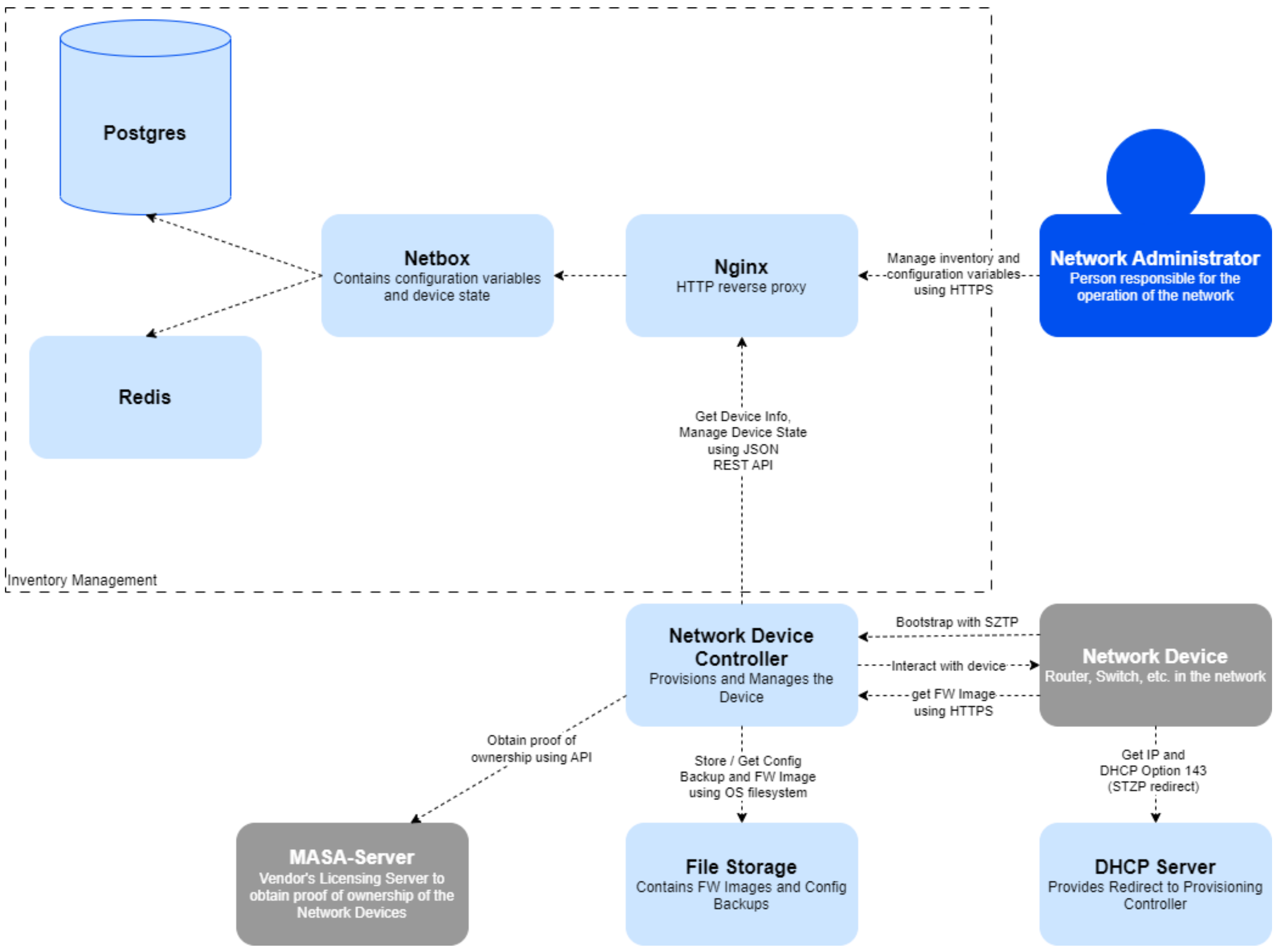
3.3 Network device controller component

The following figure focuses on the core component containing Provisioning Controller, File Controller and Operation Controller.


MASA-Server
Vendor's Licensing Server to obtain proof of ownership of the Network Devices
Obtain proof of ownership using API
Inventory Management
Get Device Info using JSON REST API
Provisioning Controller
Authenticates and provisions new Network Devices with FW Image, Initial or backup configuration, (SSH Keys and certificates)
DHCP Server
Provides Redirect to Provisioning Controller
Get Config Backup using OS filesystem
Logging service
Provides Logging for Debugging
Get Bootstrap Data using HTTPS
Get IP and DHCP Option 143 (STZP redirect)
File Storage
Contains FW Images and Config Backups
Store Config Backup using OS filesystem
Operation Controller
Get running configuration and FW version.
Network Device
Router, Switch, etc. in the network
Get FW Image using OS filesystem
File Controller
Provides FW Images
get FW Image using HTTPS
Network Device Controller

3.4 Inventory management component

The following figure focuses on the Inventory Management for the Network Campus Controller.



Postgres
Netbox
Contains configuration variables and device state
Nginx
HTTP reverse proxy
Manage inventory and configuration variables using HTTPS
Network Administrator
Person responsible for the operation of the network
Redis
Get Device Info, Manage Device State using JSON REST API
Inventory Management
Network Device Controller
Provisions and Manages the Device
Bootstrap with SZTP
Interact with device
get FW Image using HTTPS
Network Device
Router, Switch, etc. in the network
Obtain proof of ownership using API
Store / Get Config Backup and FW Image using OS filesystem
Get IP and DHCP Option 143 (STZP redirect)
MASA-Server
Vendor's Licensing Server to obtain proof of ownership of the Network Devices
File Storage
Contains FW Images and Config Backups
DHCP Server
Provides Redirect to Provisioning Controller

4 Workflow of the Ownership Voucher OV

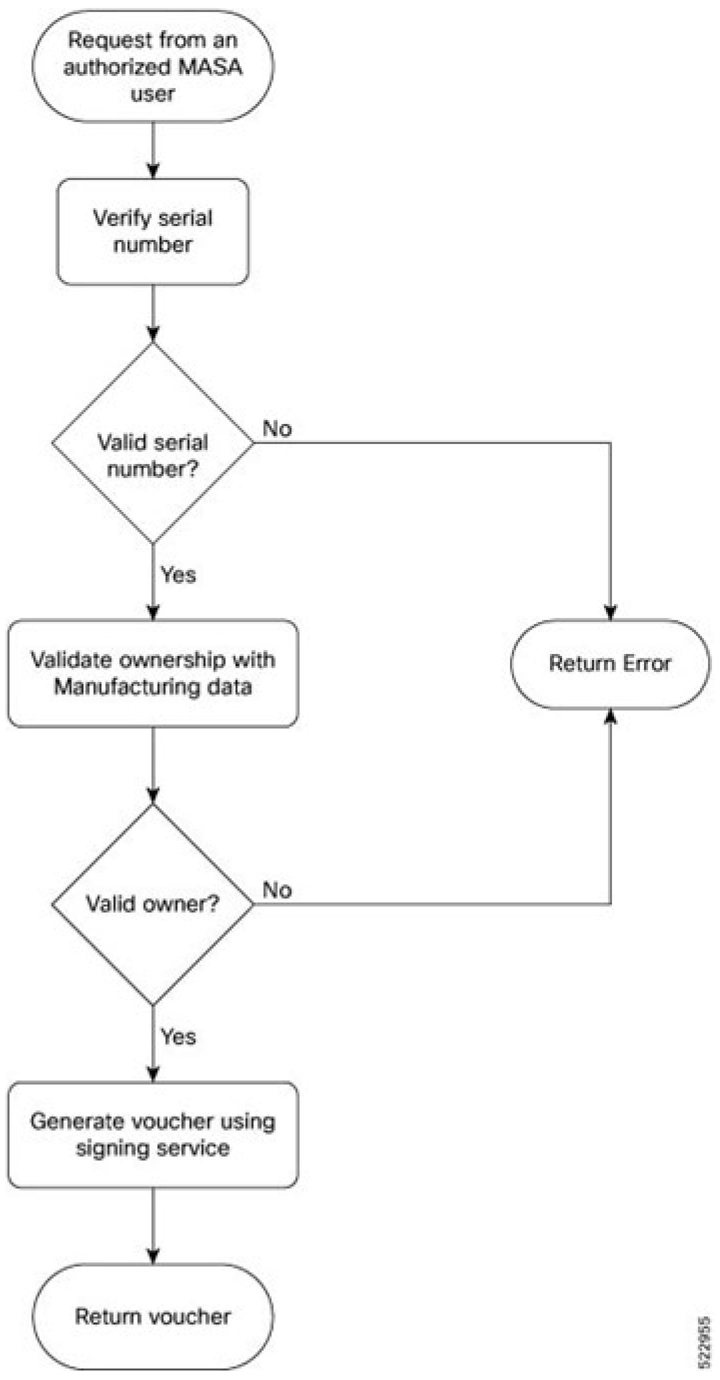
The Ownership Voucher OV is a certificate from Cisco. With this certificate the customer can prove that this device belongs to their organization. The goal is for the device to be able to authenticate the server. When the device boots, it validates the OV and verifies the signature of the received bootstrapping data.

4.1 Key Terms

- **MASA Server** (Manufacturer Authorized Signing Authority): This service is in our case offered by Cisco. With this service the OV is generated with the Owner Certificate and the serial number of the device.
- **OV (Ownership Voucher)**: to securely identify the devices associated owner organization. The OV is generated by Cisco, for generation the PDC and the serial number of the device is required. The OV is used to complete the chain of trust, which is leading to the PDC. This is described in the RFC 8366.
- **OC (Owner Certificate)**: This is an X.509 certificate. This certificate is used for identifying the organization. The OC is signed by an CA certificate authority. The OC contains the owner certificate itself and the PDC pinned-domain-cert, which is specified in the OV.
- **PDC (pinned-domain-certificate)**: The PDC is in the OV and pins a domain certificate.

4.2 Workflow issuing OV Ownership Voucher

The voucher can be created manually or automatically with the MASA API. Below is the workflow of obtaining the Ownership Voucher.

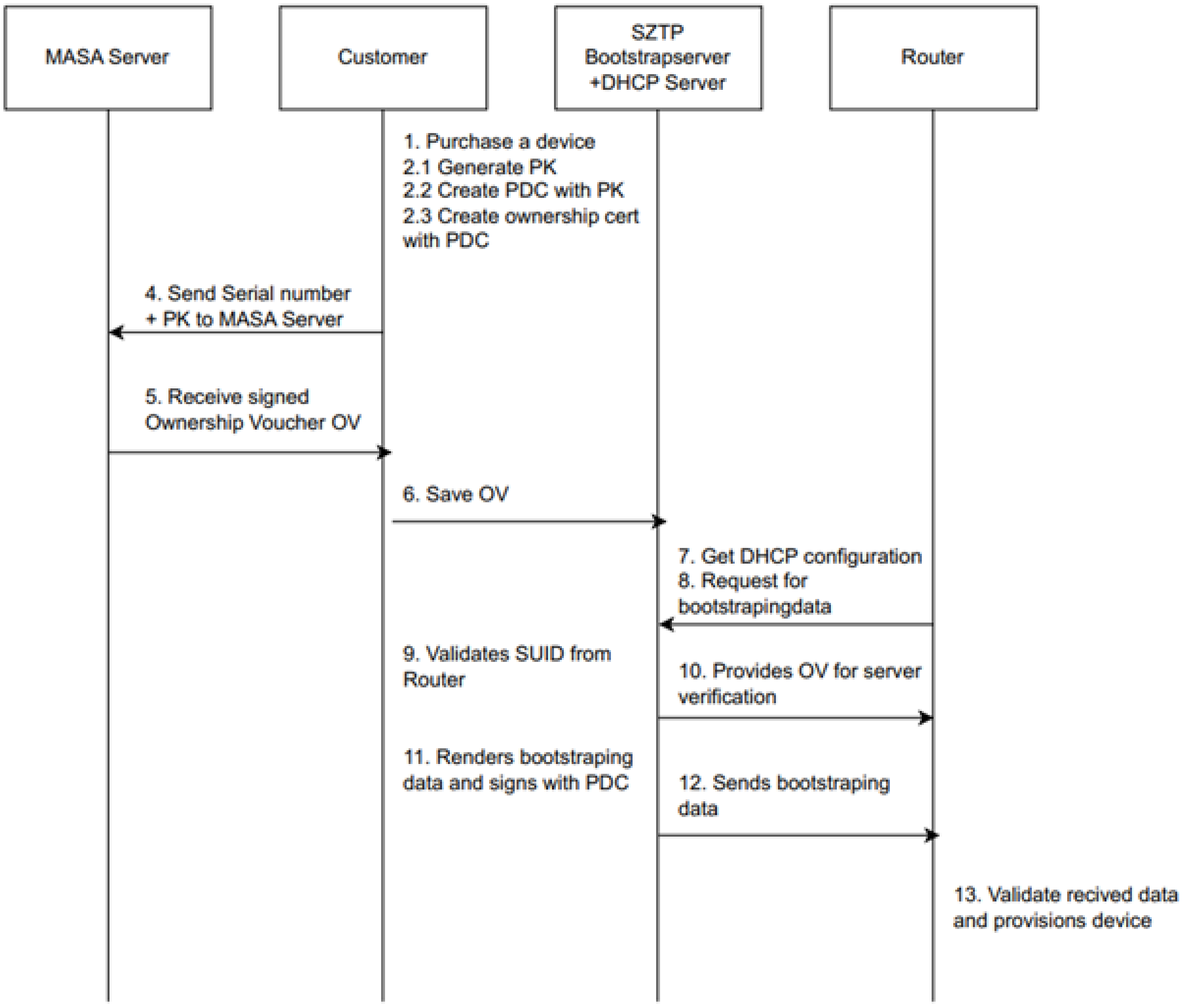

Request from an authorized MASA user
Verify serial number
Valid serial number?
No
Yes
Validate ownership with Manufacturing data
Return Error
Valid owner?
No
Yes
Generate voucher using signing service
Return voucher
522955

Reference

This workflow is the same if the MASA API or the manual way is chosen. In our case, the OVs were created manually. It is out of scope of this project to implement it automatically with the MASA API, because it requires more time than doing it manually. If time allows, we can implement this feature. After the advisors got access to the Web-interface from the MASA server, they uploaded the needed files (PDC and serial numbers) and sent us the two Ownership Vouchers.

4.3 Authentication Workflow

Below is the complete authentication workflow of authenticating a device. This workflow is for a better understanding of the ownership voucher. The diagram is adopted to our environment from the Cisco Ownership Voucher Workflow.


MASA Server
Customer
SZTP Bootstrapserver +DHCP Server
Router
1. Purchase a device
2.1 Generate PK
2.2 Create PDC with PK
2.3 Create ownership cert with PDC
4. Send Serial number + PK to MASA Server
5. Receive signed Ownership Voucher OV
6. Save OV
7. Get DHCP configuration
8. Request for bootstrapingdata
9. Validates SUID from Router
10. Provides OV for server verification
11. Renders bootstraping data and signs with PDC
12. Sends bootstraping data
13. Validate recived data and provisions device

II. User guide

5 DHCP Configuration

SZTP relies on DHCP to redirect network devices to thir bootstrapping server. The option used for the redirect via IPv4 is 143 (v4-sztp-redirect). SZTP uses URI format specified in RFC7227 under the section 5.7, however with a restriction: Even though only one URI can be encapsulated in the option, it's length still has to be included in the bytestring. Since the DHCP server does not seem to support this requirement, the option data is already given in hexadecimal encoding in the configuration below.

5.1 DHCPv4 Kea Configuration

```
{
    "Dhcp4": {
        "interfaces-config": {
            "interfaces": [ "enxc4411efecab1" ]
    },
    "control-socket": {
        "socket-type": "unix",
        "socket-name": "/run/kea/kea4-ctrl-socket"
    },
    "lease-database": {
        "type": "memfile",
        "lfc-interval": 3600
    },
        "valid-lifetime": 600,
        "max-valid-lifetime": 7200,
        "subnet4": [
            {
                "id": 1,
                "subnet": "10.0.0.0/24",
                "pools": [
                    {
                        "pool": "10.0.0.10 - 10.0.0.99"
                    }
                ],
                "option-data": [
                    {
                        "name": "routers",
                        "data": "10.0.0.1"
                    },
                    {
                        "always-send": true,
                        "code": 143,
                        "name": "v4-sztp-redirect",
                        //"data": "<2-byte url length>https://10.0.0.1:8443"
                        "data": "0x001568747470733A2F2F31302E302E302E313A38343433"
                    }
                ]
            }
        ]
    }
}
```

6 Netbox

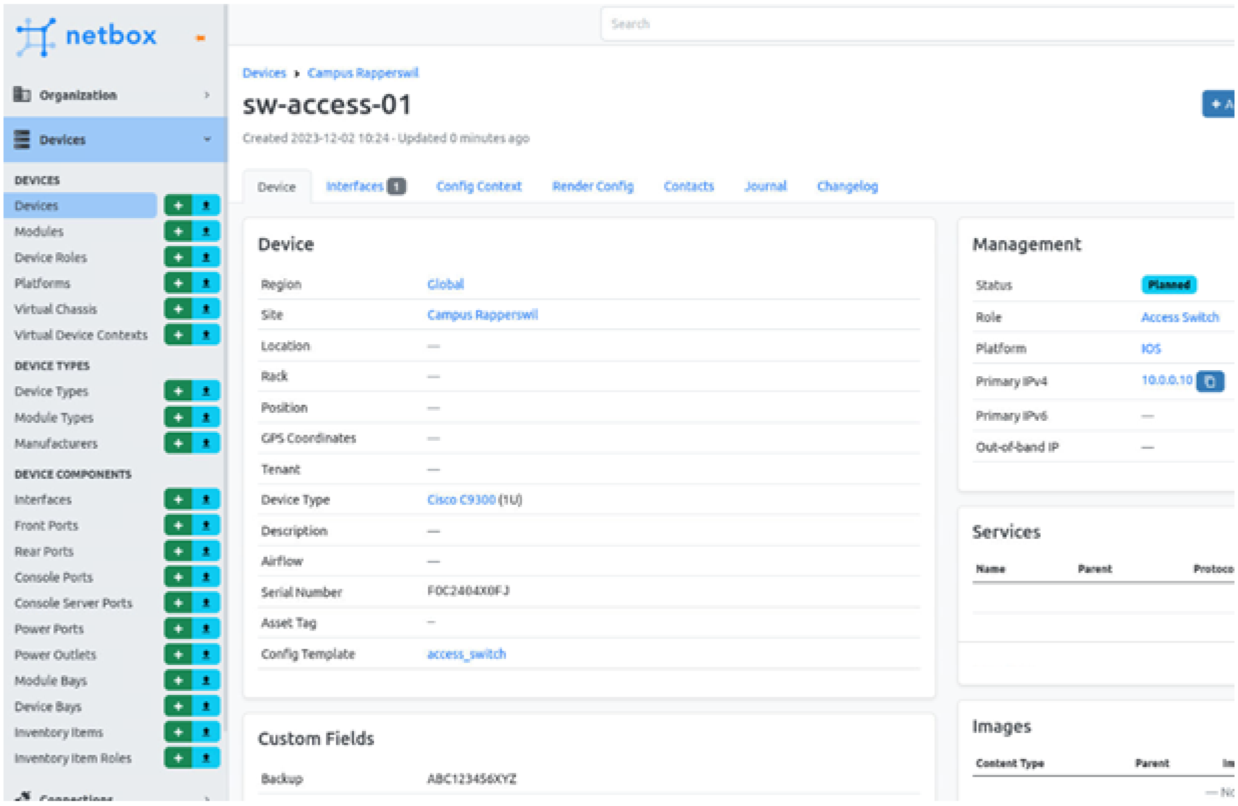
6.1 Setup

In this project, the Github repository is used to install Netbox with Docker. After the docker pull command of the docker Netbox image, the override file has to be customized. The setup is straight-forward: adapting the port setting (here 9000) and start the containers with docker compose up. The repository provides a script to create the admin user and after a few minutes the user can login on the web interface.

6.2 Device Configuration

6.2.1 Create device and related objects

To start with a Region "Global" has to be created, which can later be used to assign the SSH credentials. Also, a Site, in this case "Campus Rapperswil" is created, as well the Manufacturer "Cisco", Device Type "C9300", Device Role "Access Switch" and Platform "IOS". All this is needed to create the device. An important aspect is the Platform of the device. This is later needed for Napalm, to find the correct connection driver.



Devices ▸ Campus Rapperswil

sw-access-01

Created 2023-12-02 10:24 · Updated 0 minutes ago

Device

Region	Global
Site	Campus Rapperswil
Location	—
Rack	—
Position	—
GPS Coordinates	—
Tenant	—
Device Type	Cisco C9300 (1U)
Description	—
Airflow	—
Serial Number	FOC2404X0FJ
Asset Tag	—
Config Template	access_switch

Custom Fields

Backup	ABC123456XYZ

Management

Status	Planned
Role	Access Switch
Platform	IOS
Primary IPv4	10.0.0.10
Primary IPv6	—
Out-of-band IP	—

Services

Name	Parent	Protoco

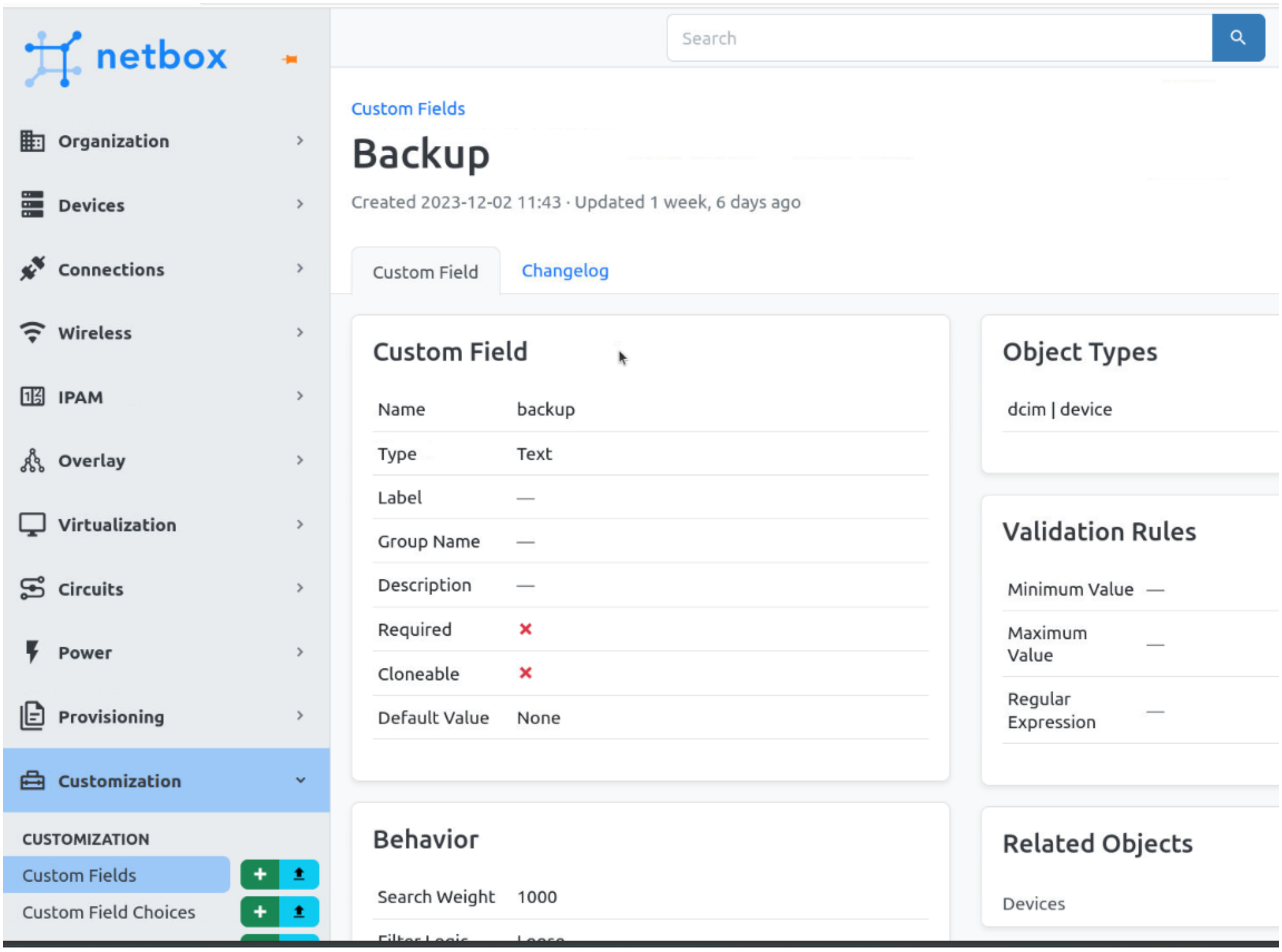
Images

Content Type	Parent	Im

6.2.2 Custom Fields

Netbox itself already provides many pre-defined fields for a device. For a more complex usage, custom fields can be used to store a new key value pair. For the backup option in this project, a custom field is used. If the device should boot with a backup, it must be filled with the serial number of the device, which will be replaced.

At creation of the custom backup field, the content type has to be set to `DCIM>Device`, the name to `backup` and the type to `text`.



Backup

Created 2023-12-02 11:43 · Updated 1 week, 6 days ago

Custom Field

Name	backup
Type	Text
Label	—
Group Name	—
Description	—
Required	✗
Cloneable	✗
Default Value	None

Object Types

dcim | device

Validation Rules

Minimum Value	—
Maximum Value	—
Regular Expression	—

Behavior

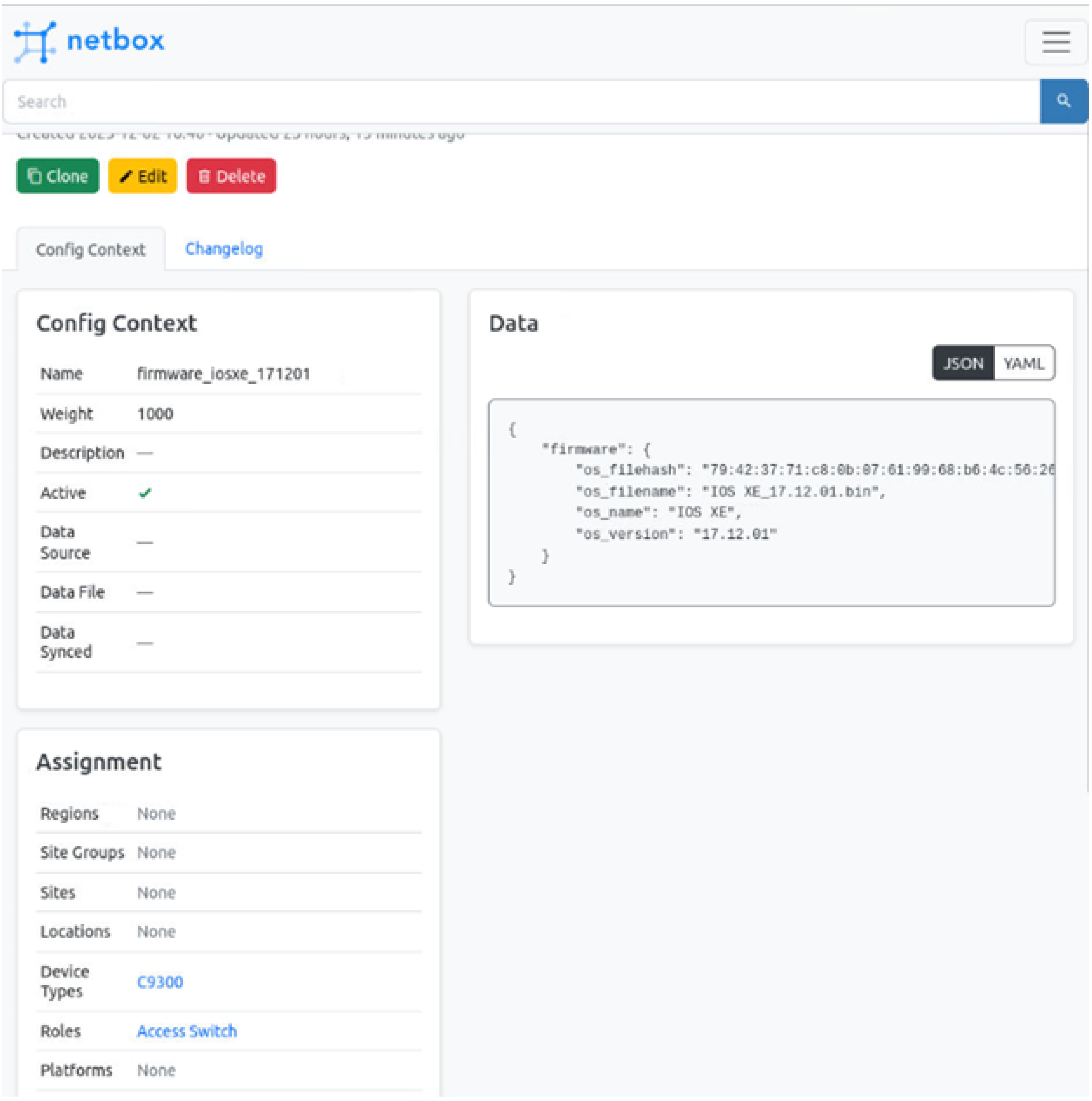
Search Weight	1000

Related Objects

Devices

6.2.3 Config Context

Next, two Config Contexts have to be created to provide configuration values that are shared by multiple devices. One of which is for the SSH settings and the other one for the target firmware. The Config Contexts can be assigned on different levels, depending on how broad they should be applied. The SSH settings have to be assigned to the “Global” Region and an example firmware to the Role “Access Switch” and Device Type “C9300” because firmware images are likely to be device dependent.



Config Context

Name	firmware_iosxe_171201
Weight	1000
Description	—
Active	✓
Data Source	—
Data File	—
Data Synced	—

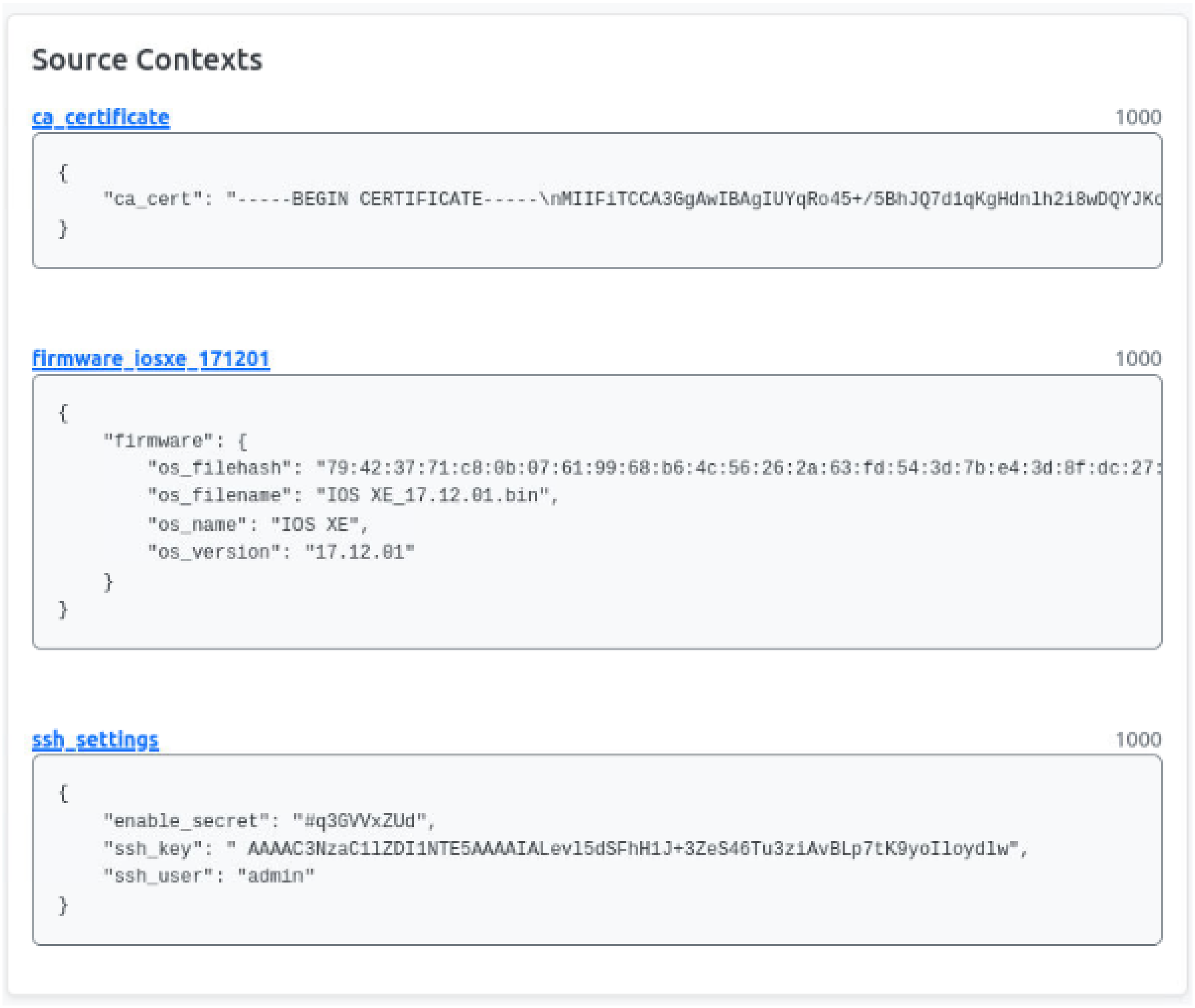
Data

```
{
    "firmware": {
        "os_filehash": "79:42:37:71:c8:0b:07:61:99:68:b6:4c:56:26
        "os_filename": "IOS XE_17.12.01.bin",
        "os_name": "IOS XE",
        "os_version": "17.12.01"
    }
}
```

Assignment

Regions	None
Site Groups	None
Sites	None
Locations	None
Device Types	C9300
Roles	Access Switch
Platforms	None

All applicable configuration contexts are automatically merged at the device level and can be queried via the API.



Source Contexts

ca_certificate 1000

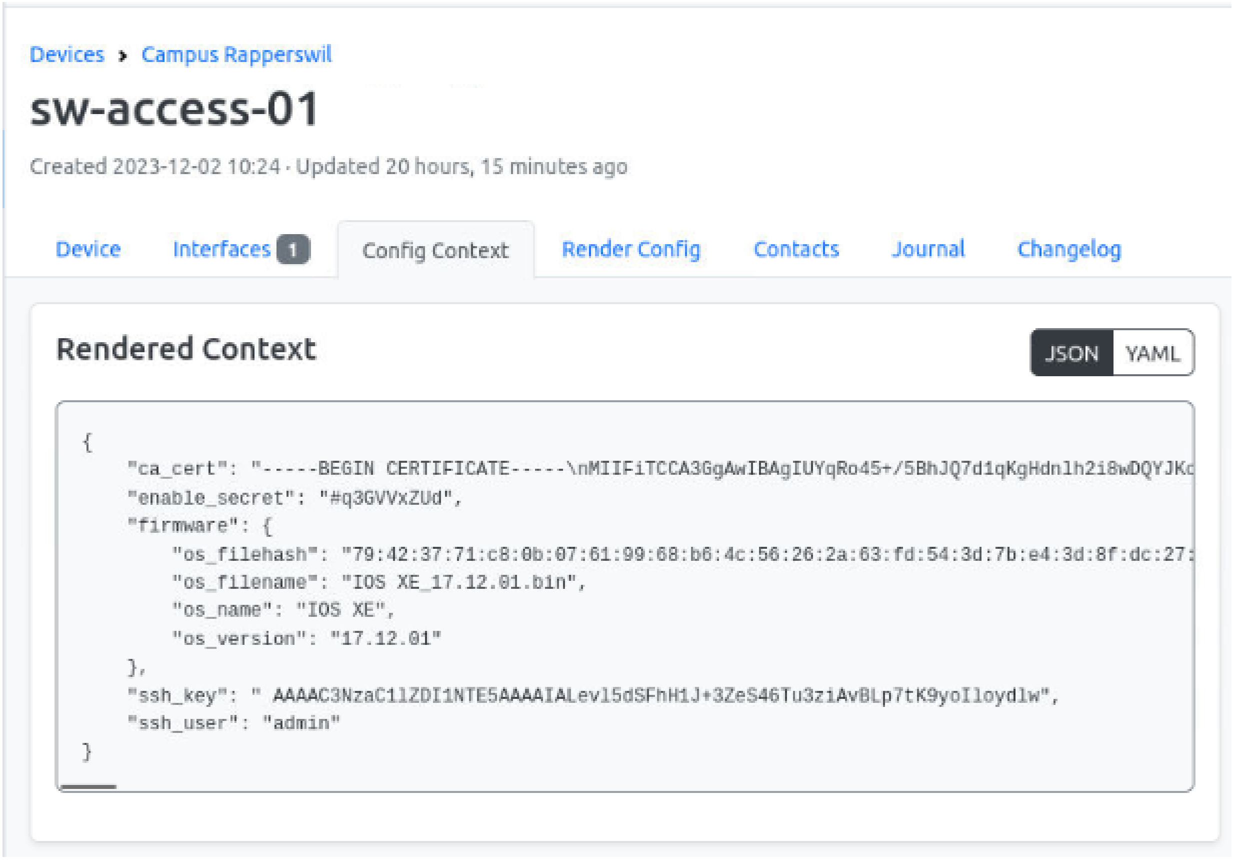
```
{
    "ca_cert": "-----BEGIN CERTIFICATE-----\nMIIFiTCCA3GgAwIBAgIUYqRo45+/5BhJQ7d1qKgHdnlh2i8wDQYJKc
}
```

firmware_iosxe_171201 1000

```
{
    "firmware": {
        "os_filehash": "79:42:37:71:c8:0b:07:61:99:68:b6:4c:56:26:2a:63:fd:54:3d:7b:e4:3d:8f:dc:27:
        "os_filename": "IOS XE_17.12.01.bin",
        "os_name": "IOS XE",
        "os_version": "17.12.01"
    }
}
```

ssh_settings 1000

```
{
    "enable_secret": "#q3GVVxZUd",
    "ssh_key": " AAAAC3NzaC1lZDI1NTE5AAAAIALevl5dSFhH1J+3ZeS46Tu3ziAvBLp7tK9yoIloydlw",
    "ssh_user": "admin"
}
```



Devices › Campus Rapperswil

sw-access-01

Created 2023-12-02 10:24 · Updated 20 hours, 15 minutes ago

Rendered Context

```
{
    "ca_cert": "-----BEGIN CERTIFICATE-----\nMIIFiTCCA3GgAwIBAgIUYqRo45+/5BhJQ7d1qKgHdnlh2i8wDQYJKc
    "enable_secret": "#q3GVVxZUd",
    "firmware": {
        "os_filehash": "79:42:37:71:c8:0b:07:61:99:68:b6:4c:56:26:2a:63:fd:54:3d:7b:e4:3d:8f:dc:27:
        "os_filename": "IOS XE_17.12.01.bin",
        "os_name": "IOS XE",
        "os_version": "17.12.01"
    },
    "ssh_key": " AAAAC3NzaC1lZDI1NTE5AAAAIALevl5dSFhH1J+3ZeS46Tu3ziAvBLp7tK9yoIloydlw",
    "ssh_user": "admin"
}
```

6.2.4 Config Templates

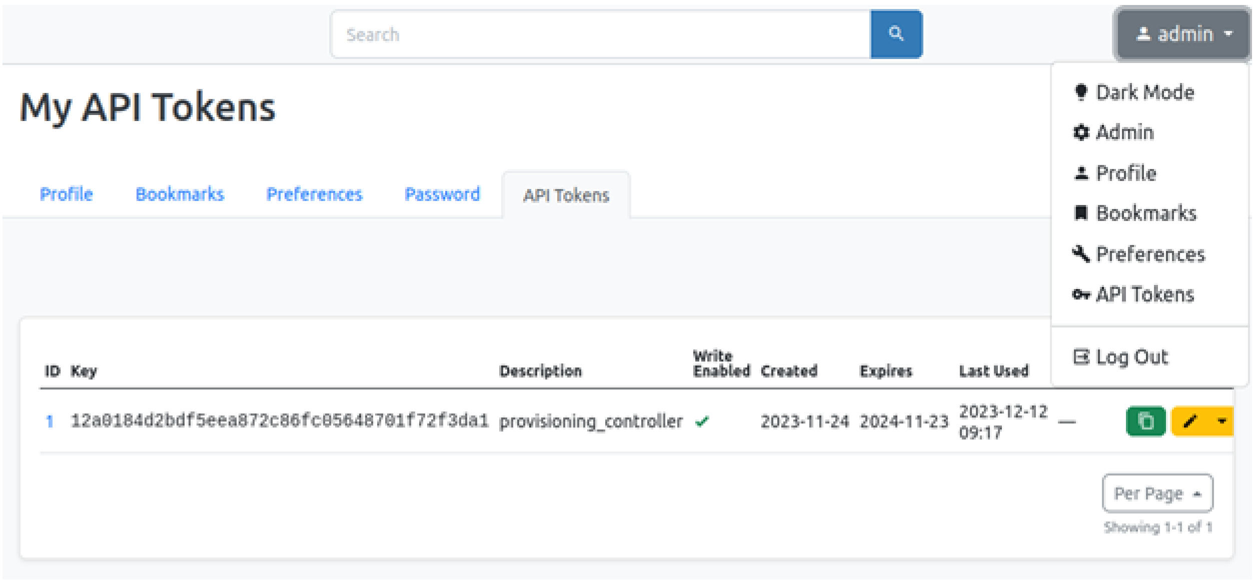
For the configuration, that needs to be written to the device, a Cisco-style configuration is needed. To achieve this, the built-in Config template option in Netbox is used. The templates are automatically rendered with the values provided by either the device object itself, Custom Fields or the Config Context. Below is the Config Template, which is used in this project.

```
configure terminal
    no logging console informational
    hostname {{ device.name }}

!Interfaces
    {% for interface in device.interfaces.filter(ip_addresses__isnull=False) -%}
        interface {{ interface.name }}
        no shutdown
        ip address {{ interface.ip_addresses.first().address.ip }} {{
interface.ip_addresses.first().address.netmask }}
        exit
    {% endfor -%}

!SSH
    ip domain name sztp.local
    crypto key generate ec keysize 384

    ip ssh version 2
    ip ssh pubkey-chain
        username {{ device.get_config_context().ssh_user }}
            key-string
                {{ device.get_config_context().ssh_key }}
                exit
            exit
        exit
    line vty 0 4
    transport input ssh
    login local
    exit
    enable password {{ device.get_config_context().enable_secret}}

!CA-Cert
    crypto pki trustpoint sztp-ca
        enrollment terminal pem
        exit
    crypto pki authenticate sztp-ca
        {{ device.get_config_context().ca_cert }}

        yes
    end
```

In case we want to restore a configuration backup file from another device, we added a custom field “Backup” to the device which would contain the serial number of the old device which the backup should be taken from. Lastly, we need to create an IP address and attach it to the corresponding interface of the device, so that it is reachable over SSH after provisioning.

6.3 API Access

Netbox features a built-in API which will be used by the controller to connect to Netbox. In order to use the API we needed to create a token for the controller. This could be easily done on the "API Tokens" menu in the user



My API Tokens

ID	Key	Description	Write Enabled	Created	Expires	Last Used	
1	12a0184d2bdf5eea872c86fc05648701f72f3da1	provisioning_controller	✓	2023-11-24	2024-11-23	2023-12-12 09:17	—

dropdown.

7 Controller

7.1 Setup

1. Install python and pip on the system.
2. Create a virtual environment for python and activate it.

   ```
   python3 -m venv venv
   source venv/bin/activate
   ```

3. Install the required pip packages. (At the time of writing this application, there was an incompatibility, that requires to install cython and pyyaml with special options.)

   ```
   pip install "cython<3.0.0" wheel
   pip install "pyyaml==5.4.1" --no-build-isolation
   pip install -r requirements.txt
   ```

7.2 Usage

This application contains two controllers: - Provisioning controller: This controller provides the endpoints for network devices to get their bootstrapping data. - Backup controller: This controller creates backups of the devices in the inventory.

7.2.1 Configuration

The provisioning controller can be configured using the `config.json` file.

The backup controller can be configured using the `backup_controller/config.yaml` file (Netbox URL) and the `backup_controller/.env` file (device authentication, Netbox token).

7.2.2 Provisioning Controller

The provisioning controller can be run directly with flasks built-in web server:

```
python3 provisioning_controller.py
```

7.2.2.1 Test

To test the Provisioning Controller (including the inventory), the mocked device test in the tests folder can be used:

```
python3 test_controller_with_mock_device.py
```

7.2.3 Backup controller

The backup controller should be run regularly. It's advised to add a cron job for the following command:

```
python3 backup_controller/backup_controller.py
```

8 About

This project was realised as a term project at the Eastern Switzerland University of Applied Sciences.

8.1 Authors

- Patrick Lenherr
- Vanessa Gyger