

B A C H E L O R A R B E I T

Cisco Digital Network Architecture und deren Programmierschnittstellen

zur Erlangung des akademischen Grades
Bachelor of Science in Engineering (BSc)

Autor: Michael Höflmaier
Matrikelnummer: 1510286006

Erstbetreuer: FH-Prof. Dipl.-Ing. Dr. techn. Klaus Raab

Zweitbetreuer: Ing. Herbert Putz

Tag der Abgabe: 12.06.2018
Name: Michael Höflmaier
Matrikelnummer: 1510286006

Geburtsdatum: 20.11.1994
Adresse: Aiglhofstraße 9, 5020 Salzburg

Eidesstattliche Erklärung

Mit dieser Erklärung versichere ich die erstmalig vorgelegte Bachelorarbeit selbstständig erstellt zu haben.

Es wurden nur die von mir angegebenen Hilfsmittel und Quellen verwendet.

Datum: 12.06.2018
Ort: Salzburg

Unterschrift:



Genderklausel

Aus Gründen der Lesbarkeit wird in dieser Bachelorarbeit darauf verzichtet, geschlechtsspezifische Formulierungen zu verwenden. Soweit personenbezogene Bezeichnungen nur in männlicher Form angeführt sind, beziehen sie sich auf Männer und Frauen in gleicher Weise.

Zusammenfassung

Diese Arbeit stellt einige praktische Anwendungsbeispiele für die Verwendung von Programmierschnittstellen in einer Cisco Digital Network Architecture-Umgebung vor. Um das Konzept von Cisco Digital Network Architecture zu verstehen, wird eine Vielzahl von Grundbegriffen, wie Software Defined Networking und Software Defined Access, erklärt. Daraufhin kann auf Features und Komponenten von Cisco Digital Network Architecture eingegangen werden. Dabei wird vor allem auf die verwendeten Technologien in Control-, Data- und Policy-Plane genauer eingegangen. Die zentrale Kontrollstelle innerhalb eines Cisco Digital Network Architecture-Netzwerkes, das Cisco Digital Network Architecture-Center und dessen Aufgaben werden danach auch noch genauer beleuchtet. Das erste Anwendungsbeispiel in dieser Arbeit zeigt, dass es möglich ist, innerhalb einer Cisco DNA-Umgebung mit Hilfe von zwei verschiedenen Programmierschnittstellen Systemkomponenten konfigurieren zu können. An dieser Stelle wird die Verwendung der Programmierschnittstellen in den Fokus gerückt und beschrieben. Dafür wird das Konzept von Yet Another Next Generation, Werkzeuge von diesem Konzept, eine Baum-Repräsentation, Internet Protocol Service Level Agreement und das Representational State Transfer Configuration Protocol vorgestellt. Das zweite Anwendungsbeispiel in dieser Arbeit zeigt auf, wie die Programmierschnittstellen einer Cisco Identity Services Engine genutzt werden können. Dabei werden speziell verschiedene Methodiken des Einspielens von Media Access Control-Adressen in die Identity Services Engine vorgestellt.

Suchbegriffe

DNA, YANG, RESTCONF, IP SLA, SDN, SDA, APIC-EM, ISE, MacCurator

Abstract

This thesis discusses different examples of the use of programming interfaces within a Cisco Digital Network Architecture-vicinity. To understand the concepts of Cisco Digital Network Architecture a multitude of basic terms like Software Defined Networking and Software Defined Access are explained. After that features and components of Cisco Digital Network Architecture can be examined. Further on technologies used on the Control-, Data- und Policy-Plane are presented and explained. The central controller within a Cisco Digital Network Architecture-network, named Cisco Digital Network Architecture-Center and its functions are highlighted. The first example within this thesis shows the possibility to configure a networking device within a Cisco Digital Network Architecture-network by the use of two different programming interfaces. Here the focus is set on the use of the programming interfaces. As a consequence of this the concept of Yet Another Next Generation, different tools of this concept, a tree-notation, Internet Protocol Service Level Agreement and the Representational State Transfer Configuration Protocol are presented. The second example within this thesis depicts how the programming interfaces of a Cisco Identity Services Engine can be used. Thereby different methodologies of installing Media Access Control-addresses into the Identity Services Engine are featured.

Search words

DNA, YANG, RESTCONF, IP SLA, SDN, SDA, APIC-EM, ISE, MacCurator

Inhaltsverzeichnis

| | |
|---|----|
| 1. Ziel und Zweck der Arbeit..... | 13 |
| 2. Ablauf und Beschreibung des Praktikums..... | 14 |
| 3. Motivation und wissenschaftliche Aufarbeitung..... | 15 |
| 4. Theoretische Grundlagen..... | 16 |
| 4.1. SDN | 16 |
| 4.2. SDA..... | 17 |
| 4.3. Cisco DNA | 18 |
| 4.3.1. Struktur und Komponenten | 19 |
| 4.3.2. Features und Eigenschaften von DNA..... | 20 |
| 4.3.3. Control Plane | 23 |
| 4.3.4. Data Plane | 26 |
| 4.3.5. Policy-Plane..... | 27 |
| 4.3.5.1. Zugangsrichtlinien | 27 |
| 4.3.5.2. Zugangskontrolle..... | 30 |
| 4.3.5.3. Applikationsrichtlinien..... | 30 |
| 4.3.6. DNA-Center | 31 |
| 5. Programmierschnittstellen – Anwendungsbeispiel 1 | 34 |
| 5.1. Problemstellung und Beschreibung..... | 34 |
| 5.2. Beschreibung des Use Case..... | 35 |
| 5.3. APIC-EM und dessen API..... | 38 |
| 5.4. Infrastruktur API | 42 |
| 5.4.1. YANG..... | 42 |
| 5.4.1.1. Überblick | 42 |
| 5.4.1.2. Strukturen..... | 44 |
| 5.4.1.3. Werkzeuge | 49 |
| 5.4.1.4. Baum-Struktur | 51 |
| 5.4.2. RESTCONF | 54 |
| 5.5. API-Abfragen | 54 |
| 5.6. IP SLA..... | 56 |
| 5.7. PHP..... | 57 |
| 5.8. DynamicConfigurationWizard..... | 59 |
| 6. Programmierschnittstellen – Anwendungsbeispiel 2 | 64 |
| 6.1. Problemstellung und Beschreibung..... | 64 |
| 6.2. Lösungsansatz..... | 64 |
| 6.3. Generelle Funktionsweise | 65 |

| | | |
|--------|---|----|
| 6.4. | MacCurator | 65 |
| 6.4.1. | Konsolen-Version..... | 65 |
| 6.4.2. | Webinterface-Version | 67 |
| 6.4.3. | Webinterface-Version mit Zeitbeschränkung | 72 |
| 7. | Fazit und Ausblick | 75 |
| 8. | Literaturverzeichnis | 77 |

Abbildungsverzeichnis

| | |
|---|----|
| Abbildung 4.1: Konzept zielbasierendes Networking [4]..... | 18 |
| Abbildung 4.2: Komponenten eines DNA-Netzwerks [4] | 19 |
| Abbildung 4.3: Over- und Underlay [4] | 21 |
| Abbildung 4.4: Virtuelle Netzwerke [4]..... | 21 |
| Abbildung 4.5: Skalierbare Gruppen [4] | 22 |
| Abbildung 4.6: Generelle Übersicht LISP [6] | 23 |
| Abbildung 4.7: ID zur Standort-Auflösung [6] | 24 |
| Abbildung 4.8: Simples Beispiel zu LISP [6]..... | 24 |
| Abbildung 4.9: Darstellung des Datenpakets bei LISP [6] | 25 |
| Abbildung 4.10: VXLAN-Paket [6] | 26 |
| Abbildung 4.11: Größe der VXLAN-Bestandteile..... | 27 |
| Abbildung 4.12: Ablauf bei der MAB-Methode [4]..... | 28 |
| Abbildung 4.13: 802.1X-Methode [4] | 29 |
| Abbildung 4.14: Central Web Authentication-Methode [4] | 29 |
| Abbildung 4.15: Beispiel für eine Verwendung von Gruppen [4] | 30 |
| Abbildung 4.16: Planübersicht über DNA-Komponenten [6]..... | 31 |
| Abbildung 4.17: Austausch von Nachrichten zwischen DNA-Center und NDP [6] | 32 |
| Abbildung 5.1: Überblick über die verfügbaren Programmierschnittstellen [10] ... | 35 |
| Abbildung 5.2: Überblick des gesamten Use Cases [10]..... | 37 |
| Abbildung 5.3: Abbildung der finalen Lösung [10] | 38 |
| Abbildung 5.4: Struktur der Tokenabfrage..... | 39 |
| Abbildung 5.5: Abfrage der physischen Topologie | 40 |
| Abbildung 5.6: cURL-Abfrage Token..... | 40 |
| Abbildung 5.7: cURL-Abfrage der physischen Topologie | 41 |
| Abbildung 5.8: Topologie des Testsystems in APIC-EM | 42 |
| Abbildung 5.9: Module-Header von Cisco-IOS-XE-sla.yang | 44 |
| Abbildung 5.10: Beispiel ‚typedef‘ aus Cisco-IOS-XE-types.yang | 45 |
| Abbildung 5.11: Beispiel ‚container‘ aus Cisco-IOS-XE-sla.yang | 45 |
| Abbildung 5.12: Beispiel ‚leaf‘ aus Cisco-IOS-XE-sla.yang | 46 |
| Abbildung 5.13: Beispiel ‚choice‘ aus Cisco-IOS-XE-sla.yang..... | 47 |
| Abbildung 5.14: Beispiel ‚grouping‘ aus Cisco-IOS-XE-sla.yang..... | 48 |
| Abbildung 5.15: Verwendung von ‚grouping‘ in Cisco-IOS-XE-sla.yang..... | 48 |
| Abbildung 5.16: HTML-Darstellung einer YANG-Datei [13]..... | 50 |
| Abbildung 5.17: Beispielhafte Verwendung des pyang-Tools | 50 |
| Abbildung 5.18: Auszug aus der Baumdarstellung für Cisco-IOS-XE-sla.yang | 51 |
| Abbildung 5.19: Generelle Struktur eines Knotens..... | 52 |
| Abbildung 5.20: Konstruktionsschema der URI | 55 |
| Abbildung 5.21: Auslesen von Daten mittels API | 55 |
| Abbildung 5.22: Ergebnis der Testabfrage | 55 |
| Abbildung 5.23: Eingabe von Daten mittels API | 56 |
| Abbildung 5.24: Beispielanwendung von IP SLA [18]..... | 57 |
| Abbildung 5.25: Auflistung aller Netzwerkkomponenten | 58 |
| Abbildung 5.26: Auflistung aller konfigurierten IP SLA-Operationen | 58 |
| Abbildung 5.27: Konfiguration einer IP SLA-Operation | 58 |
| Abbildung 5.28: Verfügbare Befehle im erstellten Prototyp | 59 |
| Abbildung 5.29: Eingabemaske für Benutzerdaten | 61 |
| Abbildung 5.30: Strukturierte Darstellung eines Moduls innerhalb der Weboberfläche | 62 |

| | |
|---|----|
| Abbildung 5.31: Beispiel von generiertem JSON..... | 63 |
| Abbildung 6.1: Struktur der Daten in einer CSV-Datei..... | 65 |
| Abbildung 6.2: Beispielhafter Aufruf des Programms | 66 |
| Abbildung 6.3: Auszug aus der PHP-Version | 67 |
| Abbildung 6.4: Eintrag einer Ziel-ISE | 68 |
| Abbildung 6.5: Eingabe per CSV-Datei | 69 |
| Abbildung 6.6: Statusanzeige der Datenübertragung | 70 |
| Abbildung 6.7: Erfolgreiche Operation an der ISE | 71 |
| Abbildung 6.8: Eingabe eines Eintrags..... | 72 |
| Abbildung 6.9: Hinzufügen von ISE-Einträgen | 73 |

Abkürzungsverzeichnis

| | |
|----------------|--|
| ACL | Access Control List |
| API | Application Programming Interface |
| APIC-EM | Application Policy Infrastructure Controller Enterprise Module |
| CLI | Command Line Interface |
| CRUD | Create Read Update Delete |
| CSV | Comma-separated Values |
| CWA | Central Web Authentication |
| DNA | Digital Network Architecture |
| DNA-C | Digital Network Architecture-Center |
| DNS | Domain Name System |
| DSDL | Document Schema Definition Languages |
| EAP | Extensible Authentication Protocol |
| EID | Endpoint Identifier |
| ETR | Egress Tunnel Router |
| FH | Fachhochschule |
| GRE | Generic Routing Encapsulation |
| HTTP | Hypertext Transfer Protocol |
| IETF | Internet Engineering Task Force |

| | |
|----------------|---|
| IP | Internet Protocol |
| IP SLA | Internet protocol service level agreement |
| ISE | Identity Services Engine |
| IT | Information Technology |
| JSON | Javascript Object Notation |
| KMP | Kapsch Management Plattform |
| LISP | Locator/Identifier Separation Protocol |
| MAB | MAC Authentication Bypass |
| MAC | Media Access Control |
| MIB | Management Information Base |
| MTU | Maximum Transmission Unit |
| NDP | Network Data Platform |
| NETCONF | Network Configuration Protocol |
| PHP | PHP: Hypertext Preprocessor |
| QoS | Quality of Service |
| REST | Representational State Transfer |
| RLOC | Routing Locator |
| SDA | Software Defined Access |
| SDN | Software Defined Networking |

| | |
|--------------|--|
| SGACL | Security Group Access Control List |
| SGT | Security Group Tag |
| SLA | Service-level Agreement |
| SNMP | Simple Network Management Protocol |
| SQL | Structured Query Language |
| SSH | Secure Shell |
| SSID | Service Set Identifier |
| URI | Uniform Resource Identifier |
| URL | Uniform Resource Locator |
| VN | Virtual Network |
| VRF | Virtual Route Forwarding |
| VXLAN | Virtual Extensible LAN |
| WLAN | Wireless LAN |
| XSLT | Extensible Style Language Transformation |
| YANG | Yet Another Next Generation |
| YDK | YANG Development Kit |

1. Ziel und Zweck der Arbeit

Die Anforderung an diese Bachelorarbeit ist, dem Leser die behandelten Themen möglichst kurz und prägnant zu vermitteln. Durch textuelle und visuelle Informationen soll ein grundlegendes Verständnis für die Materie entwickelt werden und dadurch eine Basis entstehen, mit deren Hilfe anschließend tiefer in die Thematik eingestiegen werden kann. Die Bachelorarbeit behandelt hauptsächlich neue Entwicklungen und Produkte der Netzwerktechnik, wie Cisco DNA und deren Programmierschnittstellen. Diese Schnittstellen ermöglichen es das Paradigma Software Defined Networking auf eine neue Entwicklungsstufe zu heben. Um diese Themen besser verständlich zu machen und die vorhandenen Möglichkeiten von SDN aufzuzeigen, werden verschiedene Anwendungsbeispiele vorgestellt. Überdies wird über den Verbreitungsstatus dieser Technologien in der Wirtschaft und deren voraussichtliche zukünftige Entwicklung informiert.

2. Ablauf und Beschreibung des Praktikums

Das Praktikum des Autors der vorliegenden Arbeit wurde in einem der größten Unternehmen der österreichischen Wirtschaft absolviert. Im Januar 2018 erhielt der Autor eine Zusage der Kapsch BusinessCom, dass ab Februar 2018 die Absolvierung eines Praktikums in der Zweigstelle in Salzburg möglich sei. Das Ausbildungspraktikum hatte eine Dauer von vier Monaten und endete somit Ende Mai 2018. Vorab wurde ein Aufgabenbereich grob umrissen, in dem der Ausbildungspraktikant tätig sein und in Zusammenarbeit mit Experten eigenständig an Projekten mitarbeiten sollte. Im Laufe des Praktikums wurden durch den Projektbetreuer mehrere Projekte injiziert und durch den Autor bearbeitet. Einige dieser Projekte werden im Verlauf dieser Bachelorarbeit genauer ausgeführt. Das erste Projekt mit dem Namen ‚Cisco DNA Enablement‘ wurde vor allem im ersten Monat der Praktikumstätigkeit durchgeführt. Danach wurde ein neues, auf diesem Projekt aufbauendes Projekt mit dem Namen ‚DynamicConfigurationWizard‘ bearbeitet. Ende Februar kam ein weiteres Projekt hinzu, bei dem ein Tool mit dem Namen ‚MacCurator‘ erstellt werden sollte. Hierzu wurden bis Mitte April mehrere Versionen fertiggestellt. Im Monat Mai wurden bestehende offene Projekte abgeschlossen und ein letztes API-Projekt zu Firewalls des Herstellers Checkpoint fertiggestellt.

3. Motivation und wissenschaftliche Aufarbeitung

Die Motivation, diese Arbeit zu verfassen, steht im Zusammenhang mit den darin behandelten Themen und Produkten. Neuartige Technologien, die bisher nur einer geringen Anzahl von Personen in Österreich bekannt sind und daher auch an vielen Bildungseinrichtungen noch nicht gelehrt werden, sind dabei der Hauptmotivator. Auch das Kennenlernen dieser Technologien im Rahmen der Erstellung dieser Arbeit war überdies ein großer Motivationsfaktor. Die praktische Anwendung durch Beispiele ermöglichte es, ein tieferes Verständnis der Technologien zu erreichen. All diese Faktoren und die Tatsache, dass diese Technologien in neuen Produkten verwendet werden, verleihen dem Thema eine große Anziehungskraft.

Die wissenschaftliche Aufarbeitung betreffend lässt sich sagen, dass die meisten in dieser Arbeit verwendeten Technologien und Produkte ihren Ursprung in der nahen Vergangenheit haben und dadurch die wissenschaftliche Aufarbeitung noch nicht in einem großen Ausmaß gegeben ist. Außerdem ist hierbei anzumerken, dass die beschriebenen Produkte von einem einzelnen Hersteller stammen und dass diese Produkte in vielen Unternehmen noch nicht zur Standardausrüstung gehören. Daher wurden – trotz extensiver Recherchearbeit zu den Technologien und Produkten, die in dieser Arbeit behandelt werden und dadurch Relevanz besitzen – keine wissenschaftlichen Abhandlungen gefunden. Ein Beispiel für die Gründe dieser geringen Thematisierung wäre, dass laut Aussagen von Stefan Honeder, dem Produktverantwortlichen von Cisco Österreich, die ersten technischen Beschreibungen zu dem Cisco-Produkt ‚DNA-Center‘ erst im November 2017 erschienen sind. Da sich viele Funktionalitäten des Produktes noch in Entwicklung befinden und noch nicht veröffentlicht wurden, werden diese Dokumente erst laufend aktualisiert.

4. Theoretische Grundlagen

In diesem Abschnitt wird auf verschiedene Themengebiete eingegangen, die in Zusammenhang mit den im weiteren Verlauf vorgestellten Anwendungsbeispielen stehen. Das Themenfeld unterteilt sich in SDN (siehe Abschnitt 4.1), SDA (siehe Abschnitt 4.2) und Cisco DNA (siehe Abschnitt 4.3).

4.1. SDN

Um in den weiterführenden Abschnitten die Begrifflichkeiten und den Kontext zu verstehen, ist es nötig, zuerst den Begriff SDN zu erklären.

SDN steht im Englischen für ‚Software Defined Networking‘. Dieser Ausdruck bedeutet in deutscher Sprache etwa so viel wie ‚Durch Software definierte Netzwerke‘ und meint damit, dass die Konfiguration eines Netzwerkes durch Software verändert werden kann. Meist wird bei SDN auch der Begriff eines Controllers erwähnt, der Verwaltungsaufgaben übernimmt. Dies ermöglicht Netzwerkadministratoren Netzwerkhardware nicht mehr manuell konfigurieren zu müssen, sondern dies automatisch durch Software erledigen zu lassen. SDN entkoppelt aus diesem Grund vor allem den Control Plane und den Data Plane voneinander. Der Control Plane ist die Ebene, die entscheidet, wohin der Datenverkehr geschickt werden soll und ist daher auch für die Weiterleitungsentscheidung von Daten zuständig. Zum Beispiel werden hier diese Entscheidungen durch das Routing-Protokoll getroffen. Der Data Plane setzt dann Entscheidungen, die auf dem Control Plane getroffen wurden, in eine Aktion um. Diese Aktion ist dann zum Beispiel das Senden von Daten über eine Verbindung, die durch das Routing-Protokoll definiert wurde. Dieses Beispiel ist jedoch nur ein Teilabbild der Möglichkeiten und der möglichen Definitionen, die der Begriff SDN zulässt. Dagegen lässt sich unter diesem Begriff auch jegliche Aktion einordnen, die es ermöglicht, das Verhalten eines Netzwerkes zu beeinflussen. Eine erste Definition und ein erster Konzeptentwurf kamen im Jahr 2005 durch die Stanford University zustande. Danach dominierte lange Zeit das OpenFlow-Protokoll. Seit einigen Jahren verfolgen jedoch immer mehr Hersteller von Netzwerkhardware den Ansatz, ihre eigenen SDN-Lösungen in den Markt einzuführen. Allen voran steht Cisco, die sehr viele Ressourcen in dieses Thema investieren. Dessen ungeachtet ziehen auch andere namhafte Hersteller, wie zum Beispiel Barracuda,

in den Kampf um Marktanteile in diesem Segment mit eigenen Produkten nach. In dieser Arbeit wird dagegen ausschließlich auf Lösungen des Herstellers Cisco eingegangen [1].

4.2. SDA

Die Abkürzung SDA steht im Englischen für ‚Software Defined Access‘ und ist ein Begriff, der sehr stark von Cisco benutzt wird. SDA soll, auf der Basis von SDN, den Zugriff auf Netzwerke regeln und stellt einige weitere Fähigkeiten zur Verfügung. In dieser Gruppe von Fähigkeiten findet sich eine identitätsbasierende Trennung zwischen einzelnen Gruppen, welche jedoch weiterhin dieselbe Infrastruktur benutzen. Diese Fähigkeit wird durch die Verwendung von Cisco TrustSec, LISP, VXLAN, VRF und Cisco ISE ermöglicht. Ein weiterer Vorteil, den SDA von Cisco bietet, ist, dass Automatisierung im Netzwerk sowohl durch einen zentralen Controller, Cisco DNA-Center genannt, als auch durch zur Verfügung gestellte Programmierschnittstellen erreicht werden kann. Dabei stützt sich dies auf von Cisco hergestellte Hardware- und Softwarekomponenten. Diese Komponenten sind entweder bereits für den Markt vorhanden und werden aktiv in Betrieben genutzt oder haben gerade erst die Marktreife erlangt. Bereits vorhanden sind der Cisco APIC-EM, die Cisco ISE und einige andere Hardwarekomponenten, wie Router, Switches, Accesspoints und WLAN-Controller. Das seit Herbst 2017 verfügbare DNA-Center des Herstellers Cisco bildet dabei einen zentralen Aspekt, der genauer beleuchtet werden muss und nicht getrennt vom Begriff SDA gesehen und behandelt werden kann [2].

Um eine SDA-Lösung des Herstellers Cisco in ein Netzwerk implementieren zu können, müssen einige Voraussetzungen erfüllt werden. Eine Grundvoraussetzung ist, dass die MTU innerhalb des Netzwerks auf 1550 Byte erhöht werden muss, weil aufgrund von VXLAN 50 Byte zu den ursprünglichen standardmäßigen 1500 Byte des Ethernet Frames hinzugefügt werden. Daher muss auch auf der zugrundeliegenden Hardware die Unterstützung von Jumboframes möglich sein. Dies muss geschehen, weil sonst eine Fragmentierung stattfinden würde. Eine zweite Voraussetzung für die optimale Verwendung von SDA in einem Netz ist die Verwendung des ‚Routed Access-Konzepts‘. In einer traditionellen Campus-Struktur versteht man darunter, dass die Switches des Access-Bereichs auf Layer 3 arbeiten und sich nicht nur auf Layer 2

stützen. Die wichtigste Voraussetzung ist der Besitz von SDA-kompatiblen Hardwarekomponenten [3].

4.3. Cisco DNA

Bei Cisco DNA handelt es sich sowohl um eine technologische Idee als auch um eine Software. DNA versucht durch seinen Namen und andere verwendete Mode- und Marketingwörter zu implizieren, dass im Zeitalter der Digitalisierung dieses Konzept, die Lösung für Anforderungen der Netzwerktechnik darstellt. Innerhalb dieses Konzepts findet sich auch der Begriff des DNA-Centers, welches die Hardwarekomponente darstellt, die das Konzept implementiert und ermöglicht [4].

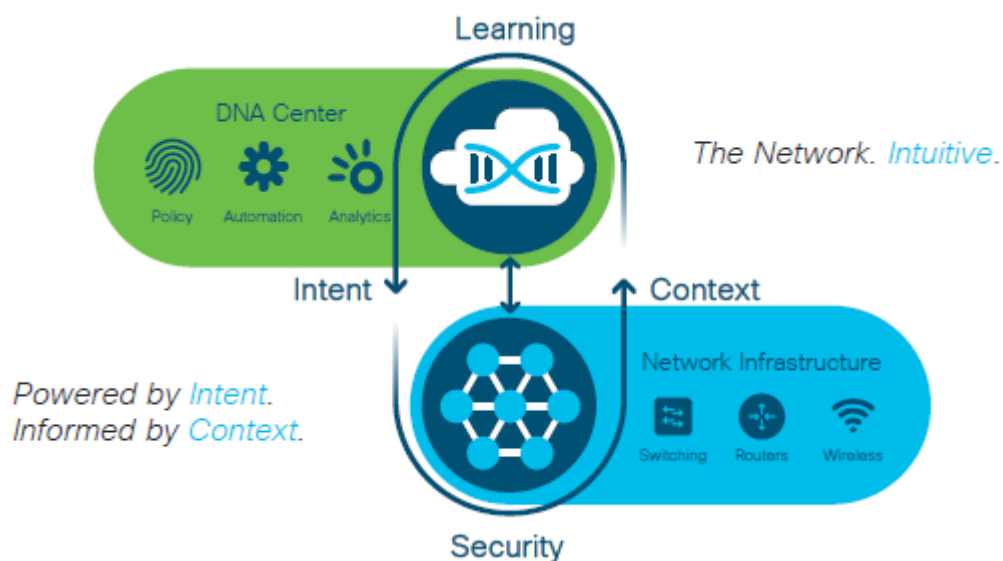


Abbildung 4.1: Konzept zielbasierendes Networking [4]

Wie in Abbildung 4.1 zu sehen, nimmt Cisco dafür den Ansatz, ein Ziel über eine stetige Verbesserung und Lernen zu verfolgen. Dabei gibt das DNA-C der Netzwerkinfrastruktur, welche aus Switches, Routern und Wireless-Komponenten besteht, ein Ziel vor und lässt aus den kontextuellen Daten, die diese Netzwerkgeräte an das DNA-C zurückliefern, ein Lagebild schmieden. Dieses wird wiederum analysiert, um neue Zielvorgaben zu liefern. Wie bereits erwähnt, befinden sich innerhalb des Konzeptbegriffs DNA verschiedene Netzwerkkomponenten [4].

4.3.1. Struktur und Komponenten

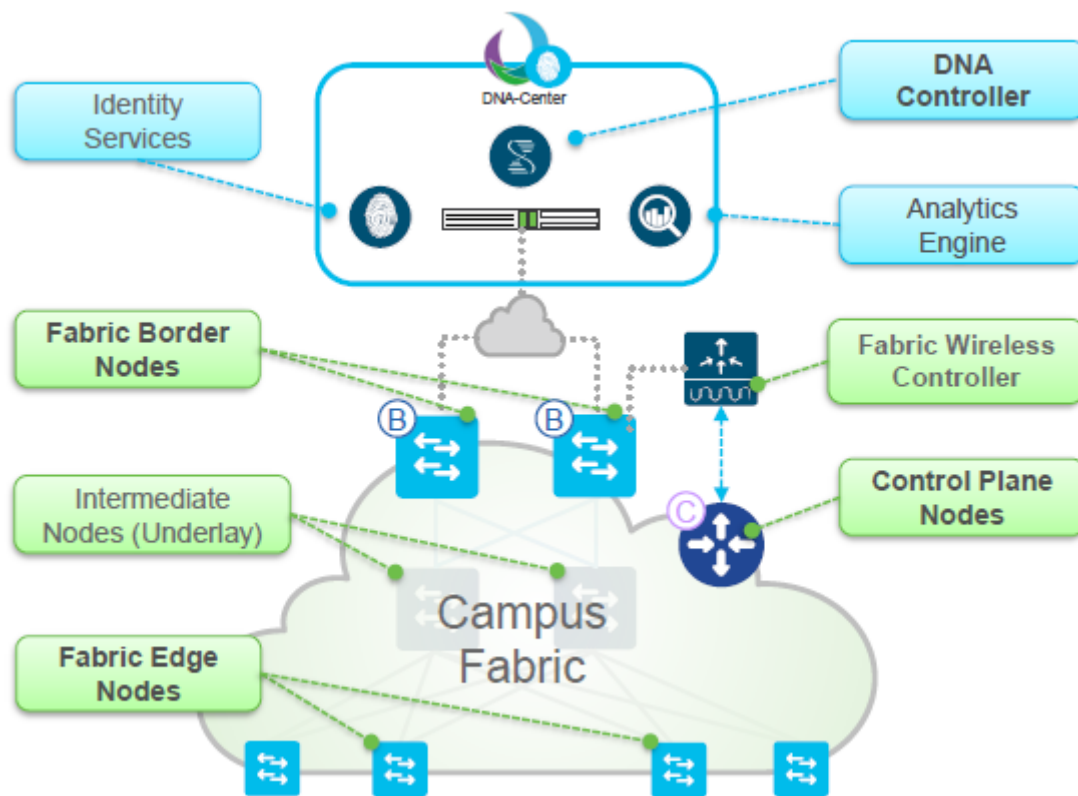


Abbildung 4.2: Komponenten eines DNA-Netzwerks [4]

In Abbildung 4.2 wird ein typisches Campusnetz innerhalb von Cisco DNA dargestellt. Das verwendete Beispielnetz enthält alle wichtigen Bestandteile, um alle Anforderungen von Cisco DNA zu erfüllen. Zentral findet sich das DNA-Center, die ISE und die Analytics Engine. Der Fabric Wireless Controller ist für Steuerungs-, Weiterleitungs- und Kontrollaufgaben der assoziierten WLAN-Accesspoints zuständig. Die Aufgaben des Control Plane Nodes werden vor allem durch die Verwendung von LISP definiert und vorgegeben. Der Control Plane Node besteht aus einer Host Tracking-Datenbank, welche die Bindung zwischen der LISP Endpoint ID und einem Fabric Edge Node beinhaltet, einem Map-Server, der durch Registrierungsnachrichten der Fabric Edge-Geräte die Host Tracking Datenbank befüllt und einem MapResolver, welcher wiederum die Aufgabe hat, auf Map-Anfragen der Fabric Edge Nodes zu reagieren und diese dem RLOC zurückzuschicken. Die Fabric Border Nodes haben allem voran die Aufgabe, das Ende der Fabric mit dem Netzwerk außerhalb der Fabric zu verbinden. Hierbei müssen Endpoint IDs weitergegeben und VXLAN Header in Cisco Meta Data umgewandelt werden. Weitere Aufgaben beziehen sich auf LISP und umfassen

unter anderem die Implementierung eines LISP Proxy Tunnel Routers. Die letzte aus Abbildung 4.2 entnommene Hardware, die in weiterer Folge genauer erklärt wird, ist der Fabric Edge Node, weil Intermediate Nodes für die Fabric nur als Transportmittel dienen und daher nicht in der Fabric integriert sind. Für an Fabric Edge Nodes detektierte Hosts werden LISP Map-Meldungen an den Control Plane Node gesendet, um diese in die Host Tracking Datenbank einzutragen. Im nächsten Schritt werden Hosts zu einem VN hinzugefügt. Hinzu kommt, dass jeder Fabric Edge Node dasselbe Gateway verwendet, das aus einer virtuellen IP und einer MAC-Adresse besteht, um die Mobilität eines Hosts über mehrere Routing Locations zu gewährleisten. Dies führt dazu, dass diese Hosts keine Änderung ihres Standardgateways vornehmen müssen, falls sie ihren Ort ändern [5].

4.3.2. Features und Eigenschaften von DNA

Der bereits mehrfach verwendete englische Begriff ‚Fabric‘ wird im folgenden Absatz genauer erklärt. Laut einer Definition von Cisco ist eine ‚Fabric‘ eine virtuelle Überlagerung, die ein logisches Netzwerk enthält. Diese Überlagerung baut auf der physisch existierenden Schicht auf. Die virtuelle Überlagerung wird daher auch ‚Overlay‘ genannt, während die physische Schicht den Namen ‚Underlay‘ erhält. Ein Beispiel dafür wäre ein GRE-Tunnel, der dem Konzept entsprechend, auf der physischen Schicht aufbaut und diese über eine virtuelle Verbindung abstrahiert. Daher kann behauptet werden, dass ein Overlay auch immer andere Weiterleitungsattribute als das Underlay besitzt. Eine beispielhafte Darstellung für dieses Konzept findet sich in Abbildung 4.3 [4].

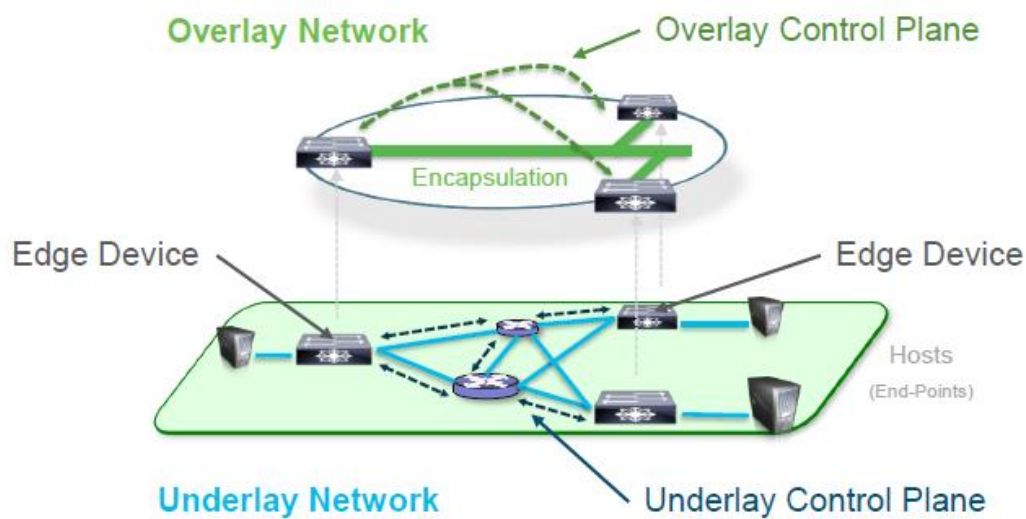


Abbildung 4.3: Over- und Underlay [4]

Solche Overlays finden in Cisco DNA dahingehend Verwendung, dass virtuelle Netzwerke und skalierbare Gruppen erstellt werden können und diese ein eigenes Overlay bereitstellen. So kann ein erstelltes VN, eine Segmentierung des physischen Netzes erreichen [4].

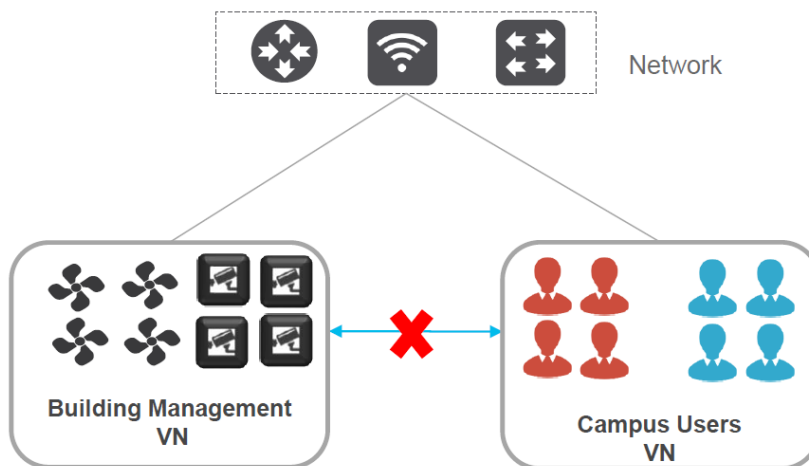


Abbildung 4.4: Virtuelle Netzwerke [4]

In Abbildung 4.4 ist zu sehen, wie eine solche Segmentierung eines physischen Netzes aussehen kann. Zum Beispiel werden hier auf der linken Seite Überwachungsinstallationen und Lüftungssteuerungen zu einem VN mit dem Namen ‚Gebäudemanagement‘ zusammengefasst. Auf der anderen Seite finden sich alle Benutzer des Campus-Netzwerkes, die in einem eigenen VN zusammengefasst werden. Die beiden dadurch entstandenen VN existieren im

gleichen Campus-Netzwerk auf denselben Hardwarekomponenten, können jedoch nicht aufeinander zugreifen, weil sie voneinander logisch getrennt sind. Diese Einteilung lässt sich jedoch auch noch feiner unterteilen [4].

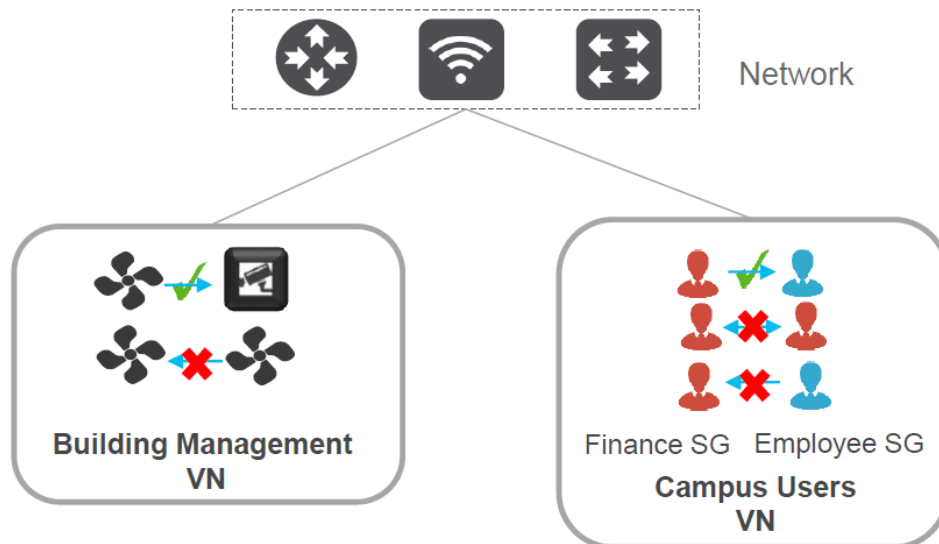


Abbildung 4.5: Skalierbare Gruppen [4]

In Abbildung 4.5 findet die Unterteilung von allen Entitäten eines VN in einzelne Gruppen statt. Diese Gruppen können dann für eine rollen- beziehungsweise gruppenbasierte Zugangskontrolle verwendet werden. Diese skalierbaren Gruppen werden mittels eines sogenannten ‚Scalable Group Tag‘ identifiziert. Das bedeutet zum Beispiel, dass am Ende dieser zweiten Segmentierung einzelne Gruppen nicht auf Teile einer anderen Gruppe im selben VN zugreifen können. Dies ermöglicht es, einfache Zugriffsregeln, welche sonst oftmals über ACL definiert werden, schneller und einfacher zu erstellen. So kann etwa festgelegt werden, dass ein mobiler Gast, welcher der Gastgruppe zugeordnet ist, auf das Internet beziehungsweise deren Gruppe zugreifen darf, aber keinen Zugriff auf eine Gruppe erhält, die kritische Serverinfrastruktur enthält. Für diesen Vorgang müssen alle Gruppen Teil desselben VN sein [4].

Durch die verwendeten Techniken entsteht innerhalb der Fabric ein kombiniertes Overlay, das Layer 2- und Layer 3-Funktionalität miteinander verbindet. Zum Beispiel bietet Layer 2 die Möglichkeit eine physische Topologie zu emulieren, während Layer 3 die Möglichkeit bietet, Verbindungen und Richtlinien zu abstrahieren. In den folgenden Abschnitten werden Technologien vorgestellt, die diese vertikale Segmentierung der Ebenen ermöglichen [6].

4.3.3. Control Plane

Der Control Plane arbeitet in der SDA-Fabric mittels LISP. LISP sollte als eigene Abkürzung nicht mit der Programmiersprache Lisp verwechselt werden [6].

Separate Identity from Location

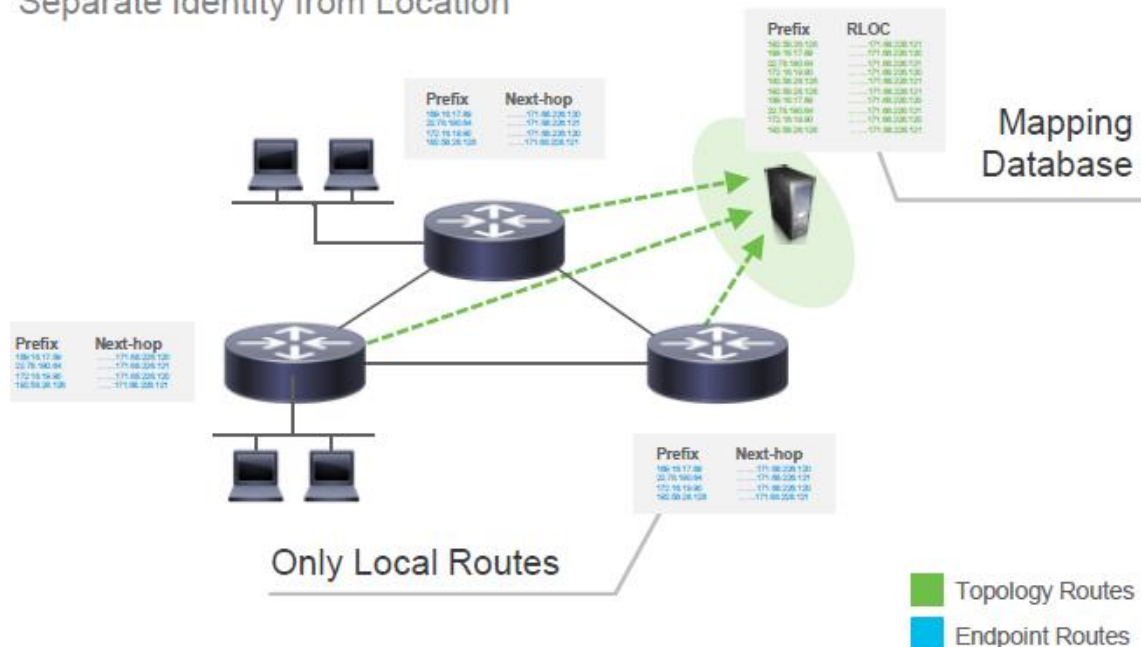


Abbildung 4.6: Generelle Übersicht LISP [6]

LISP bietet einige Vorteile im Vergleich zu anderen Protokollen. Zum Beispiel wird dabei eine Trennung von Identität und Standort des Clients vollzogen, die bei herkömmlicher Verwendung von IP nicht gegeben oder nur mit Einschränkungen zu erreichen ist. Diese Aufspaltung ist bereits in Abbildung 4.6 zu sehen. An dieser Stelle muss man sich vor Augen führen, dass jeder Router einen eigenen Standort bildet und alle Hosts an diese Standorte angeschlossen sind. Dadurch kann jedem Gerät eine Identität, also eine fixe IP-Adresse und ein Standort, meist der Router, an dem dieser Host angeschlossen ist, zugeordnet werden. Überdies lässt sich eine Aufteilung der Routingtabellen der Router erkennen, wobei auf den Routern nur noch lokale Routen gespeichert werden und alle anderen in eine zentrale Datenbank, die bereits erwähnte Host Tracking-Datenbank des Control Plane Nodes, gespeichert werden. Diese Aufteilung der Routingtabelle wird in Abbildung 4.6 ebenfalls dargestellt [6].

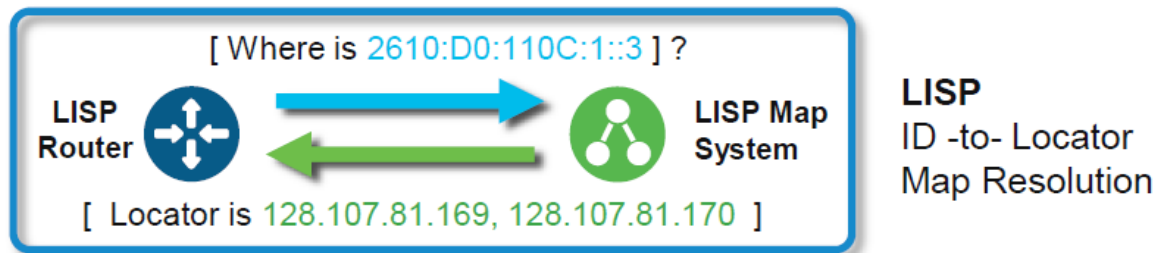


Abbildung 4.7: ID zur Standort-Auflösung [6]

Wie bereits erläutert, werden der Control Plane Node und seine Bestandteile unter anderem dazu genutzt, den Standort einer Identität für die Weiterleitung von Daten zu finden. Abbildung 4.7 stellt eine solche Standortbestimmung eines xTR, eine Kombination aus Egress Tunnel Router und Ingress Tunnel Router, dar. Der Router fragt an, wo die Ziel-IP-Adresse zu finden ist und der Control Plane Node gibt den Standort des Ziels zurück. Diese Standortangabe ist hierbei die IP-Adresse eines anderen xTR, beziehungsweise die IP-Adresse des ETR, an den der Ziel-Host angeschlossen ist. Die IP-Adresse eines Host wird bei LISP als Endpoint-Identität bezeichnet, währenddessen die IP-Adresse eines xTR oder eines ETR als RLOC bekannt ist. Es folgt nun ein Beispiel, ersichtlich in Abbildung 4.8, das helfen soll, diese Vorgänge sowie die Rollen der daran teilnehmenden Geräte besser zu verstehen [6].

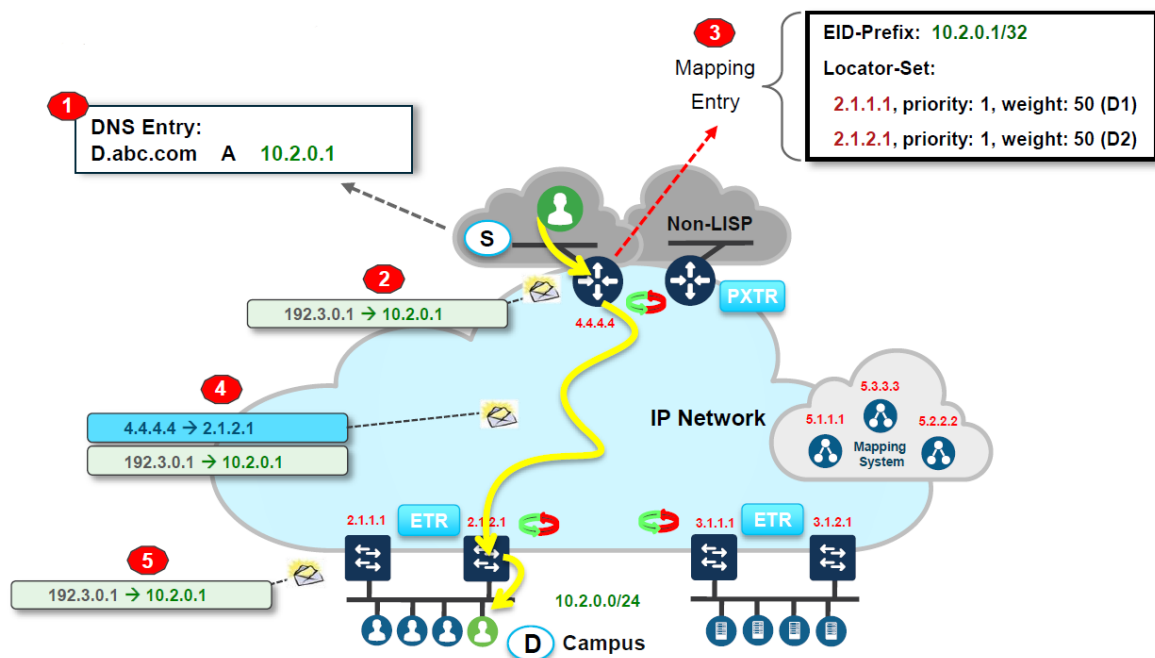


Abbildung 4.8: Simple Beispiel zu LISP [6]

Ein DNS Server liefert innerhalb eines Standorts zu einem angefragten Domain-Namen dessen IP-Adresse. In der Folge erstellt der Host ein IP-Paket, das die IP des Hosts als Source-Adresse und die IP des Domain-Namens als Destinationsadresse beinhaltet und sendet dieses zu seinem Gateway. Das Gateway, also der xTR des Standorts, findet in seinem Cache bereits die EID zu RLOC-Auflösung und nützt diese Information, um das Datenpaket durch die Fabric zu senden. Falls die Auflösung durch den Cache scheitert, kann alternativ eine Anfrage an das Mapping-System gestellt werden. Um das Paket durch das Netzwerk zu transportieren, wird es in ein IP-Paket gekapselt. Im äußeren IP-Header wird als Source-Adresse der RLOC des Source-Standortes eingetragen und die aufgelöste Ziel-RLOC als Destinationsadresse eingetragen. Am ETR angekommen, ist wieder das ursprüngliche Paket vorhanden. Dieses wird innerhalb des Standorts weitergeleitet und gelangt so zu seiner Zieladresse.

Da eine Zuordnung im Mapping-System bei LISP immer aus einem EID und einem Standort besteht, ist es leicht möglich, Multicast über LISP zu realisieren. Dies kann vor allem geschehen, weil eine solche Zuordnung mehrere Standorte, also xTRs, betreffen kann. Dabei wird in den Zielstandorten eine Multicast-Adresse definiert, die mit einer oder mehreren RLOCs assoziiert ist [6].

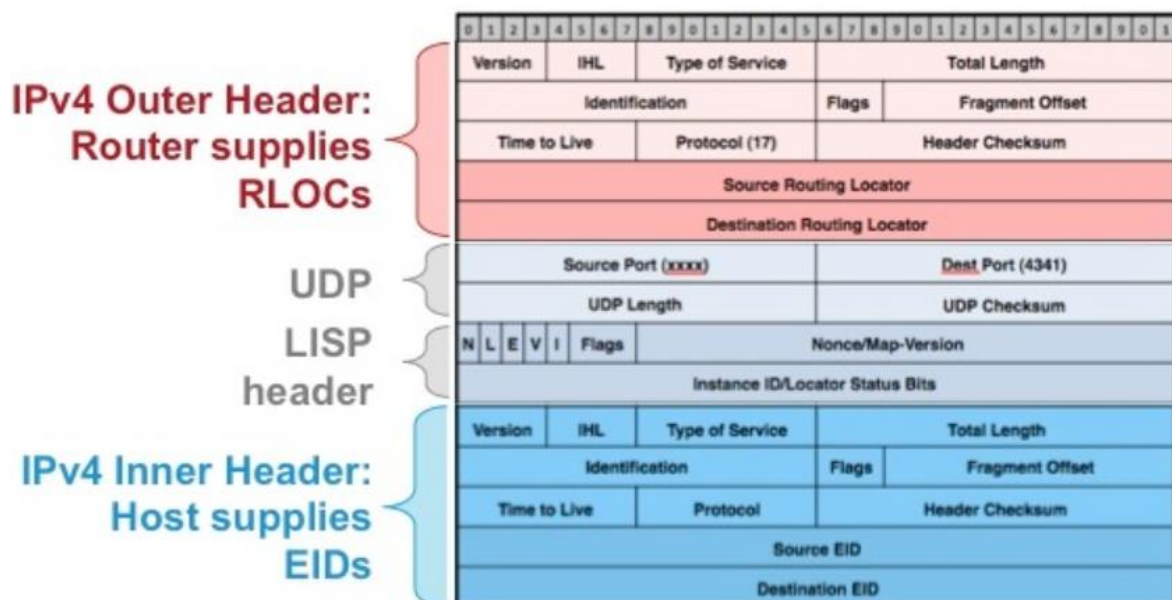


Abbildung 4.9: Darstellung des Datenpakets bei LISP [6]

In Abbildung 4.9 ist zu erkennen, dass bei LISP immer ein Layer 3 Overlay erzeugt wird. Dieses Overlay verhält sich wie bereits beschrieben und kann, wenn

man an allen xTR immer dasselbe Gateway verwendet, Mobilität für Hosts ermöglichen. Dies kann vor allem für Wireless Clients von Vorteil sein [6].

4.3.4. Data Plane

Der Data Plane wird von VXLAN dominiert. VXLAN ist eine Netzwerk-Virtualisierungs-Technologie, die Datenpakete kapseln kann und diese somit über ein existierendes Layer 3-Netzwerk transportiert. Dadurch bildet sich ein Overlay. Dabei hat VXLAN gegenüber VLAN vor allem den Vorteil, dass es auf einem darunterliegenden Routing-Protokoll aufsetzt und daher alle Pfade, die dieses Routing-Protokoll zu anderen Netzwerkkomponenten anbietet, auch verwenden kann. Ein großer Vorteil gegenüber VLAN ist, dass bei VXLAN die VN ID, welche für die Gesamtanzahl der VXLAN-Segmente verantwortlich ist, 24 Bit beträgt und somit über 16 Millionen VXLANs unterstützt. Bei VLAN gibt es im Gegensatz dazu nur 4096 VLANs, wobei einige davon bereits reserviert sind. Diese geringe Anzahl an VLANs und die mangelnde Flexibilität bei der Skalierung führen dazu, dass VXLAN im Datacenter-Bereich immer größere Verwendung findet. Wie bereits erwähnt, kapselt VXLAN das ursprüngliche Paket in UDP. Dabei ist innerhalb dieser Kapselung auch ein VXLAN Header platziert, der ein Flag, eine Segment ID, eine VN ID und acht reservierte Bits enthält. Die Segment ID ist das Feld, in dem das Security Group Tag Platz findet, welches später im Policy Plane eine zentrale Bedeutung bekommt. Der UDP Header selbst weist bei VXLAN die Besonderheit auf, dass das ‚Source Port‘-Feld einen Hashwert enthält, der sich aus den Headern des originalen Pakets zusammensetzt. Der Destination-Port hingegen ist bei VXLAN standardisiert und nimmt den Wert 4789 an. Die Kapselung zwischen den RLOCs führt dazu, dass ein Tunnel entsteht, bei dem die RLOCs gleichzeitig auch als Endpunkte für VXLAN dienen. Diese Endpunkte werden bei VXLAN auch VTP genannt. Um die bereits erwähnte Vergrößerung der MTU um 50 Byte nachvollziehen zu können, muss man einen genaueren Blick auf die Größe und Zusammensetzung des VXLAN-Headers werfen. Einen Überblick über die Struktur eines VXLAN-Pakets kann man in Abbildung 4.10 gewinnen [6], [7], [8].



Abbildung 4.10: VXLAN-Paket [6]

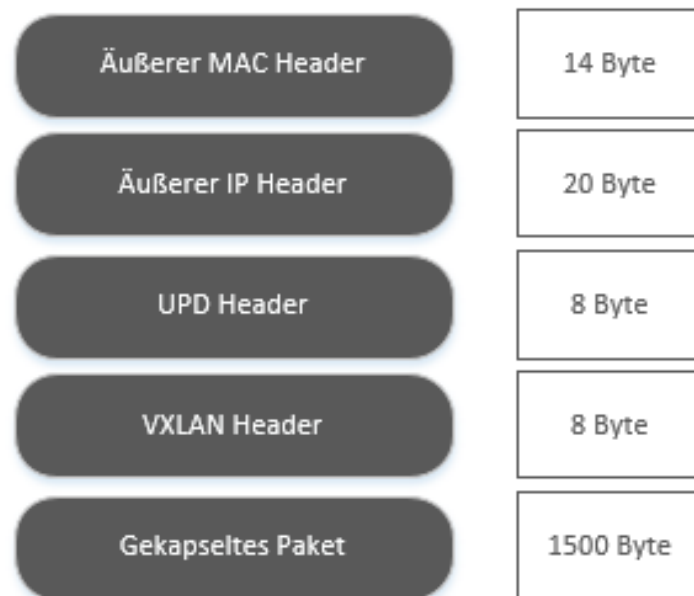


Abbildung 4.11: Größe der VXLAN-Bestandteile

Aus Abbildung 4.11 kann man entnehmen, dass alle Bestandteile des VXLAN Pakets zusammenaddiert 1550 Byte ergeben.

4.3.5. Policy-Plane

Der Policy-Plane ist die finale Ebene innerhalb von DNA und setzt sich aus drei Policy-Typen zusammen: den Zugangsrichtlinien, der Zugangskontrolle und den Applikationsrichtlinien.

4.3.5.1. Zugangsrichtlinien

Die Zugangsrichtlinien beschäftigen sich mit Authentifizierung und Autorisierung, also der Frage, aufgrund welcher Regeln ein Nutzer in einer Gruppe platziert wird. Die Authentifizierung eines Geräts oder Nutzers an der zentralen Anlaufstelle, der ISE, kann über mehrere Wege geschehen. Eine der Methoden, einen Benutzer an der ISE zu authentifizieren, ist MAB. Die Abkürzung MAB steht für MAC Authentication Bypass [4].

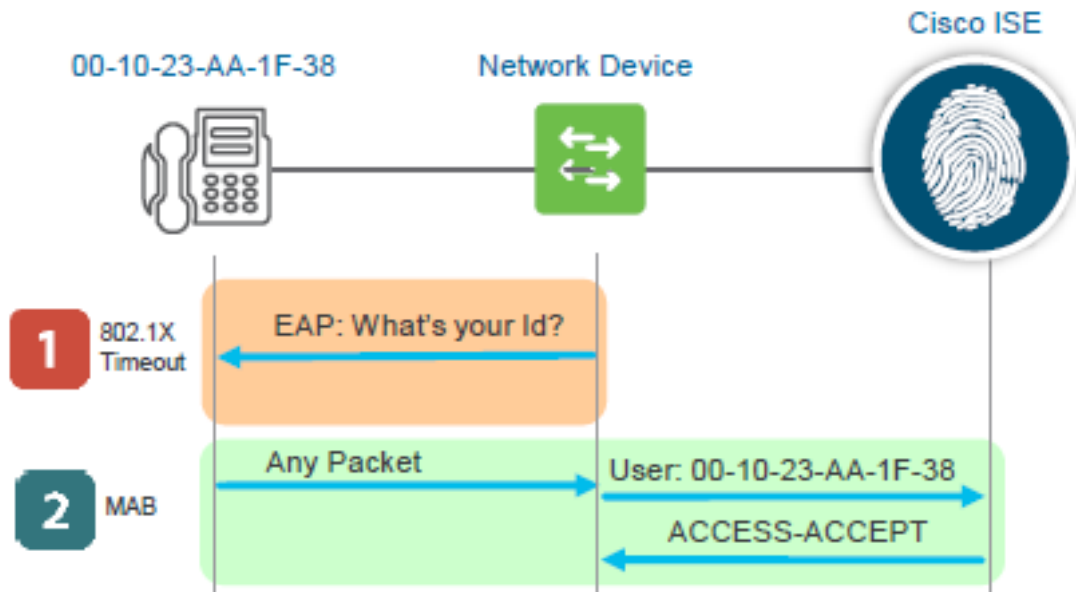


Abbildung 4.12: Ablauf bei der MAB-Methode [4]

Der Ablauf bei MAB kann Abbildung 4.12 entnommen werden. Hierbei führt eine Netzwerkkomponente, die 802.1X unterstützt, im ersten Schritt eine EAP-Anfrage auf einen sich neu im Netz befindlichen Host aus. Dabei wird diese Anfrage nicht beantwortet, weil dieses Gerät kein 802.1X unterstützt. Bei vielen modernen Endgeräten ist eine 802.1X-Unterstützung gegeben, aber bei älteren oder sehr einfachen Geräten, wie zum Beispiel einem simplen Netzwerkdrucker, muss dies nicht der Fall sein. Durch das Fehlen der 802.1X-Unterstützung kann eine 802.1X-Authentifikation nicht stattfinden und der Client wird erst authentifiziert, nachdem er ein beliebiges Paket an die Netzwerkinfrastruktur weitergeleitet hat. Diese gibt an die ISE weiter, dass ein neues Gerät im Netz ist. Die ISE überprüft daraufhin, ob für die MAC-Adresse des Clients in ihrer Datenbank ein zugelassenes Gerät vermerkt ist. Danach wird der Netzwerkinfrastruktur mitgeteilt, dass der Host am Netzwerk teilnehmen darf und lässt Datenpakete von und zum Host passieren [4].

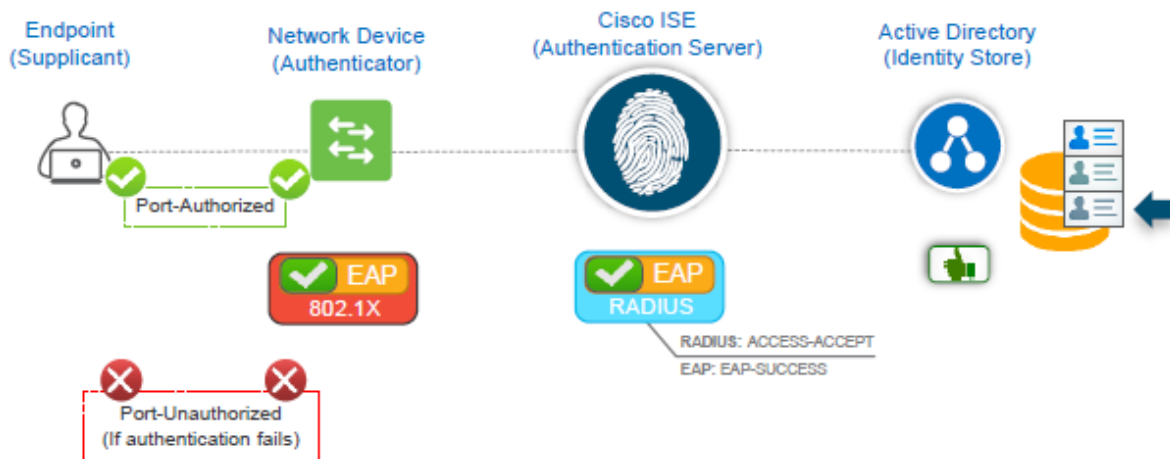


Abbildung 4.13: 802.1X-Methode [4]

In Abbildung 4.13 findet man die Authentifizierungsmethode über 802.1X. Bei dieser Methode schickt der Host, der in diesem Fall 802.1X unterstützen muss, ein 802.1X-Paket mit einer EAP-Nachricht an den Authenticator. Der Authenticator kann in diesem Fall ein Accesspoint sein, der beim Erhalt der Nachricht eine Anfrage an den Authentication Server stellt, ob der Nutzer berechtigt ist am Netzwerk teilzunehmen. Die Rolle des Authentication Servers übernimmt bei DNA immer die ISE. Die an die ISE angebundene Identität-Datenbank kann, wie in diesem Beispiel, ein ActiveDirectory sein [4].

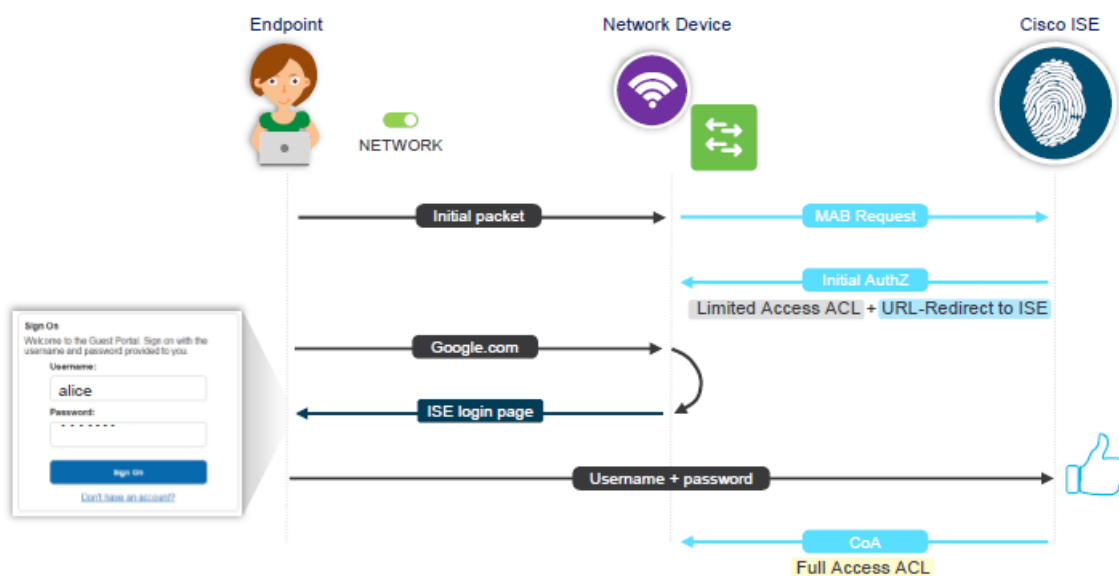


Abbildung 4.14: Central Web Authentication-Methode [4]

Ein kurzer Überblick über die Authentifizierung mittels CWA findet sich in Abbildung 4.14. Dabei ist zu sehen, dass die MAC-Adresse eines Host, der sich in das Netzwerk verbindet, durch eine zwischengeschaltete Netzwerkkomponente an die ISE weitergegeben wird. Diese entscheidet daraufhin mithilfe verschiedener

Parameter, ob eine Weiterleitungsnachricht an den Host gesendet werden soll oder nicht. Falls eine Nachricht gesendet wird, werden für diesen Host automatisch ACLs erstellt, die einen geringen Grad an Kommunikation zwischen ISE und Host ermöglichen. Wenn der Host daraufhin in einem Browser eine URL ansteuert, wird aufgrund der Weiterleitungsnachricht die Anmeldeseite der ISE angezeigt, auf der sich der Nutzer mit seinen Anmeldedaten authentifiziert. Diese werden an der ISE überprüft und daraufhin werden die Zugriffsrechte des Nutzers über ACL erweitert beziehungsweise modifiziert [4].

4.3.5.2. Zugangskontrolle

Das Thema Zugangskontrolle beschäftigt sich vor allem mit der Frage, welcher Netzteilnehmer bestimmte andere Teilnehmer erreichen darf. Wie bereits erwähnt, werden Hosts in Gruppen unterteilt. Die Zuteilung eines Host zu einer Gruppe geschieht durch die ISE. Diese Gruppen werden mittels SGT, auch Scalable Group Tag genannt, identifiziert und sind Richtlinien unterworfen. Diese Richtlinien werden dann auf den Netzwerkgeräten mittels SGACLs implementiert. SGACL ist dabei die Abkürzung für Scalable Group Access Control List [4].

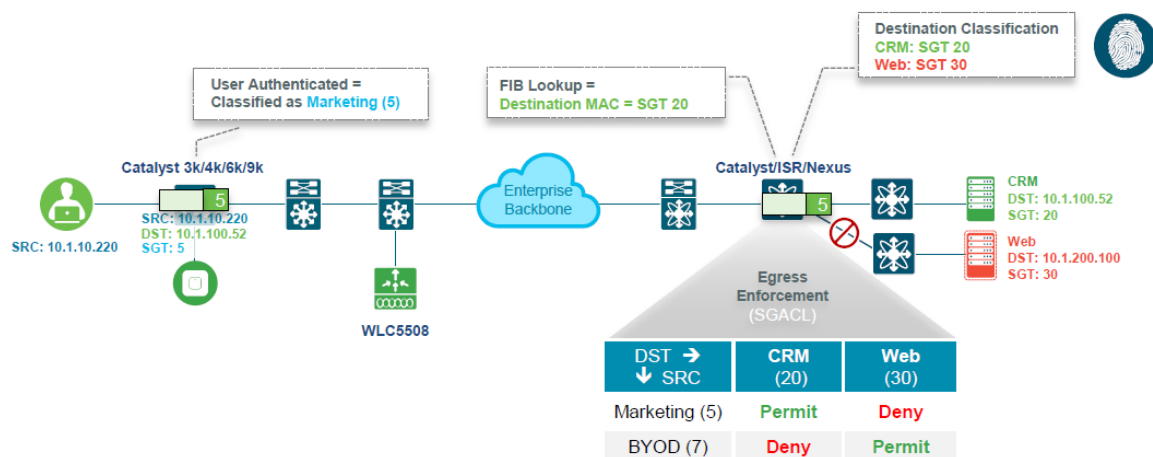


Abbildung 4.15: Beispiel für eine Verwendung von Gruppen [4]

Aus dem Beispiel in Abbildung 4.15 ist zu erkennen, dass ein Nutzer mit einem SGT 5 nicht mit der Gruppe Web, jedoch aber mit der Gruppe CRM kommunizieren darf.

4.3.5.3. Applikationsrichtlinien

Applikationsrichtlinien sind das letzte Glied in der Kette der Richtlinien, die in DNA umgesetzt werden können. Dabei spielt bei diesem Punkt vor allem QoS die

Hauptrolle. Um Applikationen besser und effizienter verwalten zu können, werden sie in ein eigenes Profil gekapselt. Dabei besteht jedes Profil aus einem Namen, Klassifikatoren und mindestens einem Endpunkt. Ein Endpunkt kann über seine Gruppe, eine IP oder eine URL angegeben werden. Die Klassifizierung eines Endpunkts erfolgt über die verwendete Portnummer des Service oder über Differentiated Services Code Points. Dadurch kann darauffolgend jede einzelne Komponente entscheiden, wie verfahren werden soll [4].

4.3.6. DNA-Center

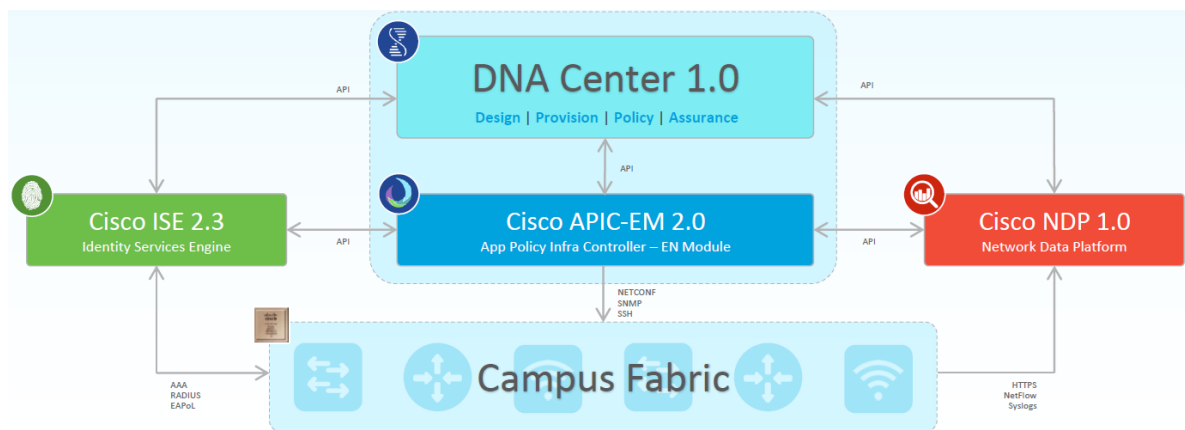


Abbildung 4.16: Planübersicht über DNA-Komponenten [6]

Bei DNA nimmt das DNA-Center eine zentrale Rolle ein. Es ist mit allen Komponenten im Netz direkt oder indirekt in Kontakt. All diese Komponenten können jedoch dem DNA-Center zugerechnet werden, weil es als Schnittstelle und Aggregationspunkt für alle anderen Komponenten dient. Zum Beispiel kommuniziert das DNA-Center mit der ISE über SSH, deren REST-APIs und pxGRID. Andere Komponenten, mit denen das DNA-C in Verbindung steht, sind das APIC-EM und die NDP. Ein Überblick über diese Verbindungen kann aus Abbildung 4.16 gewonnen werden [4], [6].

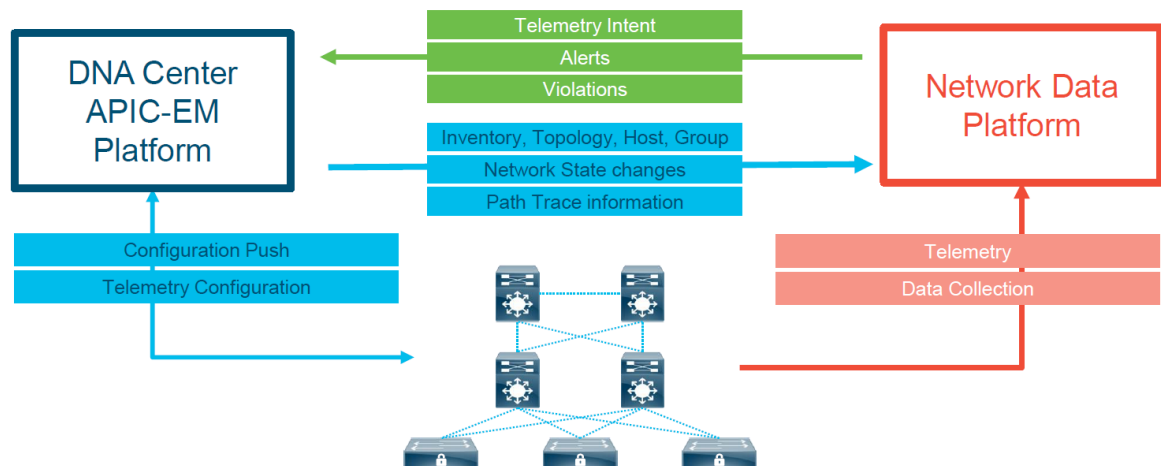


Abbildung 4.17: Austausch von Nachrichten zwischen DNA-Center und NDP [6]

In Abbildung 4.17 ist zu erkennen, dass die vom DNA-C und APIC-EM gemachte Konfiguration des Netzes durch die NDP ausgewertet wird. Falls Probleme entdeckt werden, werden diese aufgezeigt und Lösungsvorschläge zu deren Beseitigung an das DNA-C übermittelt. Um diese Funktionen der NDP zu realisieren, hat Cisco maschinelles Lernen in diese Plattform integriert. Diese dient hauptsächlich dazu, die erhaltenen Daten aus dem Netzwerk zu analysieren. Durch die so unterstützte Analyse soll gleichzeitig ermöglicht werden, dass Trends rechtzeitig erkannt werden und so Fehlern vorgebeugt werden kann. Um Fehler zu analysieren, wird auf ein Schema zurückgegriffen, das sich in vier Schritte unterteilt. Als erstes werden Symptome erkannt, daraufhin die Fehler lokalisiert, um zu ermöglichen, dass eine Ursache festgestellt werden kann und schlussendlich ein Lösungsvorschlag für das Problem bereitgestellt wird [4], [9].

Das DNA-Center an sich wird in vier Aufgabenbereiche unterteilt. Der erste Aufgabenbereich ist Design. Hierbei werden essentielle Grundeinstellungen gemacht, wie zum Beispiel die Definition von IP Pools zur Ermöglichung eines Adressmanagements von Standorten, oder auch die Definition von SSIDs in Verbindung mit anderen grundlegenden WLAN-Einstellungen. Hinzukommend kann für jeden Standort ein genauer Lageplan erstellt werden, um den Standort von Komponenten, wie zum Beispiel Accesspoints, darin festzuhalten und, um alle Standorte in einen visuellen Kontext zu bringen. Kurzum, in diesem Bereich wird das grundlegende Setup und die Erreichbarkeit des Underlayer behandelt [9].

Der zweite Aufgabenbereich sind Richtlinien-Einstellungen. Unter diesen Punkt fallen das Definieren und Konfigurieren von VN, das Erstellen von skalierbaren

Gruppen und die Versorgung dieser Gruppen mit Richtlinien. Zusammenfassend kann gesagt werden, dass in diesem Aufgabenbereich die meisten Einstellungen hinsichtlich Zugangsrichtlinien und Segmentierung getroffen werden [9].

Der dritte Aufgabenbereich behandelt vor allem das Overlay und den Data-Plane. Hierzu kann man das Hinzufügen von Geräten zu den im ersten Aufgabenbereich erstellten Standorten zählen. Überdies werden hier die Endpunkte und Komponenten, welche die DNA ausmachen, hinzugefügt und deren Rolle bestimmt [9].

Der vierte und letzte Aufgabenbereich behandelt die Darstellung der ‚Gesundheit‘ und Statistiken von Clients, Netzwerkkomponenten und Services. Dabei werden diese mit einfachen Mitteln, wie zum Beispiel einem Health-Score, dargestellt. Das Aufzeigen von alarmierenden Ereignissen, die besagte Geräte betreffen, ist auch in diesem Aufgabengebiet angesiedelt [9].

Um die Integration von bereits bestehender Infrastruktur und Abläufen zu garantieren, hat Cisco, wie bei vielen anderen Komponenten von diesem Konzept, eine API für das DNA-Center bereitgestellt. Jedoch wird die Funktionalität des DNA-Centers mit der Zeit noch anwachsen und daher ist die Programmierschnittstelle noch nicht im vom Cisco geplanten Umfang vorhanden [6].

5. Programmierschnittstellen – Anwendungsbeispiel 1

In diesem Kapitel wird das erste Anwendungsbeispiel vorgestellt, das sich mit Cisco DNA und deren Programmierschnittstellen beschäftigt.

5.1. Problemstellung und Beschreibung

Netzwerke werden immer mehr virtualisiert und daher kann die Ausrichtung des Firmenbetriebs in diese Richtung nicht ausbleiben. Gleichzeitig ist man überzeugt, dass Cisco DNA auch in Zukunft im SDN-Bereich eine große Rolle spielen wird. Dadurch würden sich viele Firmen aber auch immer weiter von einem Hardwarekonfigurationsexperten zu einem Softwareentwicklungsexperten wandeln. Um die Entwicklungen in diesem Bereich zu beschleunigen, gibt es von Cisco sogenannte ‚DNA Programmability QuickBet‘-Programme. Diese sollen vor allem dazu dienen, firmeninternes Wissen über diese Technologien aufzubauen und eigene Software zu entwickeln, um weiterhin am Markt bestehen zu können.

Viele Firmen der Telekommunikationsbranche haben über die Jahre ein eigenes Management- und Monitoringtool entwickelt. Diese bieten Kunden oftmals den Vorteil, plattform- und herstellerunabhängig, auf flexible Art und Weise, umfassenden Einblick in das Netzwerk zu gewinnen. Die Integration von Cisco DNA in solche Tools stellt hierbei eine Herausforderung dar. Um diese Herausforderung besser bewältigen zu können, kann vorab ein Prototyp getestet werden. Zur Erstellung eines solchen Prototyps werden Programmierschnittstellen der Infrastruktur- und Controllerebene verwendet.

Das Programm teilt sich dabei in drei Phasen beziehungsweise Meilensteine auf. Der erste Meilenstein ist die technische Umsetzung. Um die Anforderungen erreichen zu können, werden testweise sowohl physische als auch virtuelle Netzwerkkomponenten installiert. In Phase zwei wird die Technologie meist einigen ausgewählten Stammkunden präsentiert und angeboten. Dabei muss der Kunde Interesse an der Technologie mitbringen, bereits länger Kunde des ausführenden Unternehmens sein, hauptsächlich oder ausschließlich Hardware von Cisco betreiben sowie DNA-fähige Geräte besitzen oder die Bereitschaft haben, in diesen Bereich zu investieren. Der letzte Meilenstein beschäftigt sich vor allem mit der Generierung der Nachfrage bei einem breiteren Kundenstamm. Dieser Meilenstein sollte durch Erfolgsberichte aus der vorangehenden Phase

beflügelt werden. Generell ist zu sagen, dass dies kein einseitiges Programm zum Nutzen der ausführenden Firmen verkörpert, sondern eine kooperative Partnerschaft zwischen Cisco und dem Unternehmen ist. Die finanziellen Mittel für die Durchführung des Programms werden sowohl von Cisco als auch vom Unternehmen selbst getragen. Die Geldmittel werden allem voran für Softwareentwicklung, Lizenzen und benötigte Netzwerkhardware verwendet. Die Erreichung der Meilensteine wird daher von Cisco überwacht und das Programm erfolgsbasierend weiter unterstützt.

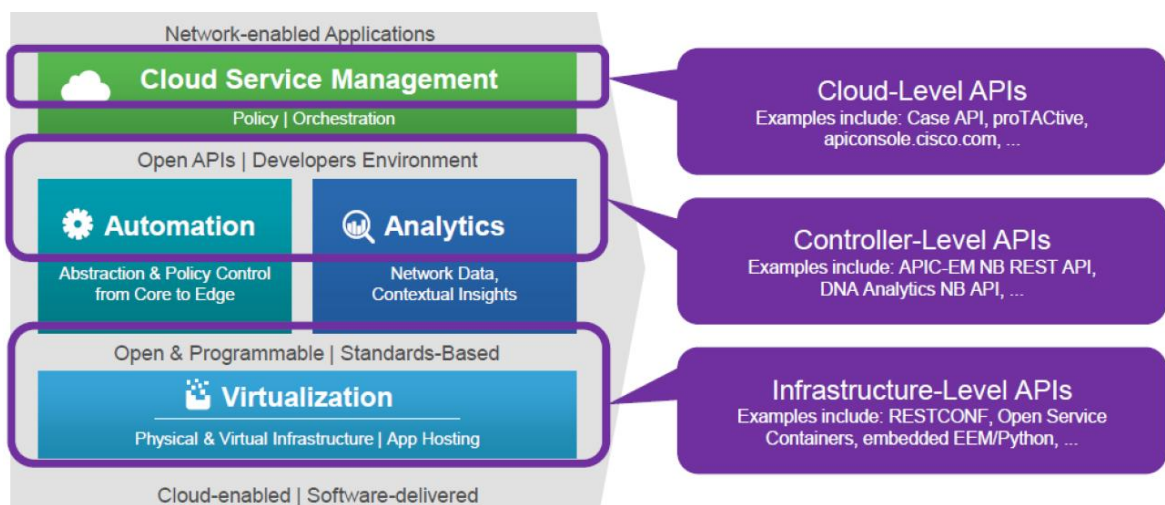


Abbildung 5.1: Überblick über die verfügbaren Programmierschnittstellen [10]

In Abbildung 5.1 sind die möglichen APIs zu sehen. Zwei davon mussten ausgewählt werden, um die Bedingungen von Cisco zu erfüllen. Nichtsdestotrotz fiel, wie bereits erwähnt, die Wahl auf eine Controller-Level API (APIC-EM REST API) und eine Infrastructure-Level API (RESTCONF). Eine API aus der Cloud-Ebene wurde nicht verwendet.

Die APIC-EM-Programmierchnittstelle wurde verwendet, um Topologie- und Geräteinformationen auszulesen. Die Infrastruktur-API, mit der einige virtuelle Router angesprochen werden, wurde dazu verwendet, die jeweiligen Netzwerkkomponenten zu konfigurieren.

5.2. Beschreibung des Use Case

Das für das Managementtool zu erstellende ‚DNA Modul‘ und alle Prototypen sollten in PHP geschrieben werden, um eine Kompatibilität mit der bereits existierenden Software zu erreichen. Im Prototyp sollte ein HTTP GET-Befehl auf das APIC-EM abgesetzt werden, der das Auslesen von IP-Adressen ermöglicht.

Diese Funktionalität sollte dann im ‚DNA Modul‘ ausgebaut werden, damit dem Nutzer zusätzliche Informationen visuell präsentiert werden können. Dabei sollte der Nutzer dann mit seinem Mauszeiger auf eine der Netzwerkkomponenten klicken können, um einige IP SLA-Parameter konfigurieren zu können. Unter diesen Parametern finden sich der IP SLA Responder, ein IP SLA-Operationstyp, ein Tag-Name und andere Parameter. Darauffolgend sollte der Nutzer seine Angaben und Parameter überprüfen können, um in der Folge auf dem ausgewählten Gerät die IP SLA-Operation durch das Managementtool konfigurieren zu lassen. Dies geschieht wiederum durch die Erstellung eines HTTP POST-Request sowie der Verwendung von RESTCONF. Die JSON-Meldung, die dabei abgegeben wird, referenziert zu einem YANG-Model und einer adäquaten URI. Diese abgesetzte HTTP-Meldung setzt dann auf dem Gerät die Konfiguration um. Das Absetzen der Konfiguration an die Netzwerkkomponente sollte im Prototyp bereits erprobt und mittels eines CLI umgesetzt sein. Alle bereits zu diesem Zeitpunkt getätigten IP SLA-Konfigurationen sollen durch die Benutzeroberfläche des Managementtools verwaltet und überwacht werden können. Bei der Überschreitung eines Schwellenwerts soll ein Alarm ausgelöst werden. Dies sollte in einem ‚IP SLA Modul‘ realisiert werden.

Daraus lässt sich ableiten, dass der Use Case in zwei unterschiedliche Use Cases, nämlich das Auslesen von Topologie-Information und das Management und die Überwachung von IP SLA, aufgeteilt werden kann. Diese zwei voneinander unabhängigen Use Cases sollten am Ende der Entwicklung ineinandergreifen und ein funktionierendes Produkt bilden. Am Ende der Entwicklung ermöglicht das Produkt den Kunden die Konfiguration ihrer Netzwerkperformance-Überwachung zu automatisieren.

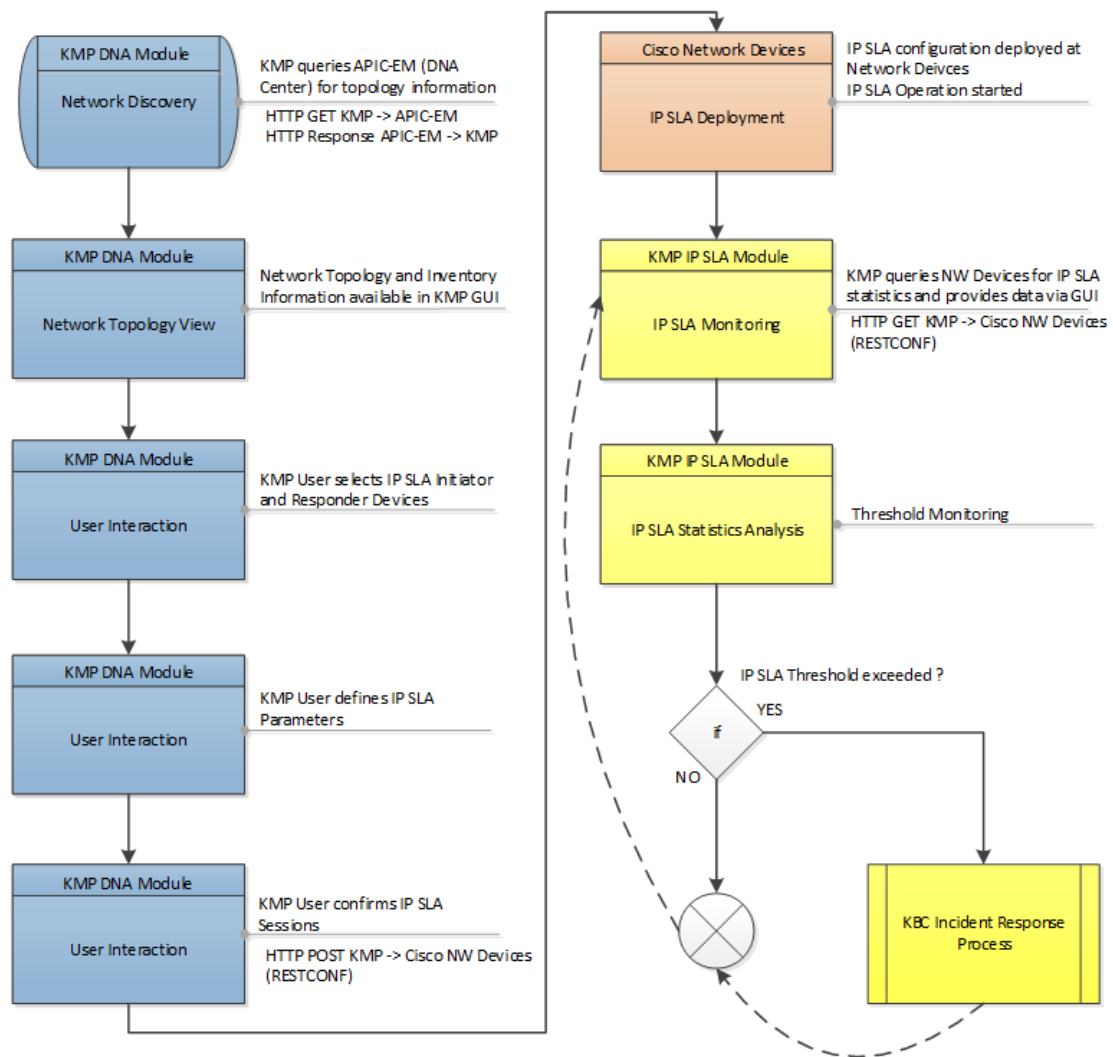


Abbildung 5.2: Überblick des gesamten Use Cases [10]

In Abbildung 5.2 ist ein Überblick über den gesamten Use Case zu sehen. Alle darin abgebildeten Funktionen werden im Managementtool durch die Summe der Teilmodule, also dem DNA- und dem IP SLA-Modul, realisiert.

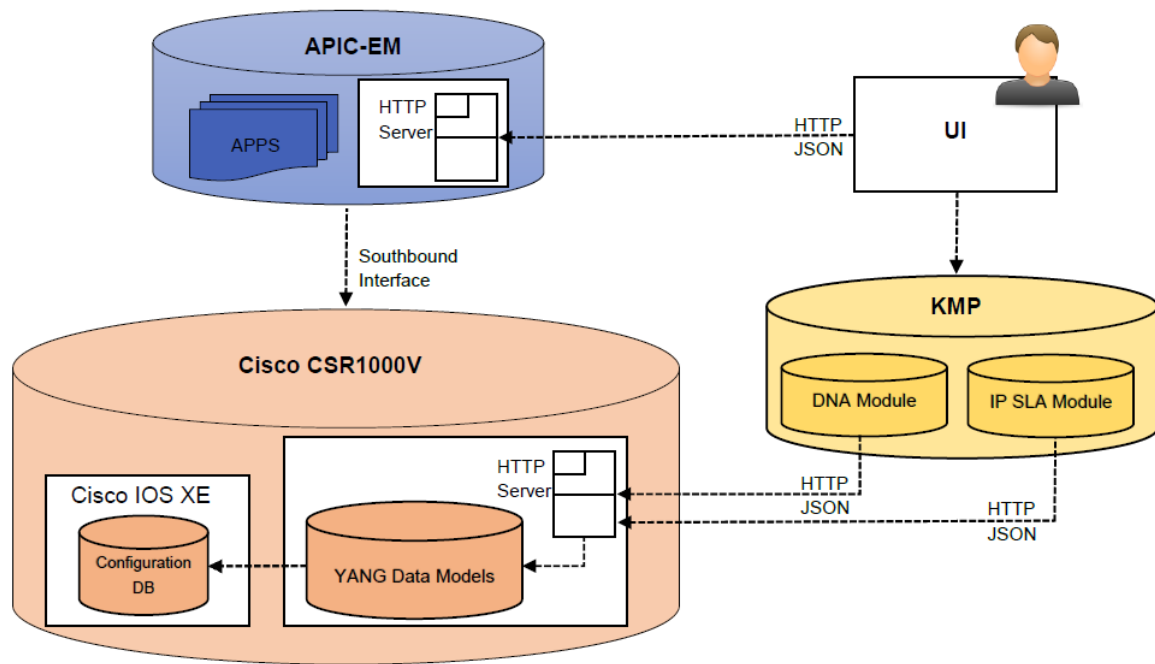


Abbildung 5.3: Abbildung der finalen Lösung [10]

Abbildung 5.3 zeigt, wie die einzelnen Teilmodule im Managementtool zusammengefasst sind und wie APIC-EM, das Managementtool, die Benutzeroberfläche und die Testhardware ineinandergreifen.

5.3. APIC-EM und dessen API

Wie bereits in den vorangegangenen Abschnitten beschrieben, ist APIC-EM eine zentrale Komponente von DNA. Es bietet dem Nutzer die Möglichkeit, das Netz von einem zentralen Zugangspunkt aus zu überblicken und stellt die zentrale Anlaufstelle für alle neu in das Netz eingebrachten Netzwerkkomponenten dar. Das Zusammenspiel aus Cisco Network Plug-and-Play und APIC-EM ermöglicht, dass das Einfügen von neuen Netzwerkkomponenten sehr vereinfacht wird und mit geringem Aufwand verbunden ist. Ein weiteres Feature dieser Lösung ist das intelligente Management des Datenstroms beim Verlassen des Netzwerks in Richtung WAN. Hierbei wird dem Thema QoS besonders große Aufmerksamkeit geschenkt. Durch die Enterprise Service Automation-Funktion wird die Einbringung von neuen Services sowie das Management von bestehenden Services im Netzwerk vereinfacht. Außerdem wird durch APIC-EM den Netzwerkadministratoren ein Feature, das auch in DNA existiert, bereitgestellt. Dieses Feature nennt sich ‚Path Trace‘ und stellt eine neue Entwicklung von Cisco dar. Mit ‚Path Trace‘ wird dem Wartungspersonal ein einfaches Inspizieren von

Netzwerkproblemen ermöglicht, weil dadurch Geräteinformationen, Zeitstempel und viele andere nützliche Daten dem Wartungspersonal übersichtlich in einer Benutzeroberfläche dargestellt werden [11].

Die durch die API des APIC-EM ausgelesenen Informationen werden dazu verwendet, die Anzahl und Information der Geräte, die sich ihrerseits über eine eigene Infrastruktur-API ansprechen lassen, zu eruieren. Genauer gesagt sollten vor allem die IP-Adressen und die Verbindungen der Geräte untereinander aus dem APIC-EM ausgelesen werden. Um diese Informationen trotzdem über die API auszulesen, muss zuerst eine Abfrage an den APIC-EM abgesetzt werden, bei der die Antwort ein Servicetoken enthält. Ohne dieses Servicetoken kann der Nutzer keine oder nur wenige nützliche Informationen mittels der Programmierschnittstelle auslesen.

POST

/ticket

addTicket

Implementation Notes

This method is used to create a new user ticket

Response Class

Model | Model Schema

```
{
  "version": "",
  "response": {
    "idleTimeout": 0,
    "sessionTimeout": 0,
    "serviceTicket": ""
  }
}
```

Response Content Type: application/json

Parameters

| Parameter | Value | Description | Parameter Type | Data Type |
|-----------|---|-------------|----------------|--|
| user | <div><div></div><div>Parameter content type: application/json</div></div> | user | body | <div>Model Model Schema</div> <div>User { password (string): password, username (string): username }</div> |

Error Status Codes

| HTTP Status Code | Reason |
|------------------|--|
| 204 | The request was successful, however no content was returned. |
| 409 | The target resource is in a conflicted state (for example, an edit conflict where a resource is being edited by multiple users). Retrying the request later might succeed. |
| 206 | The GET request included a Range Header, and the server responded with the partial content matching the range. |
| 201 | The POST/PUT request was fulfilled and a new resource has been created. Information about the resource is in the response body. |
| 504 | The server did not respond inside time restrictions and timed-out. |
| 415 | The client sent a request body in a format that the server does not support (for example, XML to a server that only accepts JSON). |
| 200 | The request was successful. The result is contained in the response body. |
| 202 | The request was accepted for processing, but the processing has not been completed. |
| 500 | The server could not fulfill the request. |

Abbildung 5.4: Struktur der Tokenabfrage

In Abbildung 5.4 ist die Beschreibung einer Programmierschnittstelle im Webinterface des APIC-EM zu sehen. In dieser Beschreibung kann man einen Beschreibungstext der jeweiligen Aktion lesen, die Struktur der von der API erwarteten Daten analysieren und die Struktur der Antwortnachricht vorab begutachten. Auch findet sich im unteren Teil dieser Schnittstellenbeschreibung eine Erklärung zu den möglichen Statuscodes und deren Bedeutung. Zudem hat ein Entwickler innerhalb einer Schnittstellendokumentation die Möglichkeit, eine Abfrage an diese Schnittstelle innerhalb der Benutzeroberfläche des APIC-EM zu starten und so die Fähigkeit Fehlerquellen vor der eigentlichen Verwendung der Programmierschnittstelle zu entdecken. Für das Auslesen der physischen Topologie, siehe Abbildung 5.5, wird dieselbe Struktur der Schnittstellenbeschreibung wie in Abbildung 5.4 verwendet. Diese genaue Strukturbeschreibung hilft, gezielt Teilinformationen einer Nachricht mittels eines Programms oder Scripts auszulesen.

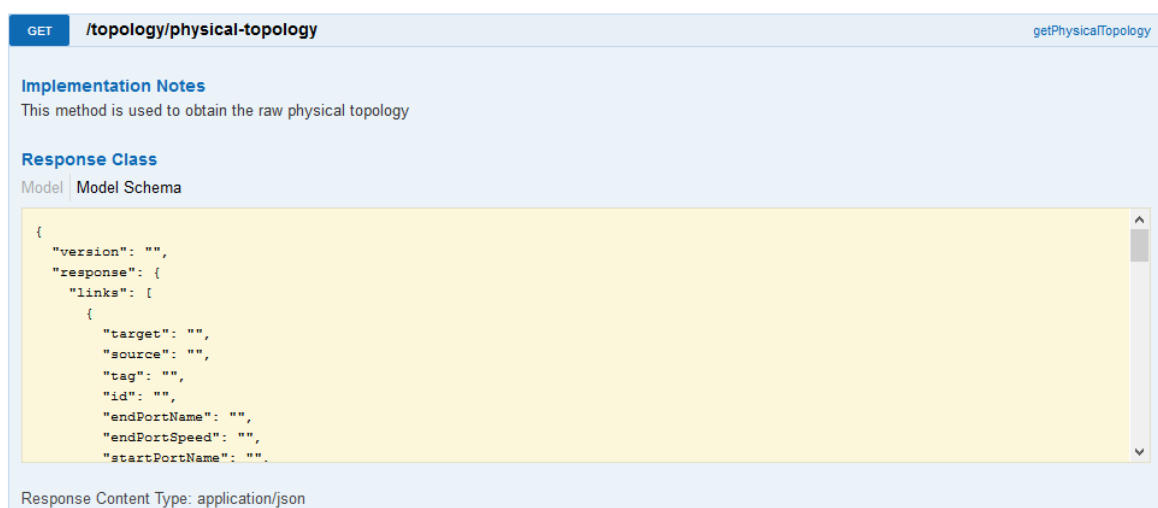


Abbildung 5.5: Abfrage der physischen Topologie

```
curl -s -k -H "Content-Type: application/json" -X POST -d '{"username":
"admin", "password": "[REDACTED]}' https://[REDACTED]/api/v1/ticket
| jq "."
{
  "version": "1.0",
  "response": {
    "sessionTimeout": 21600,
    "idleTimeout": 1800,
    "serviceTicket": "[REDACTED]"
  }
}
```

Abbildung 5.6: cURL-Abfrage Token

Die Erreichbarkeit und Funktionalität der Schnittstelle kann nach Durchsicht der Schnittstellenbeschreibung auch über cURL oder Google Postman überprüft und getestet werden. Die Methodik, die Programmierschnittstelle mittels cURL zu nutzen, findet sich auch in den erstellten Programmen wieder. In Abbildung 5.6 sieht man die Antwort des APIC-EM, wenn über cURL ein Authentifizierungstoken angefordert wird. Dieses wird bei weiteren Anfragen an die Programmierschnittstellen anstatt des Benutzernamens und des Passworts übermittelt. Dabei wird das Token als Header-Information in die Anfrage eingebettet. Diese Vorgangsweise ist in Abbildung 5.7 zu sehen. Dieselbe Abbildung zeigt auch einen Teil der Topologieinformation, welche durch die Programmierschnittstelle ausgegeben werden kann. Die dargelegten Informationen zeigen, dass zwei Router gefunden wurden. Die Abfrage aus Abbildung 5.7 liefert nicht nur Informationen über die betriebenen Geräte, sondern auch über die Verbindungen zwischen diesen Geräten. Dabei werden die Identifikationsnummern, die in Abbildung 5.7 die Schlüsselbezeichnung ‚id‘ tragen, genutzt, um Ziel- und Quellgeräte zu kennzeichnen.

```
curl -s -k -X GET -H "X-Auth-Token: [REDACTED]"
https://[REDACTED]/api/v1/topology/physical-topology | jq "." {
  "response": {
    "nodes": [
      {
        "deviceType": "Cisco Cloud Services Router 1000V",
        "label": "CSR1000V_DNA_VIE.lab.lcl",
        "ip": "[REDACTED]",
        "softwareVersion": "16.6.2",
        "nodeType": "device",
        "family": "Routers",
        "platformId": "CSR1000V",
        "tags": [],
        "role": "BORDER ROUTER",
        "roleSource": "AUTO",
        "customParam": {},
        "id": "374366e2-0dc0-4b33-a16e-b30329109e0b"
      },
      {
        "deviceType": "Cisco Cloud Services Router 1000V",
        "label": "CSR1000V_DNA_GRAZ",
        "ip": "[REDACTED]",
        "softwareVersion": "16.6.2",
        "nodeType": "device",
        "family": "Routers",
        "platformId": "CSR1000V",
        "tags": [],
        "role": "BORDER ROUTER",
        "roleSource": "AUTO",
        "customParam": {},
        "id": "649d74ca-fb02-42d3-a0a8-152eddc9e27a"
      }
    ]
  }
}
```

Abbildung 5.7: cURL-Abfrage der physischen Topologie

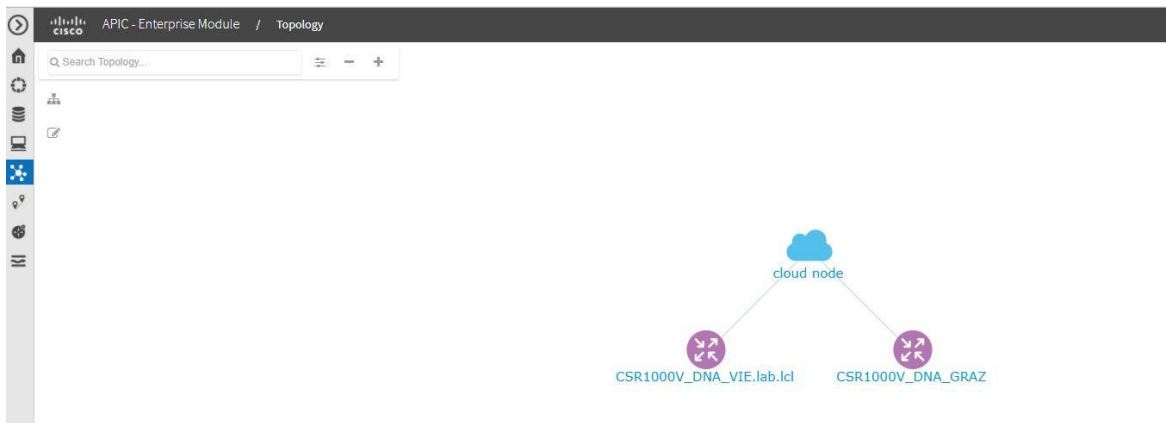


Abbildung 5.8: Topologie des Testsystems in APIC-EM

Abbildung 5.8 zeigt die visuelle Repräsentation der Topologieinformation aus Abbildung 5.7 innerhalb der Benutzeroberfläche des APIC-EM.

5.4. Infrastruktur API

Wie bereits erwähnt, kann die aus den Programmierschnittstellen aus dem APIC-EM gewonnene Topologieinformation genutzt werden, um die darin aufscheinenden Geräte über deren Infrastruktur API anzusprechen. Bei der verwendeten Hardware, zwei CSR1000v-Modelle, wird die API mittels YANG realisiert. Datenmodelle bieten meist eine hohe inhaltliche Tiefe und weil diese auch bei YANG gegeben ist, folgt eine kurze Vorstellung von YANG.

5.4.1. YANG

Da der YANG-Standard sehr umfangreich ist, werden in diesem Abschnitt nur die für das weitere Verständnis essentiellen grundlegenden Gegebenheiten und Themen behandelt.

5.4.1.1. Überblick

Die inhaltliche Tiefe, die geringe Verbreitung innerhalb des klassischen IT-Umfelds, die Vielzahl von Strukturen und die damit verbundene, sehr stark ansteigende Lernkurve machen YANG anfangs unübersichtlich und verlängern den Lernprozess, vor allem sobald komplexere Strukturen verwendet werden. Daher werden in diesem Kapitel einige wenige Strukturen, Tools, die den Einstieg erleichtern, und einige Repräsentationsarten vorgestellt.

YANG ist eine Datenmodellierungssprache, die ursprünglich dafür vorgesehen war, Konfigurations- und Statusdaten über NETCONF darzustellen. Folglich kann

und soll YANG laut Definition nicht alle möglichen Probleme lösen können, sondern in seinen Anwendungsbereichen reduziert bleiben. Seit dem Erscheinen der ersten Version wurden in der Praxis oftmals auch andere Protokolle, wie zum Beispiel RESTCONF, mit YANG in Verbindung gebracht. YANG modelliert die hierarchische Organisation von Daten in Form eines Baums. Jeder Knoten des Baums besitzt dabei einen eigenen Namen, eine kurze Beschreibung und entweder einen Wert oder eine Menge an Kinderknoten. Datenmodelle werden durch YANG in Module und Submodule eingeteilt. Ein Modul kann Definitionen sowohl aus externen Modulen importieren als auch aus Submodulen inkludieren. Zusätzlich kann die Hierarchie eines Moduls erweitert werden, indem Datenknoten eines anderen Moduls hinzugefügt werden. Dieses Hinzufügen kann von unterschiedlichen Bedingungen abhängig gemacht werden. Ein wichtiger Baustein von YANG sind Datentypen. Es gibt zwei verschiedene Arten von Datentypen: einerseits standardmäßig enthaltene und andererseits abgeleitete, also neu entworfene, Datentypen [12].

5.4.1.2. Strukturen

```
module Cisco-IOS-XE-sla {  
  namespace "http://cisco.com/ns/yang/Cisco-IOS-XE-sla";  
  prefix ios-sla;  
  
  import ietf-inet-types {  
    prefix inet;  
  }  
  
  import Cisco-IOS-XE-native {  
    prefix ios;  
  }  
  
  organization  
    "Cisco Systems, Inc.";  
  
  contact  
    "Cisco Systems, Inc.  
    Customer Service  
    Postal: 170 W Tasman Drive  
    San Jose, CA 95134  
    Tel: +1 1800 553-NETS  
    E-mail: cs-yang@cisco.com";  
  
  description  
    "Cisco XE Native Service Level Agreements (SLA) Yang model.  
    Copyright (c) 2016 by Cisco Systems, Inc.  
    All rights reserved.";
```

Abbildung 5.9: Module-Header von Cisco-IOS-XE-sla.yang

Ein Modul ist die Basiseinheit für Definitionen in YANG. Ein Modul definiert ein einziges Datenmodell. Wie schon erwähnt, kann ein Modul ein bereits existierendes Datenmodell mit zusätzlichen Knoten erweitern. Die dabei erwähnten Submodule sind Teilmodule, die zur Definition eines Moduls beitragen und daher nur zu einem Modul gehören dürfen. Das heißt, dass diese Submodule nur zu einem gesamten Namensraum beitragen. Hingegen haben Module keine Begrenzung auf die Anzahl der in ihnen enthaltenen Submodule. Die Submodule eines Moduls werden mittels einer ‚include‘-Struktur aufgelistet. Die ‚import‘-Struktur ermöglicht es, dass Definitionen innerhalb eines Moduls sich auf externe Definitionen eines anderen Moduls beziehen können. Dies geschieht über ein Prefix, das innerhalb der jeweiligen ‚import‘-Struktur definiert und für das Modul einzigartig sein muss. Ein Modul enthält immer drei Arten von Strukturen. Strukturen, die zum Modul-Header gehören, beschreiben das Modul genauer (siehe Abbildung 5.9). Überarbeitungsstrukturen geben Aufschluss über den

historischen Verlauf des Moduls sowie über vorgenommene Änderungen. Definitionsstrukturen wiederum bilden den Hauptteil des Dokuments, in dem das Datenmodell definiert ist. Wie bei allen in YANG existierenden Strukturen ist die Arbeit mit Modulen einigen Einschränkungen unterworfen. Diese Einschränkungen werden hier aber nicht alle erwähnt, weil sie entweder sehr naheliegend sind oder zu sehr in die Tiefe gehen würden [12].

```
typedef percentage {  
    type uint8 {  
        range "0..100";  
    }  
}
```

Abbildung 5.10: Beispiel ‚typedef‘ aus Cisco-IOS-XE-types.yang

Eine ‚typedef‘-Struktur, wie in Abbildung 5.10 zu sehen ist, kann mehrere Strukturen enthalten. Der durch ‚typedef‘ neu definierte Datentyp muss einen eindeutigen und vor allem einzigartigen Namen erhalten. Der abgeleitete Datentyp kann danach mittels dieses Namens referenziert und verwendet werden. Innerhalb von ‚typedef‘ muss ein ‚type‘ folgen, das anzeigt, von welchem Datentyp abgeleitet wird. Innerhalb der ‚type‘-Struktur folgt meist noch eine Struktur, wie eine Muster- oder eine Reichweitenangabe, welche den Datentyp genauer spezifiziert. Die meisten der von Cisco definierten Datentypen werden innerhalb eines eigenen Moduls definiert und zur Verwendung in den jeweiligen Modulen importiert [12].

```
container enhanced {  
    description  
        "Enable enhanced history collection";  
    leaf interval {  
        description  
            "Aggregation interval";  
        type uint16 {  
            range "1..3600";  
        }  
    }  
    leaf buckets {  
        description  
            "Number of buckets to collect data";  
        type uint8 {  
            range "1..100";  
        }  
    }  
}
```

Abbildung 5.11: Beispiel ‚container‘ aus Cisco-IOS-XE-sla.yang

Eine ‚container‘-Struktur (siehe Abbildung 5.11) wird benutzt, um einen internen Datenknoten in den Schemabaum zu platzieren. Sie besteht immer aus einem

eindeutigen Namen und einem Block an Strukturen, der in dieser enthalten ist. Das bedeutet, dass eine ‚container‘-Struktur nie einen Wert enthalten kann, sondern auf ihre Rolle als Hülle beschränkt ist. Dabei gibt es nichtsdestotrotz zwei Arten von ‚container‘-Strukturen. Man unterscheidet ‚non-presence‘- und ‚presence‘-Container. Dabei hat der ‚non-presence‘-Container keine eigene Bedeutung, sondern existiert nur, um Kinderknoten zu enthalten. Diesem entgegengesetzt steht der ‚presence‘-Container, der über seine Funktion als Container hinaus eine Bedeutung erhält. Eine ‚container‘-Struktur kann, im Gegensatz zu den meisten anderen YANG-Strukturen, fast alle Arten von Strukturen enthalten [12].

```
leaf source-ip {  
    description  
        "Source Address";  
    type union {  
        type string;  
        type inet:ipv4-address;  
    }  
}
```

Abbildung 5.12: Beispiel ‚leaf‘ aus Cisco-IOS-XE-sla.yang

Im Kontrast zu Containern und anderen Strukturen, kann ein ‚leaf‘ keine weiteren Schemaknoten beziehungsweise Kinderknoten in sich aufnehmen und bildet folglich das Ende eines Baums. Ein ‚leaf‘-Statement, wie in Abbildung 5.12 zu sehen, hält außer einem identifizierenden Namen jedoch immer auch einige Strukturen, die zusätzliche Informationen enthalten. Also wird ein ‚leaf‘ ohne Ausnahme dazu benutzt, eine wertbehaftete Variable eines bestimmten Typs zu enthalten. Daraus muss man schlussfolgern, dass der Knoten entweder nicht oder maximal einmal als Instanz existieren kann [12].

```

choice http-choice {
  case http-get {
    container get {
      description
        "HTTP get operation";

      leaf url {
        description
          "URL";
        type string;
      }
      leaf source-ip {
        description
          "Source Address";
        type union {
          type inet:ip-address;
          type string;
        }
      }
      leaf source-port {
        description
          "Source port";
        type uint16 {
          range "1..65535";
        }
      }
      leaf name-server {
        description
          "Name server";
        type union {
          type string;
          type inet:ip-address;
        }
      }
    }
  }
}

```

Abbildung 5.13: Beispiel ‚choice‘ aus Cisco-IOS-XE-sla.yang

Wie die meisten Strukturen enthält eine ‚choice‘-Struktur einen eindeutigen Namen und einen Block an Folgestatements. Der Block von Folgestatements beinhaltet hierbei eine gewisse Anzahl von ‚case‘-Strukturen, die ihrerseits bestimmte Statements enthalten. Dabei kann zu einem Zeitpunkt immer nur ein Zweig der ‚choice‘-Struktur aktiv sein. Das heißt, dass immer nur ein ‚case‘-Statement innerhalb einer ‚choice‘-Struktur ausgewählt wird. Innerhalb des Datenbaums scheinen sowohl ‚choice‘- als auch ‚case‘-Statements nicht auf. In Abbildung 5.13 ist ein ‚case‘-Statement mit dem Namen ‚http-get‘ zu sehen, das eigene Substatements enthält und wiederum selbst in eine ‚choice‘-Struktur gekapselt ist.

Es können daher beliebig viele, von den bereits bestehenden ‚case‘-Optionen unterschiedliche, zusätzliche Wahloptionen innerhalb eines ‚choice‘ existieren [12].

```
grouping config-ip-sla-grouping {  
  container sla {  
    description  
      "IP Service Level Agreement";  
    list entry {  
      key "number";  
      leaf number {  
        type uint32;  
      }  
    }  
  }  
}
```

Abbildung 5.14: Beispiel ‚grouping‘ aus Cisco-IOS-XE-sla.yang

Das ‚grouping‘-Statement wird dazu verwendet einen Block mit einer unbestimmten Anzahl von Knoten wiederverwendbar zu machen. Eine Wiederverwendung kann lokal innerhalb eines Moduls, eines Submoduls und in einem anderen Modul, das von diesem lokalen Modul importiert, geschehen. Dabei ähnelt die Struktur einem ‚struct‘-Baustein aus verschiedenen Programmiersprachen. Der Aufbau ist analog zu dem einer ‚choice‘-Struktur, denn auch hier ist ein Name vergeben und das ‚grouping‘-Statement wird mit anderen Strukturen befüllt, die dem ‚grouping‘ inhaltliche Information geben [12].

```
augment "/ios:native/ios:ip" {  
  uses config-ip-sla-grouping;  
} //augment
```

Abbildung 5.15: Verwendung von ‚grouping‘ in Cisco-IOS-XE-sla.yang

Wenn ein ‚grouping‘, wie in Abbildung 5.14 zu sehen, definiert und befüllt wurde, kann es daraufhin sehr einfach mittels eines ‚uses‘-Statements referenziert werden. Diese Referenzierung des ‚grouping‘-Statements kann in Abbildung 5.15 eingesehen werden. Hierbei wird der gesamte Block, der innerhalb des ‚grouping‘-Statements definiert wurde, in das ‚augment‘-Statement aus Abbildung 5.15 eingefügt [12].

In Abbildung 5.15 findet sich die Verwendung eines ‚augment‘-Statements. Dabei erlaubt es einem Modul beziehungsweise Submodul die Erweiterung des bereits bestehenden Schemabaums in externen, lokalen Modulen beziehungsweise Submodulen. Abbildung 5.15 zeigt eine beispielhafte und oft genutzte Möglichkeit das ‚augment‘-Statement zu verwenden, um bereits existierende Module zu

erweitern. Dabei wird innerhalb des ‚augment‘ ein ‚grouping‘-Statement mittels eines ‚uses‘-Statements aufgerufen [12].

5.4.1.3. Werkzeuge

Da YANG bereits mehrere Jahre in der Praxis verwendet wird und eine immer größere Verbreitung findet, steigt gleichzeitig die Bereitschaft von privaten Entwicklern und Firmen, Werkzeuge für diese neue Entwicklung zu erstellen. Nachfolgend werden einige Werkzeuge vorgestellt, die Neulingen diese Technologie schnell und einfach näherbringen sollen und damit Firmen ermöglicht, neue Entwicklungen für diese Technologie schneller, einfacher und vor allem kostengünstiger zu realisieren.

pyang ist ein Validator, Wandler und Generator für YANG. Wie der Name des Programms bereits andeutet, ist die Software in Python geschrieben. Es sind jedoch auch XSLT- und sh/bash-Komponenten in der Software enthalten. pyang kann über Plug-Ins beliebig erweitert werden, wird kontinuierlich weiterentwickelt und durch eine Community auf GitHub betreut. Dabei wird zum Beispiel die Umwandlung eines YANG-Modells in ein anderes Format durch eine Sammlung von Plug-Ins bewerkstelligt. Bei diesen Umwandlungsarten ist jedoch zu beachten, dass die meisten CLI-Befehle nur sehr dürftig dokumentiert sind, und eine klare oder ausführliche Beschreibung meist nicht vorhanden ist. Ein gutes Beispiel für den Mangel an Dokumentation ist jene Programmfunktion, welche die Umwandlung von bereits vorhandenen SNMP MIBs zu YANG ermöglicht, um bereits von Unternehmen getätigte Investitionen bei einem Wechsel von SNMP auf YANG nicht überflüssig zu machen. Um einen Überblick über YANG-Module zu erlangen, ist eine Darstellung als DSDL oder als Hypertree, eine Struktur die mittels des Programms Treebolic graphisch visualisiert werden kann, möglich. Eine für Webanwendungen sehr nützliche Option ist die Darstellung eines Baums in HTML. Dieser Baum wird jedoch mithilfe von JavaScript interaktiv dargestellt und kann daher durchwandert werden. In Abbildung 5.16 ist eine solche Darstellung zu sehen [13].

| Element [+Expand all [-Collapse all] | Schema | Type | Flags | Opts | Status | Path |
|--------------------------------------|-----------|---------------------|-----------|------|---------|---|
| ietf-interfaces | module | | | | | |
| interfaces | container | | config | | current | /if:interfaces |
| interface[name] | list | | config | | current | /if:interfaces/if:interface |
| name | leaf | string | config | | current | /if:interfaces/if:interface/if:name |
| description | leaf | string | config | ? | current | /if:interfaces/if:interface/if:description |
| type | leaf | ianaif:iana-if-type | config | | current | /if:interfaces/if:interface/if:type |
| location | leaf | string | config | ? | current | /if:interfaces/if:interface/if:location |
| enabled | leaf | boolean | config | ? | current | /if:interfaces/if:interface/if:enabled |
| oper-status | leaf | enumeration | no config | ? | current | /if:interfaces/if:interface/if:oper-status |
| last-change | leaf | yang:date-and-time | no config | ? | current | /if:interfaces/if:interface/if:last-change |
| if-index | leaf | int32 | no config | ? | current | /if:interfaces/if:interface/if:index |
| link-up-down-trap-enable | leaf | enumeration | config | ? | current | /if:interfaces/if:interface/if:link-up-down-trap-enable |
| phys-address | leaf | yang:phys-address | no config | ? | current | /if:interfaces/if:interface/if:phys-address |
| higher-layer-if | leaf-list | interface-ref | no config | * | current | /if:interfaces/if:interface/if:higher-layer-if |
| lower-layer-if | leaf-list | interface-ref | no config | * | current | /if:interfaces/if:interface/if:lower-layer-if |
| speed | leaf | yang:gauge64 | no config | ? | current | /if:interfaces/if:interface/if:speed |
| statistics | container | | no config | | current | /if:interfaces/if:interface/if:statistics |

Abbildung 5.16: HTML-Darstellung einer YANG-Datei [13]

Soll die Entwicklung eines YANG-Schemas oder einer Anwendung, die YANG nutzt, über XML geschehen, bietet sich die Umwandlungsfunktion von YANG zu YIN an. YIN ist die XML-Repräsentation einer YANG-Datei. Auch ein in XML geschriebenes Modul kann mithilfe von pyang in eine YANG-Datei überführt werden. Die für dieses Anwendungsbeispiel wohl hilfreichste Repräsentation ist die Darstellung in Form eines Baums. Dabei stehen verschiedene Zusatzoptionen zur Verfügung. Die Option `tree-help` gibt eine kurze Erklärung zu den in der Baumstruktur verwendeten Symbolen, wobei auf diese in Abschnitt 5.4.1.4 noch genauer eingegangen wird. Auch lässt sich nur ein Teil des Baums ausgeben. Dies kann zum Beispiel über die Befehle `tree-depth depth` oder `tree-path path` geschehen. Bei Ersterem werden nur Knoten einer gewissen Tiefe ausgegeben, bei Zweitem hingegen werden alle Elemente unterhalb eines vorgegebenen Pfads angezeigt [13].

```
pyang -p /home/hoeflmai/pyang/modules/CiscoIOSXE1662
-f tree /root/yang/vendor/cisco/xe/1662/Cisco-IOS-XE-sla.yang -o /home/hoeflmai/sla.tree
```

Abbildung 5.17: Beispielhafte Verwendung des pyang-Tools

In Abbildung 5.17 ist ein Befehl zu sehen, mit dessen Hilfe pyang die Datei `Cisco-IOS-XE-sla.yang` in eine Baumansicht überführt und diese danach in der Datei `sla.tree` als Text ausgibt. Die Option `-p` hilft alle möglichen Abhängigkeiten der YANG-Datei von anderen YANG-Modulen aufzulösen, um so bei der Umwandlung keine Fehler zu erzeugen. Ähnliche Funktionen bietet das Github-Projekt goyang, welches in der Programmiersprache Go geschrieben ist, jedoch einen geringeren Funktionsumfang als pyang aufweist [13].

Ein weiteres nützliches Werkzeug, das bei Entwicklungsprozessen eingesetzt werden kann, ist der YangExplorer. Der YangExplorer ist eine von Cisco

entwickelte und unter der Apache Lizenz veröffentlichte Software, die es dem Nutzer ermöglicht einen genaueren Blick auf YANG-Module und deren Datenstruktur zu werfen. Bei Verwendung von NETCONF kann dieses Programm auch dazu genutzt werden Remote Procedure Calls auszulösen. Allerdings ist diese Software noch immer in einem Beta-Status und wird kontinuierlich weiterentwickelt. Leider ist die Unterstützung von RESTCONF noch nicht vollständig gegeben und daher sind auch die Testfälle, die für die Evaluierung dieses Tools durchgeführt wurden, nicht geglückt. Auch müssen dem Benutzer die Abhängigkeiten zwischen einzelnen YANG-Modulen bereits im Vorhinein bekannt sein. Sollte der Benutzer dies außer Acht lassen und eine hochzuladende Datei eine Abhängigkeit zu einer noch nicht hochgeladenen Datei aufweisen, wird beim Hochladen eine Fehlermeldung ausgegeben und der Vorgang abgebrochen. Diese Gegebenheiten führen dazu, dass dieses Tool für die Anforderungen dieses Anwendungsbeispiels nicht geeignet ist [14].

5.4.1.4. Baum-Struktur

```

module: Cisco-IOS-XE-sla
augment /ios:native/ios:ip:
  +--rw sla
    +--rw entry* [number]
      | +--rw number          uint32
      | +--rw (sla-param)?
      |   +--:(icmp-echo-case)
      |     | +--rw icmp-echo
      |       | +--rw destination?      union
      |       | +--rw source-interface?  string
      |       | +--rw source-ip?        union
      |       | +--rw data-pattern?      string
      |       | +--rw frequency?        uint32
      |       | +--rw history
      |       | | +--rw buckets-kept?    uint8
      |       | | +--rw distributions-of-statistics-kept?  uint8
      |       | | +--rw enhanced
      |       | | | +--rw interval?     uint32
      |       | | | +--rw buckets?     uint8
      |       | | +--rw filter?         enumeration
      |       | | +--rw hours-of-statistics-kept?  uint8
      |       | | +--rw lives-kept?    uint8
      |       | | +--rw statistics-distribution-interval?  uint8
      |       | +--rw owner?           string
      |       | +--rw request-data-size?  uint32
      |       | +--rw tag?             string
      |       | +--rw threshold?       uint32
      |       | +--rw timeout?         uint64
      |       | +--rw tos?             uint8
      |       | +--rw verify-data?     empty
      |       | +--rw vrf?             string

```

Abbildung 5.18: Auszug aus der Baumdarstellung für Cisco-IOS-XE-sla.yang

Wie bereits in Abschnitt 5.4.1.3 erwähnt, kann eine YANG-Datei mittels pyang in eine Baumstruktur überführt werden. Für diese Art der Darstellung war nicht von Anfang an ein Standard gegeben, sondern wurde, wie so oft in der Technik, erst nach einer weiteren Verbreitung und Verwendung definiert. Leider gilt hier, wie auch bei pyang, dass der jetzt existierende Standard nicht sehr genau ist. Dies ist darauf zurückzuführen, dass die Verbindung zwischen diesem Standard und dem YANG-Standard nicht im Vordergrund dieses Baumstruktur-Standards steht. Vielmehr werden in diesem Standard die Strukturen nur isoliert betrachtet und ohne den Kontext zum YANG-Standard erklärt. Das führt dazu, dass man den Typ eines Bauelements zum Teil nur noch aus seinem Kontext erkennen kann. In der Folge ist ein vollkommenes Verstehen der Baumdarstellung, ohne vorher die korrespondierende YANG-Datei durchgesehen zu haben, nicht mehr auf den ersten Blick möglich. Daher wird in diesem Abschnitt versucht, die vorhandenen Strukturen anhand des Beispielbaums aus Abbildung 5.18 zu erklären. In Abbildung 5.18 sieht man einen Ausschnitt der Baumdarstellung, die sich bei der Überführung der Datei ‚Cisco-IOX-XE-sla.yang‘ mittels pyang in eine Baumdarstellung ergibt. In Zeile eins sieht man den Namen des Moduls. Darunter ist die Angabe zu sehen, in welchem Modul beziehungsweise an welchen Platz innerhalb der gesamten Hierarchie der Inhalt der ‚augment‘-Struktur platziert wird. Das führt dazu, dass der Inhalt aus Zeile drei den absoluten Pfad ‚native/ip/sla‘ besitzt. Die in Zeile zwei enthaltenen Präfixe werden dabei nicht benötigt. In Zeile vier findet sich eine in einen Container gekapselte ‚list‘-Struktur, die als Schlüssel den Identifikator des darunterliegenden Elements besitzt. Der Schlüssel wird in eckigen Klammern angegeben. Zeile fünf zeigt eine normale Leaf-Struktur. Dies ist daran zu erkennen, dass der Knoten keine Kinder, keine speziellen Kennzeichnungen und einen Datentyp besitzt. In den Zeilen sechs und sieben findet man die Choice-Case-Struktur, die bereits in Abschnitt 5.4.1.2 erklärt wurde. Wie aus Zeile acht von Abbildung 5.18 ersichtlich, wird die gesamte Case-Option in einen Container gekapselt. Die Zeilen neun bis dreizehn zeigen eine Sammlung von Leaf-Elementen. Diese Sammlung wird von einem Container in Zeile vierzehn gefolgt [15].

```
<status>--<flags> <name><opts> <type> <if-features>
```

Abbildung 5.19: Generelle Struktur eines Knotens

In Abbildung 5.19 ist der standardisierte Aufbau eines Knotens zu sehen. Die meisten Bestandteile dieses Aufbaus sind für die Mehrheit der Elemente des Baums von essentieller Bedeutung. Um die Tiefe des Knotens in der Baumstruktur darzustellen und die Verschachtelung der Elemente besser visualisieren zu können, wird jedes Element von keinem oder mehreren Leerzeichen angeführt. Danach folgt eines von drei Zeichen: Ein ‚+‘ steht dafür, dass das Element aktuell ist und der neuesten Version der YANG-Datei entspricht. Die Zeichen ‚x‘ und ‚o‘ stellen die anderen Optionen dar, die das Status-Flag aufweisen kann. Dabei zeigen diese Optionen, dass das Element obsolet ist. Nach dem Status-Flag finden sich immer zwei Trennzeichen in Form eines ‚-‘. Darauf folgt ein Flags-Flag, das meist nur zwei von insgesamt sieben möglichen Optionen einnimmt. Diese zwei Optionen sind ‚rw‘ und ‚ro‘, wobei ‚rw‘ anzeigt, dass das Element sowohl geschrieben als auch gelesen werden kann. Hingegen zeigt ‚ro‘ an, dass das Element nur ausgelesen werden kann. Daraufhin folgt das Name-Flag, in dem meist nur der vollständige Name enthalten ist und daher keine Rückschlüsse über den Elementtyp des Elements zulässt. Für Choice- und Case-Elemente gibt es jedoch eigene Namensfelder, die es ermöglichen, den Elementtyp des Elements präzise zu bestimmen. Hierbei ist der Name des jeweiligen Elements bei beiden Elementtypen in eine Klammer gekapselt. Der Unterschied zwischen den Elementtypen lässt sich dann daran festmachen, dass der Klammer eines Case-Knoten immer ein Doppelpunkt vorangestellt ist. Wurde ein Element aus einem anderen Modul augmentiert, ist dem Namen des Elements auch ein Präfix, das vom Namen des Elements durch einen zusätzlichen Doppelpunkt getrennt ist, vorangestellt. Das Opts-Flag bietet wiederum sechs Optionen, welche oftmals helfen den Elementtyp eines Elements einzugrenzen. Ein Fragezeichen gibt an, ob die Verwendung eines Leaf-, Choice-, Anydata- oder Anyxml-Elements optional ist. Ein Ausrufezeichen kennzeichnet ein Element des Typs ‚presencecontainer‘. Die Verwendung eines ‚*-Zeichens kommt dagegen nur bei Elementen der Typen ‚leaf-list‘ und ‚list‘ vor. Die bereits erklärte Verwendung von Keys fällt ebenso unter das Opts-Flag. Die beiden anderen Optionen in diesem Bereich, ein ‚@‘ oder die Verwendung eines ‚/‘, kommen dagegen nur sehr vereinzelt vor und werden daher nicht weiter behandelt. Das Type-Flag kommt hingegen nur bei Leaf- und Leaf-list-Elementen vor und beschreibt den Datentyp des Elements. Hierbei gibt es wiederum einige Spezialfälle, die nicht erwähnt werden. Dies betrifft auch das If-

feature-Flag, welches ebenfalls in Abbildung 5.19 dargestellt wird. Dieses tritt jedoch nicht sehr häufig auf [15].

5.4.2. RESTCONF

RESTCONF ist ein Protokoll, welches ermöglicht Daten zugänglich zu machen, die entweder in YANG oder deren Datenspeicher über NETCONF definiert werden. RESTCONF kann als Kind einer Entwicklung gesehen werden, in deren Verlauf immer mehr Services und Applikationen eine eigene REST-Programmierschnittstelle zur Verfügung stellen. Daher wuchs auch im Umfeld von SDN der Wunsch heran, die Möglichkeiten von APIs nutzen zu können. Aus diesem Grund entschied sich die IETF dazu NETCONF zu RESTCONF zu erweitern und nicht ein komplett neues Protokoll und Datenmodell zu entwickeln. Die Annahme, dass RESTCONF der Nachfolger von NETCONF ist, entbehrt jeder Grundlage, weil die Anforderungen an beide Protokolle sehr unterschiedlich sind. Ein Nachteil von RESTCONF gegenüber NETCONF ist, dass die beiden Protokolle momentan von ihrem Umfang her nicht deckungsgleich sind. NETCONF bietet wesentlich weniger inhaltliche Lücken. RESTCONF bietet wiederum den Vorteil, dass die Kommunikation über HTTP und nicht mehr über SSH läuft. Der für dieses Protokoll verwendete Port ist Nummer 8080. Wie bei anderen REST APIs auch, wird bei RESTCONF die Authentifizierung über den HTTP-Header abgewickelt. Überdies werden typische HTTP CRUD-Operationen durch POST-, GET-, PUT-, PATCH- und DELETE-Befehle implementiert. Die Datenübermittlung läuft meist entweder über XML oder JSON [16], [17].

5.5. API-Abfragen

In diesem Abschnitt werden einige Grundlagen zur Infrastruktur API vorgestellt. Um ein besseres Verständnis zu fördern und sich besser in den Strukturen der API zurechtzufinden, werden noch einige Beispiele angeführt. Ein Verständnis der YANG-Modelle ist essentiell, weil es bei den verwendeten Infrastruktur-APIs aus dem Hause Cisco meist keine Dokumentation gibt und die YANG Dateien selbst die Dokumentation darstellen. Bei der Verwendung der API ist die Konstruktion der URI eine der Herausforderungen. In Abbildung 5.20 findet sich die generalisierte Regel dazu [17].

`https://<ADDRESS>/<ROOT>/<DATA STORE>/<[YANG MODULE:]CONTAINER>/<LEAF>[?<OPTIONS>]`

Abbildung 5.20: Konstruktionsschema der URI

Einige Bestandteile dieses Schemas beziehungsweise dieser Regel werden daher hier erklärt. Die Adresse gibt an, unter welcher IP-Adresse der RESTCONF-Agent zu finden ist. Der Abschnitt ‚ROOT‘ gibt den Startpunkt der RESTCONF-Anfrage an. In den meisten Fällen ist dies bei den verwendeten APIs die Zeichenkette ‚restconf‘. Das Feld ‚DATA STORE‘ gibt an, welcher Datenspeicher genutzt werden sollte. Danach folgt der Name des verwendeten YANG-Moduls, eine optionale Folge von Containern und die optionale Verwendung eines abschließenden Leaf-Elements. Um diese Struktur herausbilden zu können, müssen die verwendeten YANG-Module und deren Zusammenhänge verstanden werden. Dies kann am einfachsten mit der Baumdarstellung der YANG-Datei geschehen. Ist diese Voraussetzung erfüllt, kann ein Tool wie cURL oder Google Postman dazu benutzt werden, die eruierte URI zu testen [17].

```
curl -s -k -H "Accept-Encoding: gzip, deflate" -H "Accept:
application/yang-data+json, application/yang-data.errors+json" -u
[REDACTED] 'https://[REDACTED]/restconf/data/Cisco-IOS-XE-
native:native/ip/Cisco-IOS-XE-sla:sla/entry=1'
```

Abbildung 5.21: Auslesen von Daten mittels API

In Abbildung 5.21 ist ein Testfall für die Überprüfung der URI ‚https://IP:PORT/restconf/data/Cisco-IOS-XE-native:native/ip/Cisco-IOS-XE-sla:sla/entry=1‘ zu sehen. Dieses Testbeispiel kann mittels der Abbildung 5.18 nachvollzogen werden. Hierbei ist darauf hinzuweisen, dass die URI auch bei einer verkürzten Schreibweise, bei der die Präfixe von Modulen weggelassen werden, seine Gültigkeit behält. Diese Variante würde wie folgt aussehen: ‚https://IP:PORT/restconf/data/native/ip/sla/entry=1‘.

```
{
  "Cisco-IOS-XE-sla:entry": {
    "number": 1,
    "icmp-echo": {
      "destination": "[REDACTED]",
      "frequency": 5
    }
  }
}
```

Abbildung 5.22: Ergebnis der Testabfrage

Die URI setzt sich aus einem Standardpfad, dem Pfad der innerhalb der ‚augment‘-Struktur angegeben ist, dem ersten Container sowie dem nachgeordneten List-Element, welches über seinen Schlüssel identifiziert wird, zusammen. Das Ergebnis zeigt den Inhalt, den diese ‚entry‘-Instanz am Agent annimmt, und ist in Abbildung 5.22 zu sehen. Es handelt sich dabei um ein ICMP-echo, das mittels IP SLA konfiguriert wurde.

```
curl -s -k -X POST -H "Content-Type: application/yang-data+json" -H
"Accept-Encoding: gzip, deflate" -H "Accept: application/yang-data+json,
application/yang-data.errors+json" -u [REDACTED]
'https://[REDACTED]/restconf/data/Cisco-IOS-XE-
native:native/ip/Cisco-IOS-XE-sla:sla/' -d '{
  "Cisco-IOS-XE-sla:entry": {
    "number": 2,
    "icmp-echo": {
      "destination": "[REDACTED]",
      "frequency": 5
    }
  }
}'
```

Abbildung 5.23: Eingabe von Daten mittels API

In Abbildung 5.23 ist das Einbringen einer neuen IP SLA-Konfiguration auf die Netzwerkkomponente zu sehen. Dabei muss die URI aus dem letzten Beispiel zuerst so modifiziert werden, dass diese an der ‚richtigen Stelle‘ im Baum eingefügt wird. Die zu sendenden Daten werden bei cURL mit der Option ‚-d‘ übermittelt und können aus den Ergebnissen der letzten Abfrage rekonstruiert werden. Wird eine erneute Abfrage auf die URI ‚https://IP:PORT/restconf/data/native/ip/sla/entry=2‘ ausgeführt, wird der soeben erzeugte Eintrag in JSON-Format ausgegeben.

5.6. IP SLA

Da innerhalb dieses Anwendungsbeispiels vor allem IP SLA behandelt und daher auch für Beispiele verwendet wird, sollte diese Technik auch noch überblicksmäßig erwähnt werden.

IP SLA ist ein Feature des durch Cisco verwendeten Betriebssystems ‚Internetwork Operating System‘. IP SLA erlaubt Netzwerkadministratoren die Leistung von Netzwerkkomponenten und deren Verbindungen durch die Bereitstellung von Leistungsmerkmalen zu überwachen. Diese Leistungsmerkmale sind zum Beispiel Antwortzeit, Pfad, Latenz, Jitter oder Paketverlust. Diese erlauben wiederum den Quality of Service von Anwendungen zu überprüfen und,

wenn nötig, Schritte zu unternehmen, um ein Service Level Agreement zu erfüllen. Die Daten, die durch IP SLA-Aktionen generiert werden, eignen sich hervorragend, um bereits bestehende Netzwerkmanagement- und Überwachungsplattformen, wie zum Beispiel Nagios, mit zusätzlichen Daten zu versorgen. Dies dient im Endeffekt dazu, ein ganzheitliches Lagebild des Netzwerkes zu erhalten und so das Incidentmanagement zu verbessern [18].

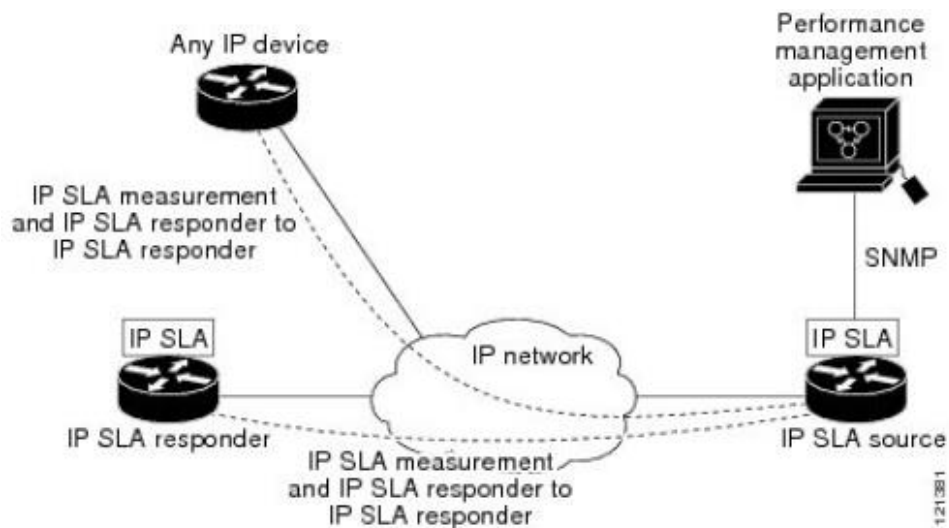


Abbildung 5.24: Beispielanwendung von IP SLA [18]

In Abbildung 5.24 ist ein Anwendungsbeispiel von IP SLA zu sehen. Hier wird die Kommunikation im WAN überprüft, weil oftmals zwischen einzelnen Standorten (Branch) die Überwachung von SLA besonderen Stellenwert besitzt. Die einfachste Art von IP SLA-Messung ist, nur einen IP SLA-Typ und dessen Parameter zu konfigurieren. Bei komplexeren IP SLA-Abfragen kann zusätzlich noch ein IP SLA Responder und ein vordefinierter Schwellenwert konfiguriert werden. Nachdem eine Operation gestartet wurde, muss dem System ausreichend Zeit gegeben werden, Daten zu sammeln. Dies ist notwendig, um am Ende der Operation genügend nützliche Daten zu erhalten. Danach ist das Auslesen dieser Daten entweder über die CLI oder über SNMP möglich.

5.7. PHP

Wie bereits erwähnt, sollte die Erstellung eines PHP-Prototyps fokussiert werden, um dadurch die Machbarkeit sowie die etwaige spätere Integration in ein Managementtool zu beschleunigen und ermöglichen zu können. In Abbildung 5.28 kann die Palette an fertiggestellten Operationen mitsamt ihren Beschreibungen

eingesehen werden. Durch diese Operationen kann das Einfügen und Auslesen von Informationen bereits effizient geschehen. Im folgenden Anwendungsfall werden alle Stärken des Programms effektiv genutzt.

```
php kbc_ip_sla_conf.php --apicem-server [REDACTED] --apicem-user [REDACTED]
--apicem-password [REDACTED] -A
```

Abbildung 5.25: Auflistung aller Netzwerkkomponenten

```
php kbc_ip_sla_conf.php -u [REDACTED] -p [REDACTED] -s [REDACTED] -d
[REDACTED] -L
```

Abbildung 5.26: Auflistung aller konfigurierten IP SLA-Operationen

```
php kbc_ip_sla_conf.php -u [REDACTED] -p [REDACTED] -s [REDACTED] -d
[REDACTED]
```

Abbildung 5.27: Konfiguration einer IP SLA-Operation

Im ersten Schritt, zu sehen in Abbildung 5.25, verfügt der Nutzer nur über die Authentifizierungspasswörter und Benutzernamen des APIC-EM und der zu konfigurierenden Netzwerkkomponente. Außerdem ist dem Benutzer die IP-Adresse des APIC-EM bekannt. Es muss Topologieinformation aus dem Controller, dem APIC-EM, ausgelesen werden, um an weitere Informationen zu gelangen. Das Ergebnis dieser Abfrage gibt die IP-Adressen der Netzwerkkomponenten an, die mittels der nächsten Anfragen angesprochen werden können. Dadurch ist es im nächsten Schritt möglich, alle auf dem gewünschten Gerät vorhandenen IP SLA-Operationen durch den Befehl in Abbildung 5.26 auflisten und ausgeben zu lassen. Danach kann eine neue IP SLA-Operation eingebracht werden, wobei, bei dem in Abbildung 5.27 abgesetzten Befehl, eine standardisierte Konfiguration eingespielt wird, bei der nur eine Zieladresse angegeben werden muss.

```
[root@kmp-demo01 cisco_dna_ena_poc]# php kbc_ip_sla_conf.php -h
kbc_ip_sla_conf.php

IP SLA config via APIC-EM API and RESTCONF API
```

```
-A ... list all APIC-EM devices
--apicem-server <apicem_server> ... Cisco APIC-EM server IP address
--apicem-user <apicem_user> ... Cisco APIC-EM username
--apicem-password <apicem_password> ... Cisco APIC-EM password
-u <restconf_username> ... username
-p <restconf_password> ... password
-L ... list all IP SLA entries on source
-s <IP address> ... source IP address
-d <IP address> ... destination IP address
-D <entry_umber> ... delete IP SLA entries number
-v ... debug verbose
-S ... list IP SLA stats on source device
```

Abbildung 5.28: Verfügbare Befehle im erstellten Prototyp

Der finale Quellcode des Prototyps weist eine Länge von circa 600 Zeilen und 22000 Zeichen auf. Aufgrund dieses Umfangs und der Tatsache, dass der Quellcode geistiges Eigentum eines privatwirtschaftlichen Unternehmens ist und das Unternehmen daher kein Interesse an einer Veröffentlichung des Quellcodes hat, wird darauf verzichtet, Ausschnitte aus diesem zu zeigen.


5.8. DynamicConfigurationWizard

Durch die Flexibilität von YANG ergeben sich viele Anwendungsmöglichkeiten. Eine Idee ist die Konfiguration einer Netzwerkkomponente gänzlich über ein selbstentwickeltes Tool ablaufen zu lassen. Die Umsetzung dieses Tools wird in der vorliegenden Arbeit als Machbarkeitsstudie bewertet. Als Vorbild dient dabei

das Github-Projekt ‚YangMan‘, das sich optisch und funktional sehr an Google Postman anlehnt. Es wurde eine Weboberfläche entwickelt, welche mit JavaScript interagirbar gemacht wurde, ein PHP-Backend aufweist, die beschriebene Funktionalität liefert und sich optisch an Google Postman orientiert. Jedoch ist anzumerken, dass man bei der Umsetzung des ‚YangMan‘-Konzepts, ohne Wissen über YANG und die Strukturen von YANG, das Tool nie effektiv bedienen könnte. Der Grund dafür ist, dass durch ein Programm, wie ‚YangMan‘, die generelle Struktur von YANG-Modulen nicht vollständig abstrahiert werden kann.

Bei der Entwicklung des DynamicConfigurationWizard gab es mehrere Herangehensweisen. Eine Variante wäre die Verwendung von YDK auf einer C++- oder Python-Plattform. Dies war nicht möglich, weil dies der Anforderung einer Weboberfläche widersprach. Eine weitere Möglichkeit wäre die Verwendung von pyang beziehungsweise goyang, oder Änderungen an deren Code, um eine bessere Aufbereitung und Repräsentation der YANG-Module zu erreichen. Dies wäre sehr zeitintensiv gewesen, weil hierbei mehrere verschiedene Plattformen verwendet werden müssten, neue Sprachen erlernt und erst die bestehenden professionell strukturierten Programme pyang und goyang analysiert werden müssten. Der daher schlussendlich verfolgte Lösungsansatz zur Umsetzung war, die Komplexität der Verbindungen zwischen YANG-Dateien mittels pyang aufzulösen und in eine Baumstruktur zu überführen. Mithilfe dieser Baumstruktur sollten dann verschiedenste Informationen ausgelesen werden. Die Komponente, die für die Bereitstellung von Informationen aus der Baumstruktur verantwortlich ist, gliedert jedes Bauelement in zehn unterschiedliche Kategorien. Der Name, die Art beziehungsweise der Typ, das status-Flag, die Tiefe des Knotens im Baum, das letzte Element auf dem das aktuelle Bauelement basiert, das flags-Flag, der eventuell vorhandene Schlüssel des Elements, die Notwendigkeit des Elements, der Datentyp und eine Struktur, die alle Kinder des Knotens enthält, werden aufgezeichnet und in eine separate JSON-Datei gespeichert. Mit diesen Informationen wird ermöglicht, Dateien in einem Webinterface einzulesen und damit den Baum graphisch zu durchwandern und editieren. Aus diesen Änderungen in der graphischen Repräsentation der YANG-Datei werden daraufhin bei einer POST- oder PUT-Operation eine URL und ein Text im JSON-Format generiert, die dann an das Zielgerät gesendet werden. Bei GET- und DELETE-

Operationen verhält es sich ähnlich. Jedoch muss die URL manuell eingegeben werden, um eine Nachricht erfolgreich senden zu können. Die Benutzeroberfläche, die es ermöglicht diese Daten einzugeben, findet sich in Abbildung 5.29.



Home

Write to network device

Device-username

Device-password

Read json-file

slamitcase.json

Path

[Imprint](#) [Kapsch Group](#) [Terms of Use](#)

Abbildung 5.29: Eingabemaske für Benutzerdaten

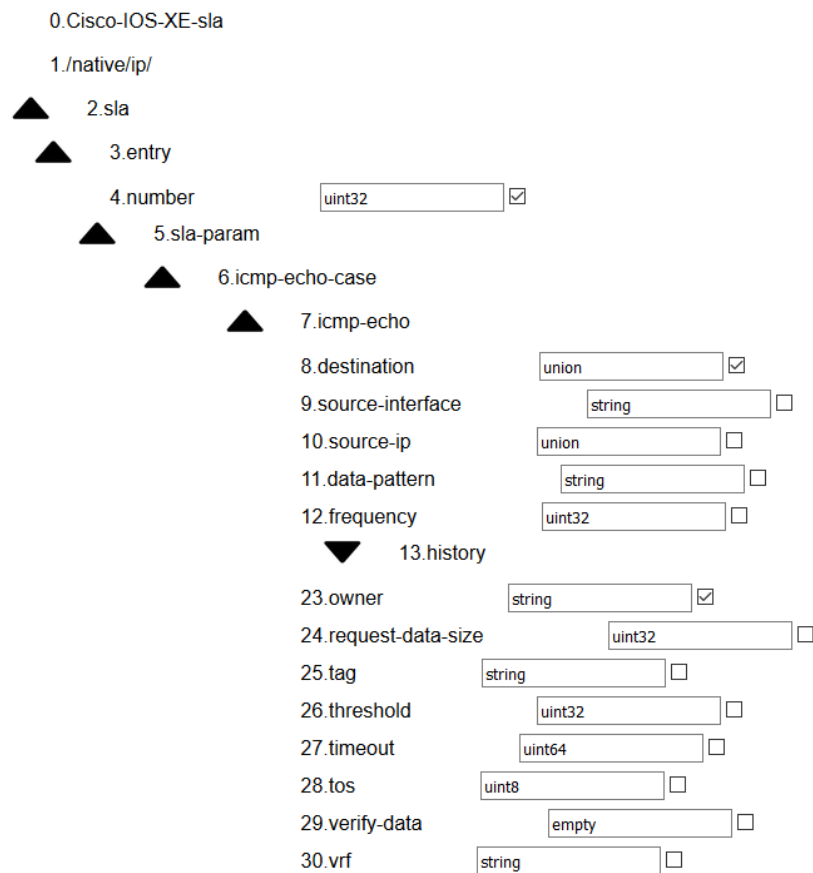


Abbildung 5.30: Strukturierte Darstellung eines Moduls innerhalb der Weboberfläche

In Abbildung 5.30 ist die Repräsentation der Datei ‚Cisco-IOS-XE-sla.yang‘ zu sehen, welche mit der Baumstruktur aus Abbildung 5.18 gleichbedeutend ist. Die verschiedenen Ebenen der Datei können durch einen Klick auf das jeweilige Zeichen, links neben dem Element, angezeigt oder verborgen werden. Wird vom Benutzer zum Beispiel die Option POST ausgewählt, kann durch das aktivieren einer der Checkboxes das Auslesen der Daten des daneben befindlichen Textfelds geschehen. Um daraufhin aus diesen Daten einen JSON-Text zu erstellen und diesen später noch durch den Benutzer überprüfen zu können, wird durch den Nutzer das Interaktionselement mit der Aufschrift ‚Generate JSON‘ genutzt. Daraufhin erscheint, wie in Abbildung 5.31 zu sehen ist, in einem Bereich der Oberfläche der JSON-Text.

Generated JSON:

```
{
  "entry": {
    "number": "uint32",
    "icmp-echo": {
      "destination": "union",
      "owner": "string"
    }
  }
}
```

Abbildung 5.31: Beispiel von generiertem JSON

Erscheint dem Nutzer der JSON-Text korrekt, kann dieser an der Netzwerkkomponente die Konfigurationsaktion durch das Klicken auf ‚Submit‘ auslösen. Sobald eine Antwort von der angesprochenen Netzwerkkomponente verfügbar ist, wird diese darunter in einem eigenen Feld dargestellt, wobei dabei auch der HTTP-Statuscode der Rückmeldung angezeigt wird.

6. Programmierschnittstellen – Anwendungsbeispiel 2

In diesem Kapitel wird das zweite Anwendungsbeispiel vorgestellt, das sich mit Cisco DNA und deren Programmierschnittstellen beschäftigt.

6.1. Problemstellung und Beschreibung

Das zweite Anwendungsbeispiel verwendet die Programmierschnittstellen der Cisco Identity Services Engine, welche eine weitere DNA-Komponente darstellt, in einer neuen Applikation. Dabei soll diese Applikation ermöglichen, MAC-Adressen und einige andere Parameter in eine Cisco Identity Services Engine einspielen zu können. Wie bereits in einem vorangegangenen Kapitel beschrieben, können durch die Cisco ISE verschiedene Authentifizierungsverfahren umgesetzt werden. Eines der verfügbaren Authentifizierungsverfahren ist MAB. MAB kann zum Beispiel eingesetzt werden, falls keine andere Authentifizierungsart möglich ist. Unterstützt ein Gerät den Standard 802.1X nicht, wird dieses Verfahren, wie bereits beschrieben, oftmals eingesetzt. Dies könnte zum Beispiel bei einem einfachen Drucker der Fall sein. Schlägt die Authentifizierung eines Netzwerkgeräts fehl oder ist diese nicht möglich, wird der Zugang dieses Geräts zum Netzwerk meist gesperrt. Um daher das Einbringen einer größeren Anzahl von Geräten in ein Netzwerk zu ermöglichen und dabei nicht alle Einträge für jedes Gerät auf aufwändige Art und Weise manuell tätigen zu müssen, sollte dieser Vorgang durch ein Programm unterstützt werden.

6.2. Lösungsansatz

Die beschriebene Problemstellung lässt sehr viel Spielraum, um eine adäquate Lösung zu finden. Die Implementierung mittels PHP zeigt sich hierbei als die einfachste und flexibelste Variante. Ein Vorteil ist, dass die Entwicklung des Programms mehrstufig auf einem PHP-Kern erfolgen kann. In jedem Entwicklungsschritt kann bei einem solchen Vorgehen immer weiter Funktionalität hinzugefügt werden, ohne die Entwicklung neu starten zu müssen. Eine Idee, die auch weiterentwickelt wurde, ist das Löschen von durch das Tool selbst eingebrachten MAC-Adressen nach einer vordefinierten Zeitspanne zu ermöglichen. Der PHP-Kern wurde mithilfe von cURL realisiert. Die beiden anderen entstandenen Varianten des Tools wurden vor allem durch HTML, JavaScript, einen SQL-Server und PHP umgesetzt. Auf die verschiedenen

Versionen, ihre jeweiligen Eigenschaften und Umsetzungen wird in Abschnitt 6.4 eingegangen.

6.3. Generelle Funktionsweise

Wie bereits angedeutet, wird bei jeder der drei Programmvarianten dem PHP-Backend eine zentrale Rolle zuteil. Diesem werden die zu übermittelnden Daten übergeben und von dort durch eine Abfolge von Abfragen auf die ISE übertragen. Um eine erfolgreiche Gesamtanfrage zu tätigen, müssen pro MAC-Adresse meist vier Teilabfragen an die ISE erfolgen. Die Daten, die für eine Gesamtanfrage benötigt werden, bestehen dabei aus vier Feldern: der zu verwendeten MAC-Adresse, einem Beschreibungstext, dem Namen eines Profils und dem Namen einer Gruppe. Die erste Teilabfrage ermittelt, ob die gegebene MAC-Adresse bereits als Endpunkt in der ISE enthalten ist. Die zweite Abfrage sucht die Identifikationsnummer zum Namen des Profils. Die gleiche Funktion erfüllt die dritte Abfrage für den Namen der Gruppe. Für die dritte Abfrage benötigt die ISE für die Gruppen- und Profileinträge nicht den Namen, sondern die Identifikationsnummern dieser Einträge. Die vierte und finale Abfrage fügt schließlich den Datensatz in die ISE ein.

6.4. MacCurator

In diesem Abschnitt werden genauere Informationen über die verschiedenen Versionen des MacCurators bereitgestellt und deren Funktionsweise erklärt.

6.4.1. Konsolen-Version

`<MAC>;<Beschreibung>;<Endpoint Policy>;<Endpoint Identity Group>`

Abbildung 6.1: Struktur der Daten in einer CSV-Datei

Die Konsolenversion des MacCurator stützt sich auf das PHP-Grundprogramm und wird mittels einiger Parameter ausgeführt. Ein Aufruf mit allen konfigurierbaren Parametern findet sich in Abbildung 6.2. In diesem Beispiel wird in einer Windowsumgebung der Befehl in ein Commandline-Tool eingegeben, wobei die erste Zeile den absoluten Pfad der Datei angibt. In den Zeilen drei bis sieben werden alle erforderlichen Parameter angegeben. Der Parameter ‚tupel‘ ist eine Kombination aus der ISE-IP-Adresse und des Zielports. Die folgenden zwei Zeilen dienen der Authentifizierung des ISE-API-Nutzers an der ISE. Der

Parameter ‚portaluser‘ gibt den Namen an, der innerhalb der ISE für den jeweiligen getätigten Eintrag aufscheint. Die letzte Zeile enthält den absoluten Pfad zu der verwendeten CSV-Datei, welche die Daten der zu tätigenen Einträge enthält. Die Struktur einer solchen CSV-Datei kann man in Abbildung 6.1 sehen. Leider ist die API der verwendeten ISE nicht sehr performant, was dazu führt, dass pro MAC-Adresseintrag im Durchschnitt beinahe zwei Sekunden benötigt werden. Aufgrund der limitierten Verfügbarkeit von unterschiedlichen Testsystemen konnte für diese Arbeit nicht nach den Gründen dieser Leistungseinschränkung geforscht werden.

```
php.exe -f "C:\Installed_programms\XAMPP\htdocs\Own_Programm\consoleversion.php"  
-- -P  
--tuple 00.00.00.00:9060  
--password hoeflmaier  
--username hoeflmaier  
--portaluser hoeflmaier  
--csv-url C:\Installed_programms\XAMPP\htdocs\Own_Programm\test.csv
```

Abbildung 6.2: Beispielhafter Aufruf des Programms

In Abbildung 6.3 findet sich ein Auszug aus dem Programmcode des fertigen Produkts. Darin ist das eigentliche Schreiben der Daten auf die Cisco Identity Services Engine zu sehen. Es ist zu erkennen, dass für den HTTP-Request cURL verwendet wird. Dies stellt unter PHP die einfachste und flexibelste Möglichkeit dar, einen HTTP-Request abzusetzen. Generell finden sich bei dieser Version alle Ressourcen, welche die Verarbeitung der bereitgestellten Daten bewerkstelligen, innerhalb einer PHP-Datei.

```

function EndpointCreationApiCall($authString, $body, $stupel) {

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, "https://".$stupel."/ers/config/endpoint");
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
    curl_setopt($ch, CURLOPT_POSTFIELDS, $body);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        'Content-Type: application/json',
        'Accept: application/json',
        'Accept-Encoding: gzip, deflate',
    ));

    $response = json_decode(curl_exec($ch), true);
    $httpcode = curl_getinfo($ch, CURLINFO_HTTP_CODE);
    curl_close($ch);

    if (substr($httpcode,0,1)!="2"){
        echo"Error occured:\nThere was a problem with the EndpointCreation-API-Call!\n";
        exit;
    }
    else{
        echo"Endpoint-creation successful\n";
    }
}


```

Abbildung 6.3: Auszug aus der PHP-Version

6.4.2. Webinterface-Version

Im Vergleich zur Konsolen-Version bietet die Webinterface-Version einige Verbesserungen und Erweiterungen, auf die in diesem Abschnitt genauer eingegangen wird. Dabei werden die in Abbildung 6.4 zu sehenden Tabs der Webseite mit ihren Funktionen und den dahinterliegenden Mechanismen vorgestellt.

Im Tab ‚Home‘ befindet sich die integrierte Beschreibung des Webservices. Darin werden verschiedene Schritte erklärt, die nötig sind, um dieses Tool effektiv nutzen zu können. Im Tab ‚Add server‘, siehe Abbildung 6.4, findet sich die Funktion, eine ISE hinzuzufügen. Dabei werden pro ISE ein Name und eine IP-Adresse plus Portnummer vergeben. Bereits verwendete Werte sind für eine erneute Eingabe verboten und führen zu einer Fehlermeldung. Wurde ein Eintrag erfolgreich getätigt, erscheint der korrespondierende Server automatisch in einer Auflistung. Innerhalb dieser Auflistung kann jeder Server jedoch auch manuell gelöscht werden. Alle Server dieser Liste werden innerhalb eines Cookies lokal im Browser gespeichert.



Home
Add server
CSV-File entries
Delete by CSV-File
Single entry
Delete single entry

Name of server

Tupel of ip and port


Submit

Server with the name ISE created!

| Servername | Tupel (IP+Port) | |
|------------|-----------------|---|
| ISE | bla:bla | <div style="background-color: #ffeb3b; padding: 5px; display: inline-block; border-radius: 5px;">Delete</div> |

Abbildung 6.4: Eintrag einer Ziel-ISE

Wie auch in der Konsolen-Version, können im Tab ‚CSV-File entries‘ Einträge mittels einer CSV-Datei erstellt werden, siehe Abbildung 6.5. Dabei finden sich alle erstellten Server innerhalb einer Auswahl. Es muss daraufhin noch das ISE-Passwort und der dazu passende Nutzernamen eingegeben werden. Im Anschluss daran kann über einer Schaltfläche eine CSV-Datei eingelesen werden. Dies führt dazu, dass die darin enthaltenen Einträge untereinander geordnet dargestellt werden. An dieser Stelle können an diesen Einträgen noch Änderungen vorgenommen werden, die mittels der Schaltfläche ‚Save changes‘ abgespeichert werden. Mit ‚Submit‘ werden diese Daten dann auf die ISE gespielt. Dabei werden im Hintergrund zuerst die Authentifizierungsdaten übertragen, welche im Backend mit Hilfe einer Session gespeichert werden. Danach werden zyklisch alle Einträge einzeln auf die ISE übertragen. Diese Übertragung und deren Rückmeldung kann in Abbildung 6.6 eingesehen werden.



Home
Add server
CSV-File entries
Delete by CSV-File
Single entry
Delete single entry

Write to server

ISE

v

ISE-username

ISE-password

CSV-File

Durchsuchen...

test.csv

✓

Submit

| Mac-address | Description | Endpoint policy | Endpoint identity group |
|-------------------|-------------------------|-----------------|-------------------------|
| 99:DF:B6:A9:B1:A1 | This is a test endpoint | Brother-Printer | Synology-Device |
| 99:DF:B6:A9:B1:A2 | This is a test endpoint | Brother-Printer | Synology-Device |
| 99:DF:B6:A9:B1:A3 | This is a test endpoint | Brother-Printer | Synology-Device |
| 99:DF:B6:A9:B1:A4 | This is a test endpoint | Brother-Printer | Synology-Device |

Imprint
Kapsch Group
Terms of Use

Abbildung 6.5: Eingabe per CSV-Datei

Die Gesamtanzahl der zu übertragenden Einträge wird immer mit der aktuellen bereits übertragenen Anzahl abgeglichen und dargestellt. Während der Übertragung werden im Hintergrund noch weitere Einträge in das Benutzerinterface platziert. Es handelt sich hierbei um Statusmeldungen zu allen einzelnen Einträgen. Diese sollten vor allem helfen, mögliche Fehler zu erkennen und diese zu beseitigen. Eine beispielhafte Fehlermeldung wird in Abbildung 6.6 gezeigt.

HomeAdd serverCSV-File entriesDelete by CSV-FileSingle entryDelete single entry

kapsch>>>
challenging limits

Write to server

ISE

ISE-username

hoefimai

ISE-password

CSV-File

Durchsuchen...test.csv

Submit

1 entries have already been processed. 4 entries are left!

Error: Password or username are incorrect!

Error: The transfer of the mac-address 99:DF:B6:A9:B1:A1 was not successfull

ImprintKapsch GroupTerms of Use

Abbildung 6.6: Statusanzeige der Datenübertragung

Die erfolgreiche Übertragung von Daten auf die ISE wird dem Nutzer mittels Statusmeldungen angezeigt. Anschließend ist es möglich, die übertragenen Daten innerhalb der ISE-Verwaltungsoberfläche zu visualisieren. Ein Beispiel für diese Darstellungsweise findet sich in Abbildung 6.7.

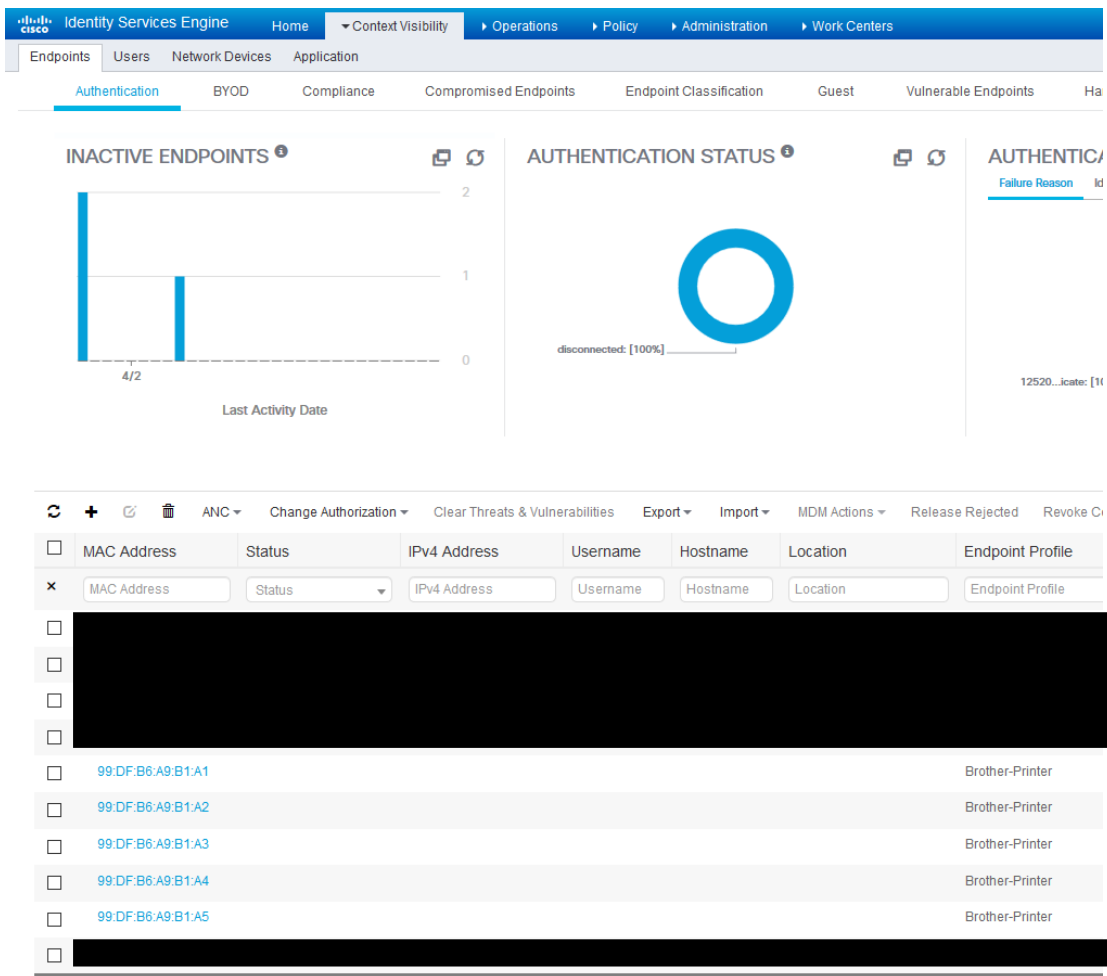


Abbildung 6.7: Erfolgreiche Operation an der ISE

Außerdem wurden, im Vergleich zur Konsolen-Version, hinsichtlich der Performance einige Verbesserungen vorgenommen. Dabei werden die bereits verwendeten Gruppennamen und deren Identifikationsnummer zwischengespeichert, um dadurch einige Abfragen an die ISE einzusparen. Derselbe Mechanismus wird auch für Profilnamen angewandt. Je nach Menge und Zusammensetzung der zu übertragenden Datensätze kann so eine erhebliche Performanceverbesserung erreicht werden. Haben alle Einträge dieselbe Gruppe und dasselbe Profil, kann bei hundert Einträgen eine Leistungsverbesserung von bis zu 50 Prozent erreicht werden. Diese Erkenntnis entstammt einer Reihe von Tests, die im Laufe des Anwendungsbeispiels durchgeführt wurden. Im Tab 'Delete by CSV-File' wird dem Nutzer die Funktion geboten, mithilfe einer CSV-Datei mehrere Einträge zu löschen. Der Mechanismus und die Methodik hinter dieser Funktion ähneln dem Hinzufügen von Daten. Die Anzahl der Abfragen

halbiert sich dabei und die Information innerhalb eines Datensatzes beschränkt sich auf die zu löschende MAC-Adresse.

The screenshot displays the Kapsch web interface with the 'Single entry' tab selected. The interface includes a navigation bar with buttons: Home, Add server, CSV-File entries, Delete by CSV-File, Single entry (highlighted), and Delete single entry. The Kapsch logo with the tagline 'challenging limits' is in the top right. The main form area contains the following fields:

- Write to server:** A dropdown menu with 'ISE' selected.
- ISE-username:** A text input field.
- ISE-password:** A text input field.
- MAC address:** A text input field.
- Description for entry:** A text input field.
- Endpoint policy:** A text input field.
- Endpoint identity group:** A text input field.

A yellow 'Submit' button is located at the bottom of the form. The footer contains links for 'Imprint', 'Kapsch Group', and 'Terms of Use'.


Abbildung 6.8: Eingabe eines Eintrags

Abbildung 6.8 veranschaulicht, wo das Tab ‚Single entry‘ zu finden ist. Hierbei kann in diesem Tab das Hinzufügen eines einzigen Datensatzes, sprich einer MAC-Adresse und deren Begleitinformation, auf die ISE bewerkstelligt werden. Die Daten müssen hierzu in die bereitgestellten Textfelder eingegeben werden und werden dann mittels der Schaltfläche ‚Submit‘ übertragen. Das letzte Tab, mit dem Namen ‚Delete single entry‘, stellt wiederum die zum Tab ‚Delete by CSV-File‘ analoge Funktion zur Verfügung, wobei hier nur jeweils ein Eintrag gelöscht werden kann. Die Identifikation des zu löschenden ISE-Eintrags erfolgt hier wieder über dessen MAC-Adresse.

6.4.3. Webinterface-Version mit Zeitbeschränkung

In dieser Version wurden einige zusätzliche Funktionen in die Applikation eingebracht. Eine ermöglicht es vorab zu definieren, zu welchem Zeitpunkt ein gemachter Eintrag automatisch gelöscht werden soll, und führt diese Löschung

automatisiert durch. Bei der Umsetzung wurde darauf geachtet, alle Funktionen der vorhergehenden Version zu erhalten. Die generelle Funktionsweise der Erstellung und der manuellen Löschung eines Eintrags blieb bei dieser Version gleich. Für die Umsetzung der Funktion des automatischen Löschsens sind jedoch einige Zusatzkomponenten in das Netz integriert worden. Es wird eine Datenbank verwendet, die alle Einträge beinhaltet, die später automatisch gelöscht werden sollen. Dabei wird zusätzlich zu jedem Datensatz die Uhrzeit der Löschung gespeichert. Das Löschen eines Knotens aus der ISE wird über ein zentrales Script geregelt. Dieses wird mittels eines Cronjobs, einem Dienst zum automatischen Ausführen einer Datei, gestartet. Hierbei überprüft das Script, ob ein Datensatz in der Datenbank gelöscht werden soll und führt die Löschung anhand seiner MAC-Adresse durch.



Home
Add server
CSV-File entries
Delete by CSV-File
Single entry
Delete single entry

Write to server

ISE

ISE-username

ISE-password

Auto-delete

☒ Auto-delete enabled

CSV-File

Durchsuchen...

test + time.csv

✓

Submit

| Mac-address | Description | Endpoint policy | Endpoint identity group | Deletion time |
|-------------------|-------------------------|-----------------|-------------------------|---------------|
| 99:DF:B6:A9:B1:A1 | This is a test endpoint | Brother-Printer | Synology-Device | 10 |
| 99:DF:B6:A9:B1:A2 | This is a test endpoint | Brother-Printer | Synology-Device | 10 |
| 99:DF:B6:A9:B1:A3 | This is a test endpoint | Brother-Printer | Synology-Device | 10 |
| 99:DF:B6:A9:B1:A4 | This is a test endpoint | Brother-Printer | Synology-Device | 10 |

Abbildung 6.9: Hinzufügen von ISE-Einträgen

Die genauen Einstellungen der verschiedenen Komponenten müssen sich an der Menge der zu tätigenen Löschungen orientieren, weil das Script zeitlichen Einschränkungen genügen muss, die mit der Menge der Löschungen skalieren. In Abbildung 6.9 ist die Benutzeroberfläche dieser Version zu sehen. Alle visuellen Änderungen zwischen der letzten und dieser Version können in Abbildung 6.9 und

Abbildung 6.5 verglichen und eingesehen werden. Bei dieser Version tritt vor allem das Feld ‚Auto-delete‘ in den Vordergrund. Dieses dient als logischer Schalter, um zu entscheiden, ob die einzutragenden Einträge auch wieder automatisch gelöscht werden sollen. Falls diese Option aktiviert ist, werden nur CSV-Dateien als Eingabemedium akzeptiert, die mindestens eine Lösungszeit eingetragen haben. Die Lösungszeit muss ein Wert in Minuten sein. Wie auch in der vorhergehenden Version ist es hier möglich, die Daten der CSV-Datei nach dem Einlesen zu editieren. Wenn die Löszeit leer ist, wird die dazugehörige MAC-Adresse nicht gelöscht, sondern verbleibt dauerhaft in der ISE. Wenn die Löszeit hingegen gleich null ist, wird der Eintrag nach einer vorab definierten Standardzeit, zum Beispiel 60 Minuten, gelöscht. Diese beiden Besonderheiten bezüglich der Löszeit finden sich auch bei der Erstellung eines einzelnen Eintrags.

Eine andere Variante, die schon durch die Versionen aus den Abschnitten 6.4.1 und 6.4.2 unterstützt wird, ist die automatische Löschung eines Eintrags durch die ISE selbst. Dabei hat diese Variante, im Gegensatz zu der zuletzt vorgestellten, den Nachteil, dass sie keine flexible Lösung für jeden einzelnen Eintrag vorsieht. Bei dieser Variante kann aufgrund einer Gruppen- beziehungsweise Profiltugehörigkeit ein Eintrag maximal einmal pro Tag zu einer bestimmten Uhrzeit gelöscht werden. Soll eine Löschung einmal pro Monat oder pro Woche unternommen werden, ist dies auch durch die ISE selbst möglich. Eine solche grobe Einstellung wäre zum Beispiel für Gastbenutzer eines Drahtlosnetzwerkes von Vorteil, um zu verhindern, dass Teilnehmer einer Vorperiode weiterhin zugangsberechtigt sind.

Die Entscheidung, welche der beiden Lösungsvarianten herangezogen wird, ist von den Anforderungen an das Tool und dem Anwendungsbereich abhängig.

7. Fazit und Ausblick

Das Ziel der Arbeit, einen Überblick über neueste Entwicklungen in der Netzwerktechnik zu liefern, wird durch die verschiedenen Anwendungsbeispiele erreicht. Es wurde versucht, die Grundkonzepte prägnant und möglichst einfach zu erklären. Die vorgenommenen Erklärungen wurden mit mehreren ausgewählten Beispielen behandelt und veranschaulicht, um eine tiefergehende Wahrnehmung der Materie zu erreichen. Das erste Anwendungsbeispiel besticht durch eine hohe Lernkurve, die durch die Verwendung von neuartigen Technologien zustande kommt. Die praktische Problemstellung des zweiten Anwendungsbeispiels wird hingegen vor allem durch bereits etablierte Techniken gelöst. Der Abschluss eines dieser Anwendungsbeispiele kann, weil diese nur auf einer Prototypbasis basieren, leider nicht genau festgelegt werden. Nichtsdestotrotz lässt sich hinter den Ideen der Prototypen eine wirtschaftliche Relevanz und ein Potenzial für Weiterentwicklungen erkennen.

Bei einem wirtschaftlichen Einsatz dieser Ideen würde, aufseiten innovationstreibender Unternehmen, sicherlich ein Interesse an einer technologischen Weiterentwicklung bestehen. Daher haben die durchgeführten Anwendungsbeispiele auch immer eine wirtschaftliche Relevanz, die entweder imminently oder erst in absehbarer Zukunft zum Tragen kommt. Einige der erstellten Beispiele könnten bereits jetzt eine beschleunigte Arbeitsweise und eine erhöhte Flexibilität von Arbeitsabläufen ermöglichen.

Der Erfolg der Anwendungsbeispiele im wirtschaftlichen Alltag würde sich hauptsächlich über die Akzeptanz der Nutzer definieren. Dabei könnten zur Steigerung der Akzeptanz später noch weitere Entwicklungsschritte erfolgen und so die Benutzerfreundlichkeit der Anwendungen erhöht werden. Auch eine Verbesserung des Oberflächendesigns könnte ein zukünftiges Ziel darstellen. Hinzukommend wäre eine Weiterentwicklung des DynamicConfigurationWizard zu einem eigenständigen Tool, nicht nur aus wirtschaftlicher Sichtweise, sehr interessant. Außerdem könnte versucht werden, mehr ‚Intelligenz‘ in das Programm zu integrieren, um dem Nutzer zu ermöglichen, weniger Wissen über die einzelnen YANG-Module mitbringen zu müssen. In Zukunft könnte ein solches Tool ein einfaches, plattform- und herstellerungebundenes und daher sehr mächtiges Konfigurationstool darstellen. Dabei müsste zuerst eine Integration aller

möglichen YANG-Strukturkombinationen in das Programm vorgenommen werden, um zu gewährleisten, dass die zu sendenden Daten bei einem POST-Befehl unter allen Umständen korrekt sind. Allem voran würde die Integration aller YANG-Strukturen ein großes Feld an weiterer Forschungstätigkeit ermöglichen, wobei dafür eine tiefe Wissensbasis der Materie erforderlich ist.

8. Literaturverzeichnis

- [1] Wikipedia: The Free Encyclopedia, „Software-defined networking,“ Wikipedia Foundation, 14 April 2018. [Online]. Available: https://de.wikipedia.org/wiki/Software-defined_networking. [Zugriff am 14 April 2018].
- [2] Cisco Systems, Inc., „Cisco Software-Defined Access FAQ,“ 18 April 2018. [Online]. Available: <https://www.cisco.com/c/dam/en/us/solutions/collateral/enterprise-networks/software-defined-access/nb-09-sda-faq-cte-en.pdf>. [Zugriff am 18 April 2018].
- [3] K. Karmarkar, „DNA Software Defined-Access Integrating with Existing Network,“ in *BRKCRS-2812*, Barcelona, 2018.
- [4] M. Victor, „Cisco SD-Access - Policy Driven Manageability,“ in *BRKCRS-3811*, Barcelona, 2018.
- [5] Cisco Systems, Inc., „Software-Defined Access Design Guide,“ Cisco Systems, Inc., San Jose, 2018.
- [6] W. Shawn, „Software Defined Access Under The Hood,“ in *BRKCRS-2810*, Las Vegas, 2017.
- [7] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell und C. Wright, „Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks,“ Internet Engineering Task Force, 18 April 2018. [Online]. Available: <https://tools.ietf.org/html/rfc7348>. [Zugriff am 18 April 2018].
- [8] Wikipedia: The Free Encyclopedia, „Virtual Extensible LAN,“ Wikipedia Foundation, 18 April 2018. [Online]. Available: https://en.wikipedia.org/wiki/Virtual_Extensible_LAN. [Zugriff am 18 April 2018].
- [9] J. Henry, „DNA Assurance: Deep Dive,“ in *BRKCRS-3033*, Barcelona, 2018.
- [10] Kapsch BusinessCom, „DNA Programmability Enablement Enterprise Networking,“ in *DNA Programmability Enablement Enterprise Networking*, Wien, 2017.
- [11] Cisco Systems, Inc., „Cisco Application Policy Infrastructure Controller Enterprise Module – Release 1.5 Data Sheet,“ Cisco Systems, Inc., 20 April 2018. [Online]. Available: <https://www.cisco.com/c/en/us/products/collateral/cloud-systems-management/application-policy-infrastructure-controller-enterprise-module/datasheet-c78-730594.html>. [Zugriff am 20 April 2018].
- [12] M. Bjorklund, „The YANG 1.1 Data Modeling Language,“ Internet Engineering Task Force, 23 April 2018. [Online]. Available: <https://tools.ietf.org/pdf/rfc7950.pdf>. [Zugriff am 23 April 2018].
- [13] GitHub Inc., „pyang - An extensible YANG validator and converter in python,“ GitHub Inc., 24 April 2018. [Online]. Available: <https://github.com/mbj4668/pyang>. [Zugriff am 24 April 2018].
- [14] GitHub Inc., „Yang-Explorer - An open-source Yang Browser and RPC Builder Application,“ GitHub Inc., 24 April 2018. [Online]. Available:

- <https://github.com/CiscoDevNet/yang-explorer>. [Zugriff am 24 April 2018].
- [15] M. Bjorklund und L. Berger, „YANG Tree Diagrams,“ Internet Engineering Task Force, 24 April 2018. [Online]. Available: <https://tools.ietf.org/pdf/rfc8340.pdf>. [Zugriff am 24 April 2018].
- [16] M. Bjorklund, A. Bierman und K. Watsen, „RESTCONF Protocol,“ Internet Engineering Task Force, 25 April 2018. [Online]. Available: <https://tools.ietf.org/pdf/rfc8040.pdf>. [Zugriff am 25 April 2018].
- [17] Cisco Systems, Inc., „Introducing the RESTCONF Protocol,“ Cisco Systems, Inc., 25 April 2018. [Online]. Available: <https://learninglabs.cisco.com/lab/06-dmi-04-introducing-the-restconf-protocol/step/1>. [Zugriff am 25 April 2018].
- [18] Cisco Systems, Inc., „Catalyst 4500 Series Switch Software Configuration Guide, 12.2(44)SG,“ 25 April 2018. [Online]. Available: <https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst4500/12-2/44sg/configuration/guide/Wrapper-44SG/swipsla.html>. [Zugriff am 25 April 2018].