Grant Agreement No.: 101015857
Research and Innovation action
Call Topic: ICT-52-2020: 5G PPP - Smart Connectivity beyond 5G

Secured autonomic traffic management for a Tera of SDN flows

TeraFlow

D3.1: Preliminary Evaluation of Life-cycle Automation and High Performance SDN Components

Deliverable type	R
Dissemination level	PU
Due date	31/12/2021
Submission date	29/12/2021
Lead editor	Georgios P. Katsikas (UBITECH)
Authors	Thanos Xirofotos, Dimitrios Klonidis (UBITECH), Ricard Vilalta, Lluis Gifre, Ricardo Martínez (CTTC), Javier Moreno, Sergio González (ATOS), Sami Petteri Valiviita (INF), Oscar Gonzalez de Dios (TID), Peer Stritzinger (STR), Adrian Farrel, Daniel King (ODC)
Reviewers	Ricard Vilalta (CTTC), Georgios P. Katsikas (UBITECH)
Quality check team	Adrian Farrel, Daniel King (ODC)
Work package	WP3

Abstract

This deliverable leverages MS2.1 and MS3.1 to provide (i) implementation aspects of the core components of the TeraFlow operating system (OS) along with (ii) a preliminary evaluation of these components. This preliminary evaluation will provide useful feedback on functional, performance, and scalability aspects of these components; in turn, this feedback will result in (i) a revised TeraFlow architecture with (ii) additional and/or revised requirements and (iii) a potentially improved TeraFlow component design towards the final release (i.e., MS3.3) and the final evaluation (i.e., D3.2) of the core TeraFlow OS components.

Acknowledgment

The research conducted by TeraFlow receives funding from the European Commission H2020 programme under Grant Agreement No 101015857. The European Commission has no responsibility for the content of this document.

Revision History

Revision	Date	Responsible	Comment
0.1	16.07.2021	Editor	Initial document structure
0.2	10.11.2021	UBITECH, CTTC, ATOS, INF, SIAE, TID, ODC	First round of contributions including design overview and interfaces for the core TeraFlow OS components
0.3	10.12.2021	UBITECH, CTTC, ATOS, INF, SIAE, TID, ODC, STR	Second round of contributions with refined design and interfaces per component as well as preliminary results
0.4	17.12.2021	UBITECH, CTTC	Internal review
0.5	23.12.2021	ODC, UBITECH	Quality check and submission

[1] http://creativecommons.org/licenses/by-nc-nd/3.0/deed.en_US

EXECUTIVE SUMMARY

This deliverable summarizes the activities of WP3 during the first year of the project. The objective of this document is to provide: (i) a detailed overview of each core TeraFlow OS component, including internal architecture and adopted technologies; (ii) a set of interfaces per core TeraFlow OS component with clear interactions both with other TeraFlow components and/or external entities; (iii) preliminary results per core TeraFlow OS component gathered throughout the first year of the project.

This document begins with an introductory section that highlights the purpose of this deliverable, its relationship with other deliverables, and an outline of the structure of this document. The second section maps partners to core TeraFlow OS components and presents a taxonomy of these core components across key WP3 aspects detailed in Sections 3-6. Specifically, Section 3 tackles components related to performance (T3.1), Section 4 describes components related to heterogeneous hardware and multi-layer service integration (T3.2), Section 5 addresses components related to SDN automation (T3.3), and Section 6 presents the slicing and multi-tenancy component (T3.4).

This document concludes with Section 7 which sketches an outline for the final WP3 milestone (MS3.3) and deliverable (D3.2) as well as Section 8 which serves as an annex with technical details about various components, such as data models for ONF TR-532 devices and RPC examples for OpenConfig devices.

Table of Contents

List of Figures

List of Tables

Abbreviations

5G	Fifth Generation
5G-PPP	5G Infrastructure Public Private Partnership
ABNO	Application Based Network Optimization
API	Application Programming Interface
ASIC	Application-Specific Integrated Circuit
B5G	Beyond 5G
BGP	Border Gateway Protocol
BSS	Business Support System
DB	Database
E2E	End-to-End
ECA	Event-Condition-Action
FPGA	Field-Programmable Gate Array
FRR	Free-Range Routing
gNMI	gRPC Network Management Interface
gNOI	gRPC Network Operations Interface
gRPC	gRPC Remote Procedure Call
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
I/O	Input-Output
IP	Internet Protocol
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
L1	Layer 1
L2	Layer 2
L3	Layer 3
L3NM	Layer 3 Network YANG Model
LSP	Label Switched Path
MS	Milestone
MW	Microwave
NBI	North-Bound Interface
NOS	Network Operating System
OAS	OpenAPI Specification
OC	OpenConfig
OLS	Open Line System
ONF	Open Networking Foundation
ONOS	Open Network Operating System
OS	Operating System
OSPF	Open Shortest Path First
OSS	Operation Support System
P4	Programming Protocol-independent Packet Processors
PCE	Path Computation Element
PCEP	Path Computation Element Protocol
QoS	Quality of Service
REST	Representational State Transfer
RPC	Remote Procedure Call
SBI	South-Bound Interface
SDN	Software-Defined Networking
SDO	Standards Development Organization

SLA	Service-Level Agreement
SLE	Service-Level Expectation
SLI	Service-Level Indicator
SLO	Service-Level Objective
SQL	Structured Query Language
SR	Segment Routing
TAPI	Transport API
TE	Traffic Engineering
TED	Traffic Engineering Database
VLAN	Virtual Local Area Network
VPN	Virtual Private Network
WP	Work Package
XML	eXtensible Markup Language
ZTP	Zero-Touch Provisioning

1. Introduction

The TeraFlow operating system (OS) is a novel software-defined networking (SDN) controller architecture, aiming at capabilities and deployments of beyond fifth generation (B5G) networks. In this context, TeraFlow OS bridges key gaps in state-of-the art SDN controllers in four distinct areas organized as WP3 tasks:

Focus Area 1 (T3.1): high-performance control plane operations through a revolutionary cloud-native network operating system (NOS) design, based on distributed and fully disaggregated microservices.

Focus Area 2 (T3.2): native support for key transport technologies, such as Internet Protocol (IP), optical, and microwave (MW), as well as emerging next-generation SDN technologies, such as the programmable protocol-independent packet processors (P4).

Focus Area 3 (T3.3): automated, zero-touch provisioning (ZTP) of network services & NOS lifecycle operations.

Focus Area 4 (T3.4): multi-tenant network slicing as a service coupled with service-level agreement (SLA) requirements.

1.1. Objectives

The purpose of this deliverable (D3.1) is threefold. The first objective of the deliverable is to provide core TeraFlow OS components for addressing the four areas introduced above. This is done by mapping all core TeraFlow OS components to the various WP3 tasks (each task corresponds to a section between Section 3 and Section 6 of this document), while providing basic concepts and a detailed design overview per component. The second objective is to position all core TeraFlow OS components within the same ecosystem, thus prescribing how they communicate with each other as well as how they communicate with external entities and systems. This is achieved by associating each component description with a dedicated sub-section in this document describing its interfaces. The third objective of this deliverable is to provide a preliminary evaluation of the features of the core TeraFlow OS components through another sub-section per component outlining unit and functional tests, as well as deployment examples per component.

1.2. Relation with Other Tasks and Deliverables

This deliverable takes inputs from MS2.1, where the preliminary architecture of the TeraFlow OS was introduced. It also takes input from the initial WP3 milestones, i.e., MS3.1 "Study of technical aspects of relevant SDN, Cloud-native and SDO solutions" [8] and MS3.2 "Code freeze for TeraFlow OS components (v1)" [9]. MS3.1 provided the necessary key performance indicators (KPIs) for selecting the proper high-performance cloud-native framework to program TeraFlow micro-services, while MS3.2 captures all TeraFlow OS component development activities throughout the first year of the project through a project-level software repository and installation guidelines.

In turn, this deliverable acts as an outlook for the forthcoming features of the core TeraFlow OS components in the final release, which will be documented in the final deliverable of WP3, i.e., D3.2 "Final evaluation of Life-cycle automation and high performance SDN components". At the same time, the core TeraFlow OS components documented in this deliverable will provide key services and interfaces to the TeraFlow netapps designed and developed in the context of WP4 and documented in D4.1 “Preliminary evaluation of TeraFlow security and B5G network integration”. Finally, the combination of core and netapp TeraFlow components will be put together onto a set of testbeds, forming an integrated TeraFlow OS prototyping environment documented in D5.1 “Testbed setup and prototype integration report”.

1.3. Deliverable Structure

In the rest of this deliverable, Section 2 presents an overview of the core TeraFlow OS components that comprise the entire WP3. Sections 3, 4, 5, and 6 highlight the design overview, interfaces, and preliminary results of the various core TeraFlow OS components across the four tasks in WP3 respectively. Section 7 concludes this work, while laying out a development strategy towards the final evaluation of life-cycle automation and high performance SDN components, which will be reported in the context of D3.2. Finally, Section 8 (Annex) reports data models and example RPCs for certain TeraFlow OS device driver plugins.

2. Core TeraFlow OS Components' Overview

This section provides an overview of the core TeraFlow OS components in the context of WP3. Table 1 shows how these components are mapped to the various WP3 tasks and the corresponding partners that have been carrying out their design, implementation, and preliminary evaluation during the first year of the project.

Table 1: Mapping of core TeraFlow components to WP3 tasks and the contributing partners.

WP3 Task	Component Name	Involved Partners
T3.1	Context Management	CTTC
	Monitoring	ATOS
	Traffic Engineering	STR
	Auto Scaling	Features covered by Kubernetes Orchestrator (see MS3.2)
	Load balancing	
T3.2	Device	CTTC, TID, SIAE, INF, UBI
	Service	CTTC
T3.3	Automation (ZTP)	UBI
	Policy Management	UBI, ODC
T3.4	Slice Management	ODC

In the following technical sections:

- Section 3 deals with components related to SDN performance, specifically the context management (Section 3.1), monitoring (Section 3.2), and traffic engineering (Section 3.3). As noted in Table 1 and MS3.2 [9], auto-scaling and load-balancing components are provided by the Kubernetes orchestrator framework, which serves as a deployment engine for the TeraFlow microservices;
- Section 4 presents hardware and multi-layer service integration components, namely the Device component with various driver plugins for different SDN devices, such as emulated devices, open line system (OLS) transport API (TAPI) switches, OpenConfig (OC) routers, microwave devices, and P4 whiteboxes, and the Service component;
- Section 5 introduces the automation zero-touch provisioning (ZTP) component as well as the policy management component; and
- Section 6 presents the slice management component.

3. High-Performance SDN Framework

This section provides a design overview, the northbound and southbound interfaces, and preliminary results of the core TeraFlow OS components of T3.1, i.e., the Context Management component (see Section 3.1), the Monitoring component (see Section 3.2), and the Traffic Engineering (TE) component (see Section 3.3).

3.1. Context Management Component

In this section, we describe the Context Management component in charge of storing the configurations and attributes of the different network elements managed by the TeraFlow OS.I In particular, it stores the active contexts, topologies, devices, links, and the services created. For scalability purposes, it makes use of a No-SQL database to optimize the concurrent access into the same storage infrastructure. In the following subsections, we describe the architectural design of the Context Management component, the interfaces it exposes, and we provide some preliminary results of its operation.

3.1.1. Design Overview

The architectural design of the Context Management component is depicted in Figure 1. The component consists of two NBIs, namely a gRPC interface exposed to the rest of the TeraFlow OS components, and a REST interface exposed to external systems, such as Operations Support System (OSS) or Business Support System (BSS), that might have to retrieve the internal status of the resources managed by the TeraFlow OS. Both interfaces rely on a Context Servicer that dispatches the incoming requests and interacts with the Database API used to interact with a No-SQL database. The Database API enables the Context Management component to use different database backends depending on the needs of the users. Currently, a pure in-memory database backend and a Redis database backend are available. The Context Management component also incorporates publish-subscribe mechanisms over gRPC to broadcast context change events to the rest of the TeraFlow OS components.

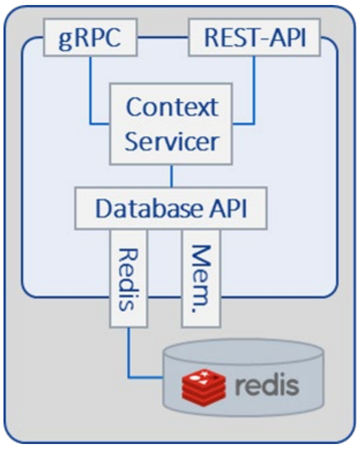


Figure 1: Architecture of the Context Management component.

3.1.2. Interfaces

In this section, we specify the relevant interfaces for the Context Management component. We specify the gRPC interface, the REST interface, and the Database interface.

- **gRPC Interface**

The gRPC interface is offered to the rest of Teraflow OS components to enable them to interact with the Context Management component and get/set/delete information from/into the database, as well as to receive asynchronous events reporting on changes made by other components to the elements stored in the database. The RPC methods exposed by this interface are summarized in Table 2. In short, the interface offers methods to handle the Context, Topology, Device, Link, and Service objects; for each kind of object, the methods provided enable the interface to be used to list identifiers and objects, retrieve, update, and delete objects, and subscribe to receive a stream of change events (create/update/delete event types are supported) per object type.

Table 2: gRPC interface definition for Context Management component.

RPC Method Name	**Parameters**	**Results**
ListContextIds	---	ContextIdList
ListContexts	---	ContextList
GetContext	ContextId	Context
SetContext	Context	ContextId
RemoveContext	ContextId	---
GetContextEvents	---	stream ContextEvent
ListTopologyIds	ContextId	TopologyIdList
ListTopologies	ContextId	TopologyList
GetTopology	TopologyId	Topology
SetTopology	Topology	TopologyId
RemoveTopology	TopologyId	---
GetTopologyEvents	---	stream TopologyEvent
ListDeviceIds	---	DeviceIdList
ListDevices	---	DeviceList
GetDevice	DeviceId	Device
SetDevice	Device	DeviceId
RemoveDevice	DeviceId	---
GetDeviceEvents	---	stream DeviceEvent
ListLinkIds	---	LinkIdList
ListLinks	---	LinkList
GetLink	LinkId	Link
SetLink	Link	LinkId
RemoveLink	LinkId	---
GetLinkEvents	---	stream LinkEvent
ListServiceIds	ContextId	ServiceIdList
ListServices	ContextId	ServiceList
GetService	ServiceId	Service
SetService	Service	ServiceId
RemoveService	ServiceId	---
GetServiceEvents	---	stream ServiceEvent

- **REST Interface**

The REST interface is offered to external systems, such as an OSS/BSS, that need to track the status of the different TeraFlow OS context objects. For this reason, this interface provides only read-only methods to list and retrieve the TeraFlow OS context objects using a JSON-encoded REST-API. The methods available in this interface are summarized in Table 3.

Table 3. REST-API interface definition for Context Management component.

Method	Endpoint URL	Results
GET	/api/context_ids	ContextIdList
GET	/api/contexts	ContextList
GET	/api/context/<context_uuid>	Context
GET	/api/context/<context_uuid>/topology_ids	TopologyIdList
GET	/api/context/<context_uuid>/topologies	TopologyList
GET	/api/context/<context_uuid>/topology/<topology_uuid>	Topology
GET	/api/device_ids	DeviceIdList
GET	/api/devices	DeviceList
GET	/api/device/<device_uuid>	Device
GET	/api/link_ids	LinkIdList
GET	/api/links	LinkList
GET	/api/link/<link_uuid>	Link
GET	/api/context/<context_uuid>/service_ids	ServiceIdList
GET	/api/context/<context_uuid>/services	ServiceList
GET	/api/context/<context_uuid>/service/<service_uuid>	Service

- **Database Interface**

The Database interface provides a list of methods to be implemented in case a developer wants to adapt the TeraFlow OS Context Management component to use a different No-SQL database backend. Right now, backends for an in-memory database, as well as a Redis database are available. The methods required by the Database API are listed in Table 4.

Table 4. Database API Interface definition for Context Management component.

Method	Description
Lock(keys : List[List[str]], owner_key : Optional[str] = None) -> Tuple[bool, str]	Locks/unlocks database objects pointed to by the keys parameter. Each key is a list of strings to enable hierarchical representations of data. The owner_key parameter is a randomly generated string key to tag the locks. If not specified in the lock method, a random owner_key is generated. Only the owner of the lock that knows the owner_key can unlock their locked objects. The lock method returns a Boolean value indicating if the lock was acquired for all requested keys or for none of them, and the owner key used. The unlock method returns a Boolean value indicating if the lock was released correctly for all the keys requested.
Unlock(keys : List[List[str]], owner_key : str) -> bool	
Keys() -> List[str]	Lists the available keys in the database. NOTE: this method traverses the entire database; it should only be used for debug purposes.

Exists(key : List[str]) -> bool	Returns a Boolean value indicating if a key exists in the database.
Delete(key : List[str]) -> bool	Deletes the specified key and returns a Boolean value indicating if the key was properly deleted. If the object does not exist, the error is silently ignored.
Dict_get(key : List[str], fields : List[str] = []) -> Dict[str, str]	Returns the dictionary pointed to by the key parameter. If the fields parameter is specified, it should contain a subset of the fields in the dictionary to be retrieved. If empty, all fields are retrieved. If a field does not exist, only the existing fields are retrieved. If the dictionary does not exist, the error is silently ignored, and an empty dictionary is returned.
Dict_update(key : List[str], fields : Dict[str, str] = {}) -> None	Updates the dictionary pointed to by the key parameter. In particular, it updates the fields specified with the new values provided. If the dictionary does not exist, a new one is created.
Dict_delete(key : List[str], fields : List[str] = []) -> None	Deletes the fields specified by the fields parameter in the dictionary pointed to by the key parameter. If a field does not exist, the error is silently ignored. If the dictionary does not exist, the error is silently ignored.
List_get_all(key : List[str]) -> List[str]	Retrieves all the items in the list pointed to by the key parameter.
List_push_last(key : List[str], item : str) -> None	Pushes a new item at the end of the list pointed to by the key parameter. If the list does not exist, a new one is created.
List_remove_first_occurrence(key : List[str], item: str) -> None	Removes the first occurrence of the item in the list pointed to by the key parameter. If the list does not exist, the error is silently ignored.
Set_add(key : List[str], item : str) -> None	Adds a new item to the set pointed to by the key parameter. If the item already exists, nothing is done. If the set does not exist, a new one is created.
Set_has(key : List[str], item : str) -> bool	Returns a Boolean value indicating if the item specified exists in the set pointed to by the key parameter. If the set does not exist, it assumes it is an empty set and False is returned.
Set_get_all(key : List[str]) -> Set[str]	Retrieves all the items in the set pointed to by the key parameter.
Set_remove(key : List[str], item : str) -> None	Removes the item specified in the item parameter from the set pointed to by the key parameter. If the item does not exist in the set, the error is silently ignored. If the set does not exist, the error is silently ignored.
Dump() -> List[Tuple[str, str, Any]]	Dumps the content of the database as a list of tuples, each containing the object key, the object type, and the object value. NOTE: this method traverses the entire database; it should only be used for debug purposes.

3.1.3. Preliminary Results

Many unit tests have been implemented for the Context Management component. In this section, we report the results of these tests illustrating that all the operations are properly implemented. The results of the tests passed for the “in-memory” and the “Redis” backend, and for the gRPC and REST interfaces are summarized in Table 5. Note that for gRPC tests, each test involves the listing and retrieval of non-existing objects, the creation and update of the objects, the retrieval of existing objects, and the removal of the objects. In all cases, the appropriate constraints (existence of dependencies, correctness of data types, etc.) are checked. Moreover, all the operations tested are interleaved with the testing of the publish-subscribe mechanism used to retrieve notifications when the database objects are created, updated, or deleted.

Table 5. Unit tests passed for the Context Management component.

```
$ docker exec -i $IMAGE_NAME bash -c "pytest --log-level=DEBUG --verbose
$IMAGE_NAME/tests/test_unitary.py"
============================== test session starts ==============================
platform linux -- Python 3.9.6, pytest-6.2.4, py-1.10.0, pluggy-0.13.1 -- /usr/local/bin/python3
cachedir: .pytest_cache
benchmark: 3.4.1 (defaults: timer=time.perf_counter disable_gc=False min_rounds=5 min_time=0.000005
max_time=1.0 calibration_precision=10 warmup=False warmup_iterations=100000)
rootdir: /var/teraflow
plugins: benchmark-3.4.1
collecting ... collected 43 items
context/tests/test_unitary.py::test_grpc_context[all_inmemory] PASSED [  2%]
context/tests/test_unitary.py::test_grpc_topology[all_inmemory] PASSED [  4%]
context/tests/test_unitary.py::test_grpc_device[all_inmemory] PASSED [  6%]
context/tests/test_unitary.py::test_grpc_link[all_inmemory] PASSED [  9%]
context/tests/test_unitary.py::test_grpc_service[all_inmemory] PASSED [ 11%]
context/tests/test_unitary.py::test_rest_populate_database[all_inmemory] PASSED [ 13%]
context/tests/test_unitary.py::test_rest_get_context_ids[all_inmemory] PASSED [ 16%]
context/tests/test_unitary.py::test_rest_get_contexts[all_inmemory] PASSED [ 18%]
context/tests/test_unitary.py::test_rest_get_context[all_inmemory] PASSED [ 20%]
context/tests/test_unitary.py::test_rest_get_topology_ids[all_inmemory] PASSED [ 23%]
context/tests/test_unitary.py::test_rest_get_topologies[all_inmemory] PASSED [ 25%]
context/tests/test_unitary.py::test_rest_get_topology[all_inmemory] PASSED [ 27%]
context/tests/test_unitary.py::test_rest_get_service_ids[all_inmemory] PASSED [ 30%]
context/tests/test_unitary.py::test_rest_get_services[all_inmemory] PASSED [ 32%]
context/tests/test_unitary.py::test_rest_get_service[all_inmemory] PASSED [ 34%]
context/tests/test_unitary.py::test_rest_get_device_ids[all_inmemory] PASSED [ 37%]
context/tests/test_unitary.py::test_rest_get_devices[all_inmemory] PASSED [ 39%]
context/tests/test_unitary.py::test_rest_get_device[all_inmemory] PASSED [ 41%]
context/tests/test_unitary.py::test_rest_get_link_ids[all_inmemory] PASSED [ 44%]
context/tests/test_unitary.py::test_rest_get_links[all_inmemory] PASSED [ 46%]
context/tests/test_unitary.py::test_rest_get_link[all_inmemory] PASSED [ 48%]
context/tests/test_unitary.py::test_grpc_context[all_redis] PASSED [ 51%]
context/tests/test_unitary.py::test_grpc_topology[all_redis] PASSED [ 53%]
context/tests/test_unitary.py::test_grpc_device[all_redis] PASSED [ 55%]
context/tests/test_unitary.py::test_grpc_link[all_redis] PASSED [ 58%]
context/tests/test_unitary.py::test_grpc_service[all_redis] PASSED [ 60%]
context/tests/test_unitary.py::test_rest_populate_database[all_redis] PASSED [ 62%]
context/tests/test_unitary.py::test_rest_get_context_ids[all_redis] PASSED [ 65%]
context/tests/test_unitary.py::test_rest_get_contexts[all_redis] PASSED [ 67%]
context/tests/test_unitary.py::test_rest_get_context[all_redis] PASSED [ 69%]
context/tests/test_unitary.py::test_rest_get_topology_ids[all_redis] PASSED [ 72%]
context/tests/test_unitary.py::test_rest_get_topologies[all_redis] PASSED [ 74%]
context/tests/test_unitary.py::test_rest_get_topology[all_redis] PASSED [ 76%]
context/tests/test_unitary.py::test_rest_get_service_ids[all_redis] PASSED [ 79%]
context/tests/test_unitary.py::test_rest_get_services[all_redis] PASSED [ 81%]
context/tests/test_unitary.py::test_rest_get_service[all_redis] PASSED [ 83%]
context/tests/test_unitary.py::test_rest_get_device_ids[all_redis] PASSED [ 86%]
context/tests/test_unitary.py::test_rest_get_devices[all_redis] PASSED [ 88%]
context/tests/test_unitary.py::test_rest_get_device[all_redis] PASSED [ 90%]
context/tests/test_unitary.py::test_rest_get_link_ids[all_redis] PASSED [ 93%]
context/tests/test_unitary.py::test_rest_get_links[all_redis] PASSED [ 95%]
context/tests/test_unitary.py::test_rest_get_link[all_redis] PASSED [ 97%]
context/tests/test_unitary.py::test_tools_fast_string_hasher PASSED [100%]
============================== 43 passed in 5.97s ==============================
```

3.2. Monitoring Component

The Monitoring component manages the monitoring processes in the TeraFlow OS controller where external agents (subscribers) can subscribe to be able to include or receive information, i.e., metrics or composite key performance indicators (KPIs), coming from different parts of the system depending on their nature, such as the lifecycle of microservices, the connectivity of network services/slices or network devices. In this sense, we assume that a KPI can be composed of one or more metrics available for one or more subscribers. In this regard, for instance, the aggregate outgoing traffic on a switch device can be assumed as a KPI, and the traffic leaving each port of the switch will be established as individual metrics.

The set of functions offered by the Monitoring component will be updated during the project. Some of the functions to be added later in the project include:

- To provide multiple metrics and KPIs
- To manage multiple external subscriptions
- To generate alarms and notifications to the subscribers
- To allow subscribers to add, modify, and visualize metrics and KPIs.

3.2.1. Design Overview

The architecture of the Monitoring component is composed of two main blocks, the Monitoring Core and the Metrics Database, as depicted in Figure 2.

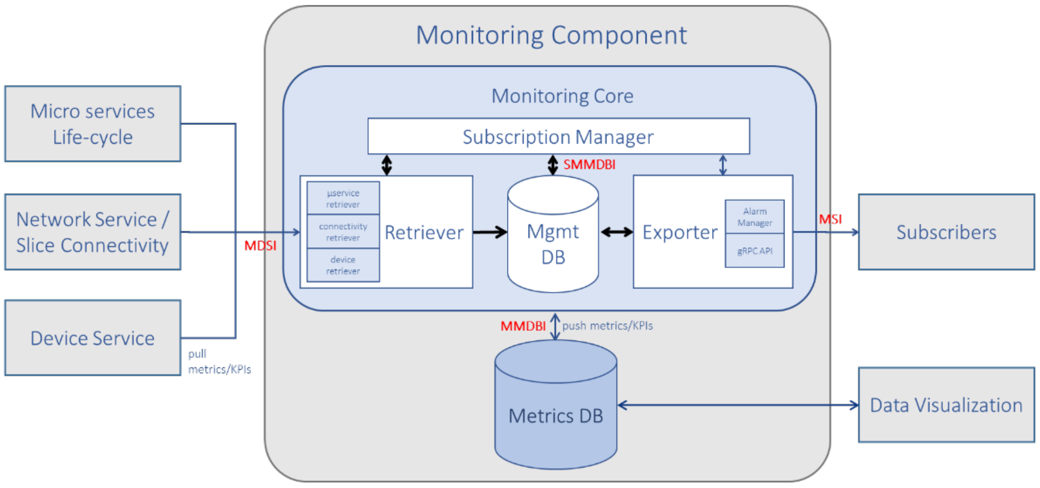


Figure 2: Architecture of the Monitoring component.

- **Monitoring Core**

  This is the main block of the Monitoring component. It implements subscription management and the necessary features to be offered by its server. Likewise, the Monitoring Core can be decomposed in four main sub-blocks according to its functionality:

  - *Subscription Manager* is responsible for managing the information of the subscribers, and for coordinating the actions inside the Monitoring Core among the different sub-components.

- *Management Database* is used to support the Subscription Manager by storing and managing information associated with subscribers, metrics, and KPIs. Here, the metrics and KPIs will have a concrete structure to be in line with the monitoring information models.
- *Retriever* interfaces with external entities to retrieve the necessary information (metrics) requested by the subscribers. Such information is divided into three main groups: microservice life cycle, network service/slice connectivity, and device metrics. Each group will be handled independently within the Retriever.
- *Exporter* provides metrics and KPIs to the subscribers. Additionally, it handles the alarms defined to notify the subscribers according to some specified metrics/KPIs threshold or ranges.

- **Metrics Database**

  The Metrics Database stores and manages the information related to the metrics/KPIs. The structure of the samples to be stored must be mapped to fields in the monitoring information model to provide full interoperability with the Monitoring Core block. Additionally, the Metrics Database directly integrates with a data visualization tool; this way, not only is integration complexity reduced, but the potential issues derived from low-level interoperability are also minimized.

Figure 3 shows a diagram with an example of how to monitor a KPI (or metric) from its registration request to its final delivery in the Data Viewer to be available to users. Such a workflow is divided into five main stages: i) listen for new device events from the context service; ii) create a monitoring KPI; iii) request to monitor the KPI from the Device Service; iv) collect KPI from devices; and v) plot the data in the Data Viewer. For the sake of simplicity, in this example we assume a KPI with only one metric, so KPI and metric terms in this example are interchangeable. These five stages are described in more detail after the figure.

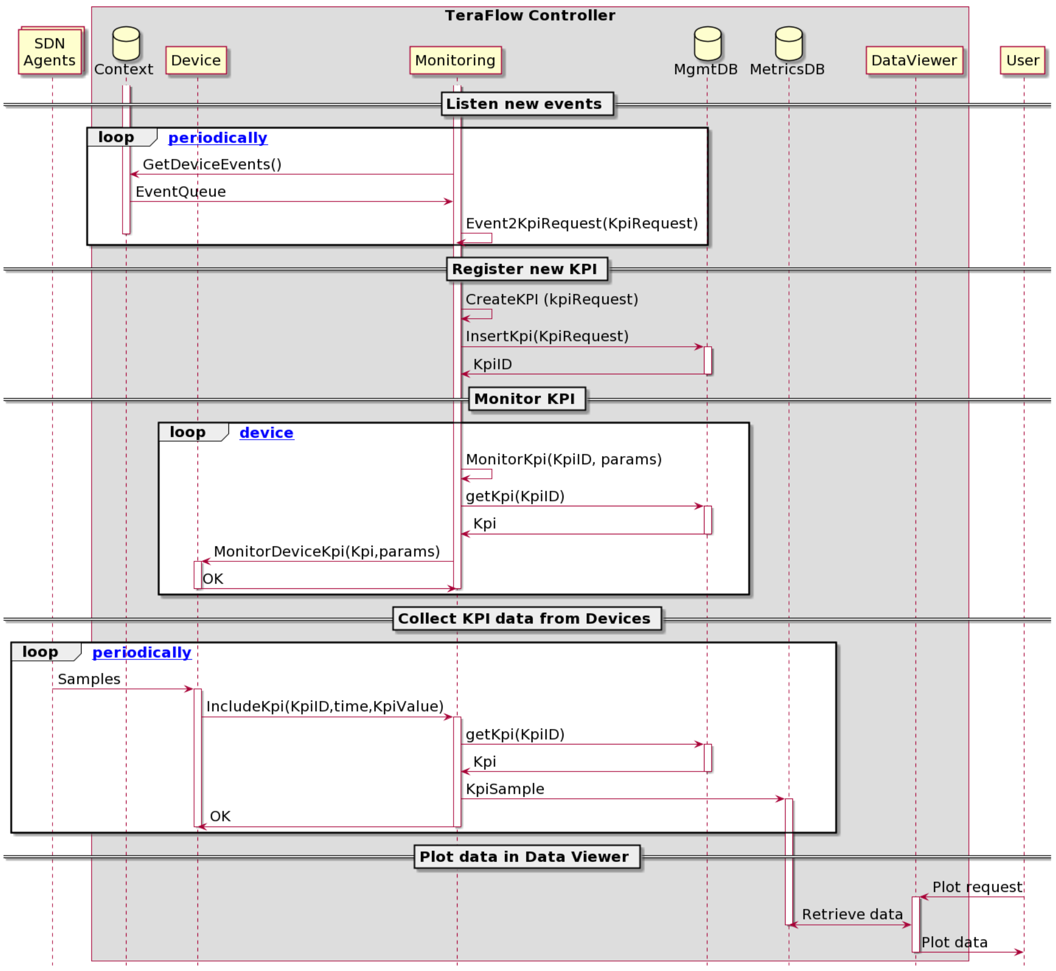


Figure 3: Complete KPI monitoring exemplary workflow.

1. *Listen for new device events from the context service*:
    - This workflow is triggered when the Monitoring component starts listening to the context service about new device events.
    - Once the *EventQueue* is returned as a non-empty list of events, each event will be translated to *KpiRequests* according to its nature and processed individually by the monitoring service.
    - All the necessary information to define a skeleton (structure) for the KPI with some specific fields is extracted from the event and forwarded to the *KpiRequest*, like the *Kpi Sample Type*, *Device ID*, *Service* ID or *Slice ID*.
2. *Create a Monitoring KPI*:
    - Each new *KpiRequest* is registered as a new KPI into the Monitoring component. To do so, the Monitoring Core sends information to the Management DB to include a new KPI by storing the KPI structure, and assigns a unique ID for this KPI request.
    - Then, the *KpiID* assigned by the Management DB and is sent back to the Monitoring Core where the new KPI is correctly registered in the system.
3. *Request to monitor the KPI*:
    - Once a KPIs is registered with the Monitoring component, the Monitoring component requests start monitoring the KPI. Note that this action can be executed automatically by the Monitoring component in this workflow or manually requested by an external subscriber via the monitoring client.

- The Monitoring Core block receives the monitoring KPI request and queries the Management DB about the associated KPI. The Management DB returns the complete information about the KPI.
- The Monitoring component sends a notification to the device service (source information from the monitoring perspective) with all the necessary information about the KPI.
- The device service receives the KPI information and triggers a parallel workflow to retrieve the actual device information from the system. This is the service responsible for the interface between the TeraFlow OS and the SDN agents to communicate with the actual devices.

4. *Collect KPI data from Device*:
   - If the parallel procedure triggered by the device server is properly executed, the device starts to send monitoring data related to the KPI to the device service. Once data reaches the device service, it is forwarded to the Monitoring component by sending a message with a *KpiID*, a timestamp of the sample as measured by the origin, and a value for that measure attached to the message.
   - Then, the Monitoring component uses the *KpiID* to retrieve all the structure defined for the KPI to build its corresponding structure for the KPI sample by including the timestamp and the KPI value.
   - Finally, the complete KPI sample is stored into the Metrics DB.
5. *Plot data in Data Viewer*:
   - The Metrics DB is configured to automatically export the stored data to other external entities, such as a Data Viewer. This way, when KPI samples are available in the Metrics DB, they are ready to be visualized by a user via the Data Viewer.

3.2.2. Interfaces

The Monitoring component has three main interfaces that enable connectivity with external entities that permit it to perform its tasks. and the interfaces are grouped according to their functional nature as presented in red font in Figure 2: the Monitoring-Data-Source Interface (MDSI), the Monitoring-Subscribers Interface (MSI), and the Monitoring-MetricsDB Interface (MMDBI).

The information model of the Monitoring component is defined by using the Protocol Buffer syntax that is inherently related to gRPC communications, so the MDSI and MSI are implemented using gRPC-based connectivity. Thus, the Monitoring component provides a gRPC server that exports the methods defined in the information models to be exploited by specific clients to be integrated into the corresponding entities that want to communicate with the Monitoring component.

- **gRPC Methods**

  The list of RPC methods defined in the Monitoring component information model is exposed by the gRPC server is shown in Table 6. This list will be updated in the following versions of the component.

Table 6: gRPC interface definition for Monitoring component.

RPC Method Name	Parameters	Results
CreateKpi	KpiDescriptor	KpiId
IncludeKpi	Kpi	-

MonitorKpi	MonitorKpiRequest	-
GetKpiDescriptor	KpiId	KpiDescriptor
GetStreamKpi	KpiId	stream Kpi
GetInstantKpi	KpiId	Kpi

- **Monitoring-MetricsDB Interface**

Both blocks that comprise the Monitoring component, i.e., the Monitoring Core and the Metrics DB, are deployed in separate Docker containers that belong to the same pod deployed on a Kubernetes cluster. In this case, InfluxDB v1.8 is chosen to act as the Metrics DB. Thus, the MMDBI is built under the InfluxDB v1.8 API by using HTTP-based communication. Based on that, the methods developed for the MMDBI are as shown in Table 7. Like the gRPC Interface, the list of available methods in this interface will be upgraded in the following iterations.

Table 7: MMDBI definition for Monitoring component.

MMDBI Method Name	**Parameters**	**Results**
write_KPI	time,kpi_id,device_id,kpi_sample_type,kpi_value	-
read_KPI_points	-	List(points)

3.2.2.1. Internal Interface

Additionally, we also consider an internal interface within the Monitoring Core block to enable connectivity between the Subscription Manager and the Management Database (SMMBDI). In our implementation the database chosen is SQLite. The methods currently implemented for this interface are presented in Table 8.

Table 8: SMMDBI definition for Monitoring component

SMMDBI Method Name	**Parameters**	**Results**
insert_KPI	kpiDescription,device_id,kpi_sample_type	KpiID
delete_KPI	device_id,kpi_sample_type	-
delete_kpid_id	kpi_id	-
get_KPI	kpi_id	Kpi
get_KPIs	-	List(Kpi)

3.2.3. Preliminary Results

It is worth mentioning that some of the functional blocks exposed by the Monitoring component are only partially implemented at the time of this report. The component will be upgraded and completed in the following iterations.

Here, we present the screenshots of some preliminary results achieved during the development and testing of the Monitoring component regarding the parts that have been fully implemented.

First, Figure 4 shows the results of the unit tests passed related to the gRPC, SQLite, and InfluxDB interfaces as well as the methods implemented in the monitoring server (Monitoring Core), for instance, to retrieve and process new events.

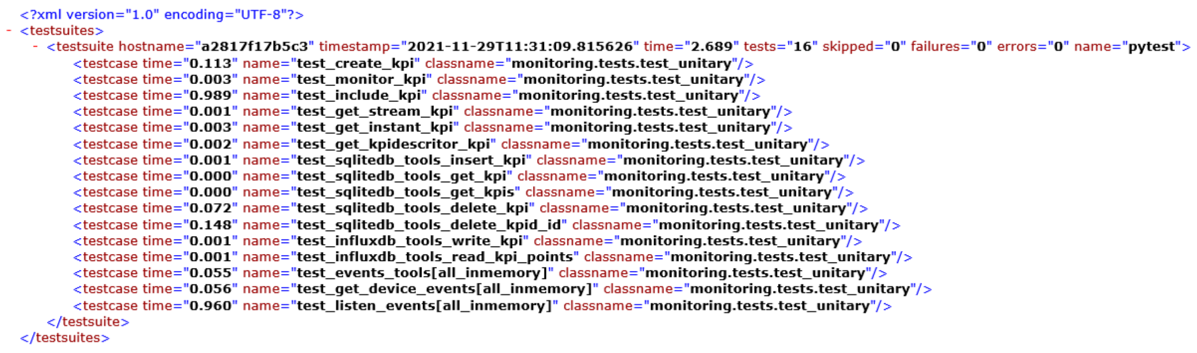

```
<?xml version="1.0" encoding="UTF-8"?>
- <testsuites>
  - <testsuite hostname="a2817f17b5c3" timestamp="2021-11-29T11:31:09.815626" time="2.689" tests="16" skipped="0" failures="0" errors="0" name="pytest">
        <testcase time="0.113" name="test_create_kpi" classname="monitoring.tests.test_unitary"/>
        <testcase time="0.003" name="test_monitor_kpi" classname="monitoring.tests.test_unitary"/>
        <testcase time="0.989" name="test_include_kpi" classname="monitoring.tests.test_unitary"/>
        <testcase time="0.001" name="test_get_stream_kpi" classname="monitoring.tests.test_unitary"/>
        <testcase time="0.003" name="test_get_instant_kpi" classname="monitoring.tests.test_unitary"/>
        <testcase time="0.002" name="test_get_kpidescritor_kpi" classname="monitoring.tests.test_unitary"/>
        <testcase time="0.001" name="test_sqlitedb_tools_insert_kpi" classname="monitoring.tests.test_unitary"/>
        <testcase time="0.000" name="test_sqlitedb_tools_get_kpi" classname="monitoring.tests.test_unitary"/>
        <testcase time="0.000" name="test_sqlitedb_tools_get_kpis" classname="monitoring.tests.test_unitary"/>
        <testcase time="0.072" name="test_sqlitedb_tools_delete_kpi" classname="monitoring.tests.test_unitary"/>
        <testcase time="0.148" name="test_sqlitedb_tools_delete_kpid_id" classname="monitoring.tests.test_unitary"/>
        <testcase time="0.001" name="test_influxdb_tools_write_kpi" classname="monitoring.tests.test_unitary"/>
        <testcase time="0.001" name="test_influxdb_tools_read_kpi_points" classname="monitoring.tests.test_unitary"/>
        <testcase time="0.055" name="test_events_tools[all_inmemory]" classname="monitoring.tests.test_unitary"/>
        <testcase time="0.056" name="test_get_device_events[all_inmemory]" classname="monitoring.tests.test_unitary"/>
        <testcase time="0.960" name="test_listen_events[all_inmemory]" classname="monitoring.tests.test_unitary"/>
    </testsuite>
</testsuites>
```

Figure 4: XML output of the unit tests implemented in the monitoring server.

Figure 5 shows the log output from the monitoring gRPC server. First, the server can be seen running and listening on port 7070. In addition, the log shows the output of the monitoring server when a new KPI is registered with the Monitoring component and shows the unique ID assigned to that KPI action performed by the SQLite database. This action is initially triggered by an external entity through the monitoring client when executing the *CreateKPI* RPC method.

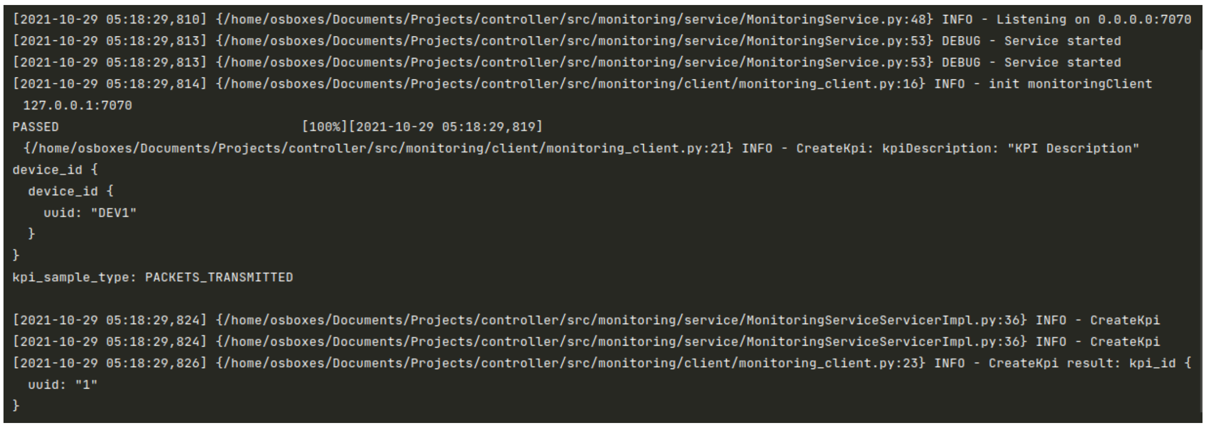

```
[2021-10-29 05:18:29,810] {/home/osboxes/Documents/Projects/controller/src/monitoring/service/MonitoringService.py:48} INFO - Listening on 0.0.0.0:7070
[2021-10-29 05:18:29,813] {/home/osboxes/Documents/Projects/controller/src/monitoring/service/MonitoringService.py:53} DEBUG - Service started
[2021-10-29 05:18:29,813] {/home/osboxes/Documents/Projects/controller/src/monitoring/service/MonitoringService.py:53} DEBUG - Service started
[2021-10-29 05:18:29,814] {/home/osboxes/Documents/Projects/controller/src/monitoring/client/monitoring_client.py:16} INFO - init monitoringClient
 127.0.0.1:7070
PASSED                                  [100%][2021-10-29 05:18:29,819]
 {/home/osboxes/Documents/Projects/controller/src/monitoring/client/monitoring_client.py:21} INFO - CreateKpi: kpiDescription: "KPI Description"
device_id {
  device_id {
    uuid: "DEV1"
  }
}
kpi_sample_type: PACKETS_TRANSMITTED

[2021-10-29 05:18:29,824] {/home/osboxes/Documents/Projects/controller/src/monitoring/service/MonitoringServiceServicerImpl.py:36} INFO - CreateKpi
[2021-10-29 05:18:29,824] {/home/osboxes/Documents/Projects/controller/src/monitoring/service/MonitoringServiceServicerImpl.py:36} INFO - CreateKpi
[2021-10-29 05:18:29,826] {/home/osboxes/Documents/Projects/controller/src/monitoring/client/monitoring_client.py:23} INFO - CreateKpi result: kpi_id {
  uuid: "1"
}
```

Figure 5: Monitoring server logs when a KPI is registered in the Monitoring DB.

Figure 6 shows the log of the InfluxDB. The following results are presented: (i) the POST query triggered to include a KPI point from the Monitoring component (*teraflow* name in the figure); and (ii) a GET query associated to the *read_KPI_points* method defined in the MMDBI. This query is also requested by the Monitoring Core block by using the InfluxDB v1.8 API.

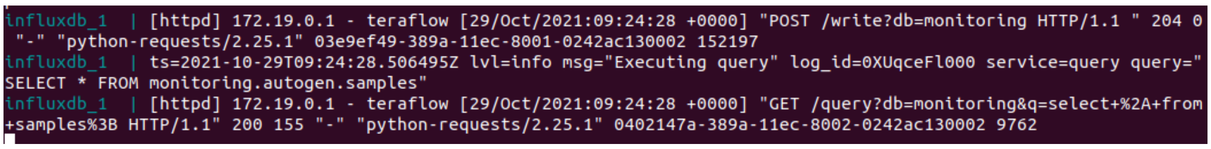

```
influxdb_1  | [httpd] 172.19.0.1 - teraflow [29/Oct/2021:09:24:28 +0000] "POST /write?db=monitoring HTTP/1.1 " 204 0
 "-" "python-requests/2.25.1" 03e9ef49-389a-11ec-8001-0242ac130002 152197
influxdb_1  | ts=2021-10-29T09:24:28.506495Z lvl=info msg="Executing query" log_id=0XUqceFl000 service=query query="
SELECT * FROM monitoring.autogen.samples"
influxdb_1  | [httpd] 172.19.0.1 - teraflow [29/Oct/2021:09:24:28 +0000] "GET /query?db=monitoring&q=select+%2A+from
+samples%3B HTTP/1.1" 200 155 "-" "python-requests/2.25.1" 0402147a-389a-11ec-8002-0242ac130002 9762
```

Figure 6: InfluxDB logs when Monitoring component executes a query to store a KPI point.

3.3. Traffic Engineering Component

The Traffic Engineering (TE) component is responsible for setting up and optimizing Segment Routing (SR) paths in the infrastructure exposed by the Device component. It does so by building a Traffic Engineering Database (TED) from the topology exposed by the Context component and synchronizing all the SR Label Switched Path (LSP) known by the routers that connect to the Path Computation Element (PCE) through the PCE Protocol (PCEP).

The path computation can be fine-tuned by a set of constraints:

- Desired/required path hop-count,
- Desired/required path minimum bandwidth, and
- Desired/required path maximum delay.

3.3.1. Design Overview

The TE component is composed of three sub-components, the TED service, the PCE service, and the Smart Traffic Engineering service as shown in Figure 7.

- **Traffic Engineering Database**
  This part of the component is responsible for building and maintaining a coherent view of the topology.
- **Path Computation Element**
  This part of the component is responsible for establishing PCEP communications with the routers in order to establish new SR paths on request, and to optimise paths it owns (i.e., paths that have been delegated to it).
- **Smart Traffic Engineering**
  This part of the component is the brain that calculates requested LSPs and optimises existing LSPs to satisfy the required constraints.

These services are developed using the Erlang language. With the Erlang virtual machine, more than a million processes can be spawned on modern hardware, with the main limitation being the memory and the input/output (I/O), offering good vertical scaling. In addition, Erlang provides transparent distribution and inter-process communication, making it possible for all these processes to run on different physical machines, therefore scaling the system horizontally. Erlang is a functional language where all the processes are isolated and communicate between each other with messages (i.e., there is no shared memory). This makes Erlang applications reliable and fault-tolerant, as no state corruption is ever propagated to the rest of the system. In addition, the native tools for process monitoring and supervision allow the system to heal itself automatically by restarting any failing process from a known working state.

The Erlang PCE spawns two processes per PCEP connection. The first one handles the TCP/IP connection and the encoding and decoding of the PCEP messages from/to internal data structures. The decoded PCEP messages are sent to the second process that manages the protocol state machine and the cached context data. This second process presents a high-level API to the rest of the service that is mostly PCEP-independent. The PCE itself is formed of a pool of processes each one owning a shard of the flow space. The PCE processes are aware of the session processes with flows defined in their managed space.

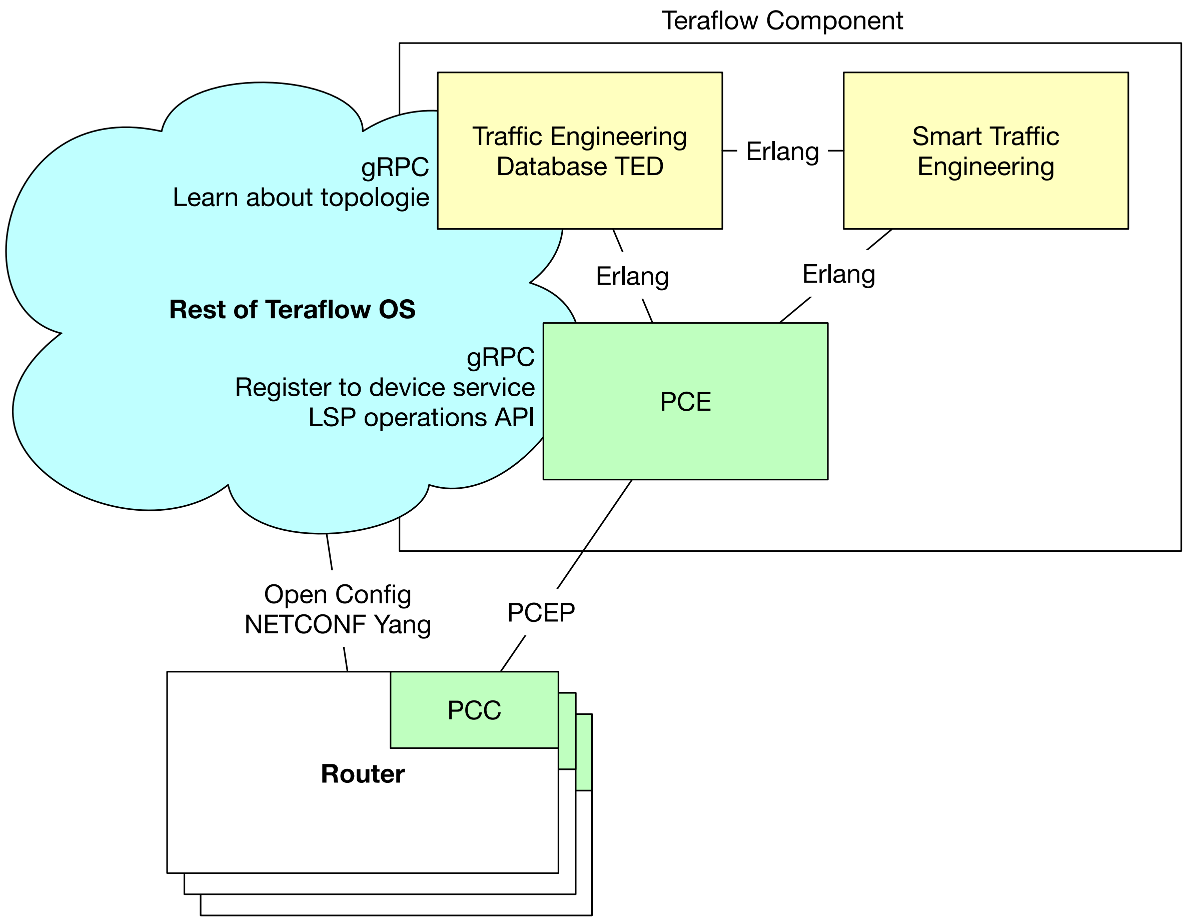


Figure 7: Overview of the TE component.

3.3.2. Interfaces

The interface to the TE component is a set of gRPC calls for the Service component for the Segment Routing LSPs. These calls are presented in Table 9, while their workflows are visualized in Figure 8. This list will be updated in the following versions of the component.

RPC Method Name	Parameters	Results
RequestLSP	Service	ServiceState
UpdateLSP	ServiceId	ServiceState
DeleteLSP	ServiceId	-

Table 9: TE component interface methods.

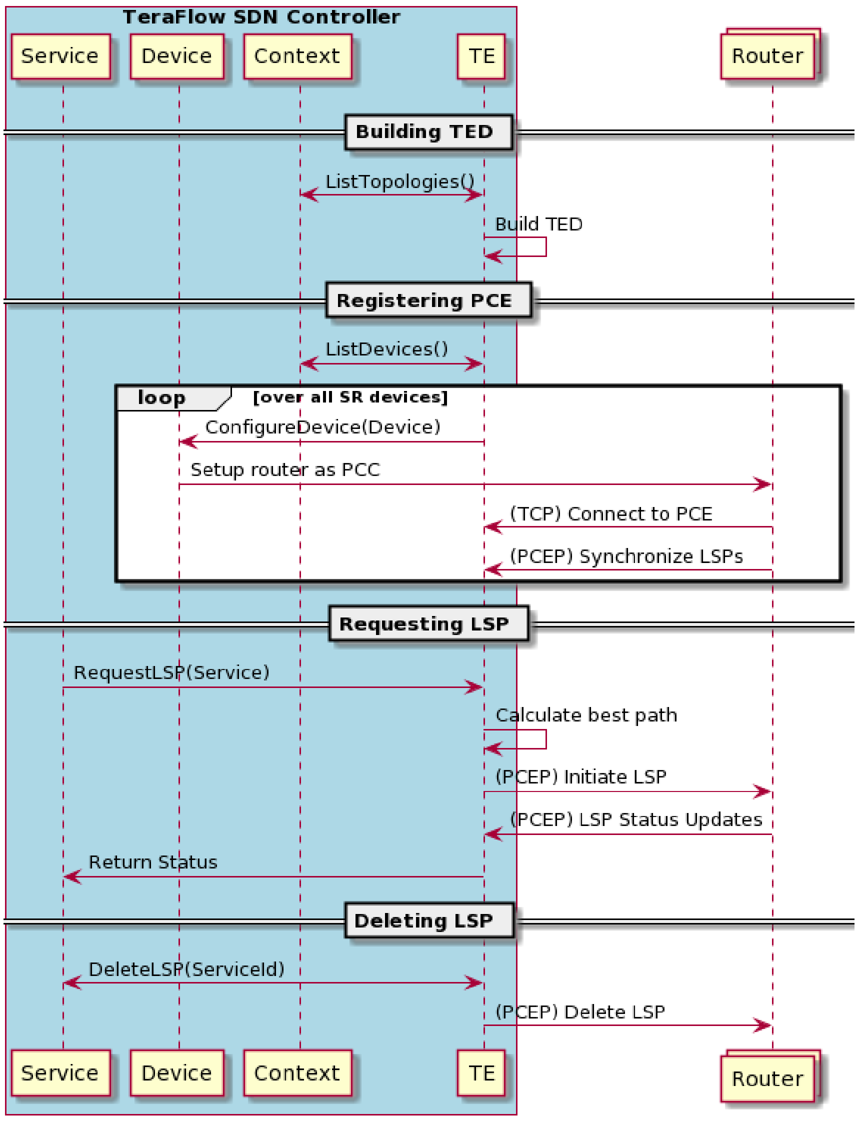


Figure 8: Flow diagram for the main RPCs of the TE component.

1. **Building TED**

   When started, the TE component builds its view of the topology using the API of the Context component.

2. **Registering PCE**
   - The component then iterates over all the relevant devices and configures them to connect to the PCE using PCEP.
   - When the devices' PCCs are configured, they connect to the PCE and start synchronizing their LSPs through PCEP.

3. **Requesting LSP**

   The Service component requests a computed path using the gRPC API. The TE component computes the optimum path and initiates it on the head-end routers connected through PCEP. The TE component monitors the status of the created LSP and, when it is active, responds to the Service component with a status.

4. **Deleting LSP**

The Service component can delete a previously requested LSP through this gRPC API method. This method removes the corresponding LSP from the relevant head-end routers through PCEP.

3.3.3. Preliminary Results

The Traffic Engineering component has already been partially implemented, even though not yet integrated as a fully-fledged TeraFlow Component. Most notably the PCE element was demonstrated at the 2021 IEEE NFV-SDN conference.

The demonstration visualizes the Erlang PCE setting-up and performing Traffic Engineering over a virtualized network composed of six interconnected routers as shown in Figure 9, where each head-end router connects to the Erlang PCE through PCEP. The routers are instances of Free-Range Routing (FRR) daemons running on a single desktop machine with a network virtualized using Linux namespaces.

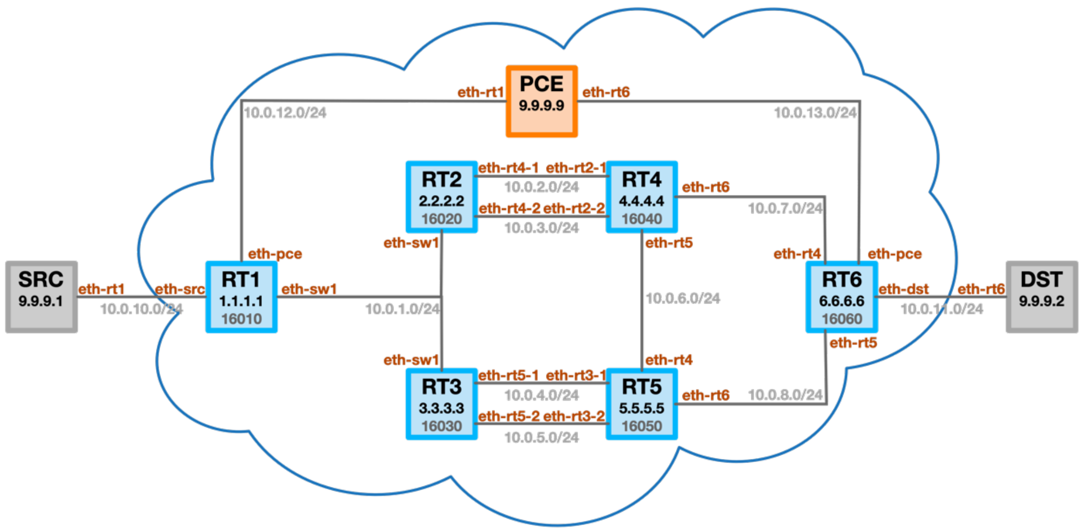
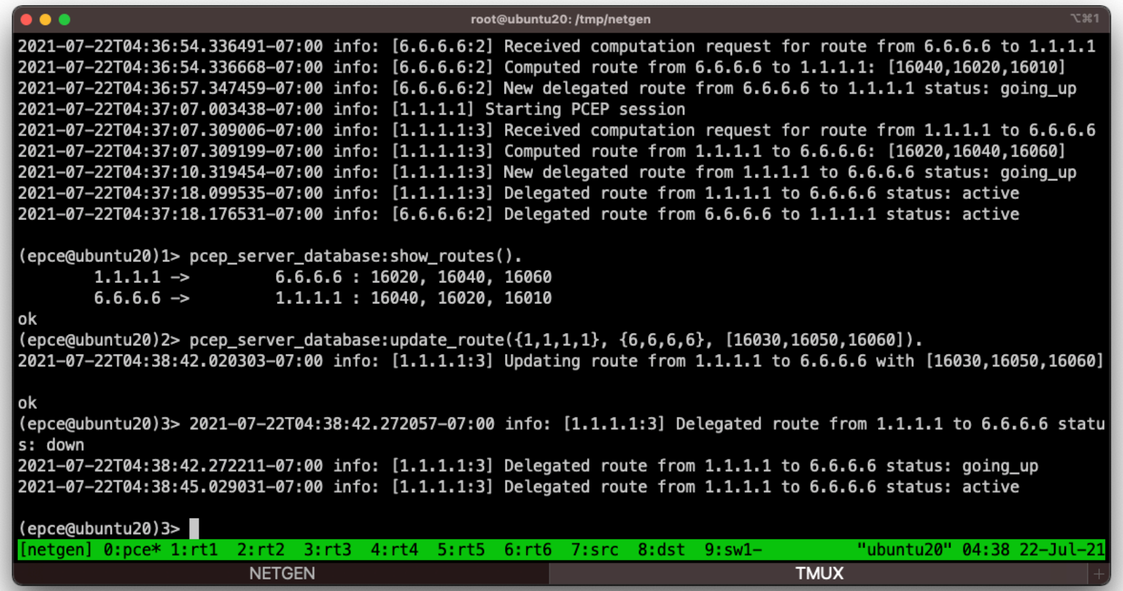


Figure 9: Virtual topology used for demonstrating Transport Engineering.

A flow of ICMP packets is initiated between the source and destination, and the flow is directed from one side of the topology to the other side using the PCE console as shown in Figure 10. The demonstration then validates whether flow migration is effective by using packet inspection.



```
2021-07-22T04:36:54.336491-07:00 info: [6.6.6.6:2] Received computation request for route from 6.6.6.6 to 1.1.1.1
2021-07-22T04:36:54.336668-07:00 info: [6.6.6.6:2] Computed route from 6.6.6.6 to 1.1.1.1: [16040,16020,16010]
2021-07-22T04:36:57.347459-07:00 info: [6.6.6.6:2] New delegated route from 6.6.6.6 to 1.1.1.1 status: going_up
2021-07-22T04:37:07.003438-07:00 info: [1.1.1.1] Starting PCEP session
2021-07-22T04:37:07.309006-07:00 info: [1.1.1.1:3] Received computation request for route from 1.1.1.1 to 6.6.6.6
2021-07-22T04:37:07.309199-07:00 info: [1.1.1.1:3] Computed route from 1.1.1.1 to 6.6.6.6: [16020,16040,16060]
2021-07-22T04:37:10.319454-07:00 info: [1.1.1.1:3] New delegated route from 1.1.1.1 to 6.6.6.6 status: going_up
2021-07-22T04:37:18.099535-07:00 info: [1.1.1.1:3] Delegated route from 1.1.1.1 to 6.6.6.6 status: active
2021-07-22T04:37:18.176531-07:00 info: [6.6.6.6:2] Delegated route from 6.6.6.6 to 1.1.1.1 status: active

(epce@ubuntu20)1> pcep_server_database:show_routes().
        1.1.1.1 ->         6.6.6.6 : 16020, 16040, 16060
        6.6.6.6 ->         1.1.1.1 : 16040, 16020, 16010
ok
(epce@ubuntu20)2> pcep_server_database:update_route({1,1,1,1}, {6,6,6,6}, [16030,16050,16060]).
2021-07-22T04:38:42.020303-07:00 info: [1.1.1.1:3] Updating route from 1.1.1.1 to 6.6.6.6 with [16030,16050,16060]

ok
(epce@ubuntu20)3> 2021-07-22T04:38:42.272057-07:00 info: [1.1.1.1:3] Delegated route from 1.1.1.1 to 6.6.6.6 statu
s: down
2021-07-22T04:38:42.272211-07:00 info: [1.1.1.1:3] Delegated route from 1.1.1.1 to 6.6.6.6 status: going_up
2021-07-22T04:38:45.029031-07:00 info: [1.1.1.1:3] Delegated route from 1.1.1.1 to 6.6.6.6 status: active

(epce@ubuntu20)3> 
[netgen] 0:pce* 1:rt1  2:rt2  3:rt3  4:rt4  5:rt5  6:rt6  7:src  8:dst  9:sw1-         "ubuntu20" 04:38 22-Jul-21
```

Figure 10: Performing Traffic Engineering from the Erlang PCE console.

4. Hardware and L0/L3 Multi-layer Integration

This section provides a design overview, the northbound and southbound interfaces, and preliminary results of the core TeraFlow OS components of T3.2, i.e., the Device component (see Section 4.1) and the Service component (see Section 4.2).

4.1. Device Component

In this section, we describe the Device component in charge of interacting with the underlying network equipment. Different protocols and data models might be needed to manage the network equipment; for this reason, the Device component provides a Driver API that enables developers to implement new drivers and integrate them into the TeraFlow OS. We describe the Device component's architectural design, the plugins' framework, and the Driver interface, as well as the interface it exposes to the rest of the TeraFlow OS components, while providing some preliminary results of its operation.

4.1.1. Design Overview

The architectural design of the Device component is depicted in Figure 11. The component consists of a gRPC-based NBI exposed to the rest of TeraFlow OS components, and a set of SBIs that interact with different network equipment using appropriate protocols and data models. In between, the Device Servicer block dispatches the incoming requests and interacts with the SBI Driver API to choose the appropriate driver for each network device. Given that the Device component needs to know about the state and details of the network devices, it makes use of the Context Management component to store and retrieve up-to-date details about the devices using the Context Management gRPC interface. The SBI Driver API enables the Device component to be extended to use different protocols and data models to communicate with various types of programmable devices.

The available driver plugins are listed below, with a link to the corresponding subsection:

- An emulated driver plugin for testing purposes (see Section 4.1.2.1);
- An OLS ONF Transport API [TR547] driver plugin (see Section 4.1.2.2);
- An ONF TR-532 microwave driver plugin (see Section 4.1.2.3);
- A NETCONF [15]/OpenConfig [16] driver plugin for packet routers (see Section 4.1.2.4); and
- A P4 [4] driver plugin for next-generation white box switches (see Section 4.1.2.5).

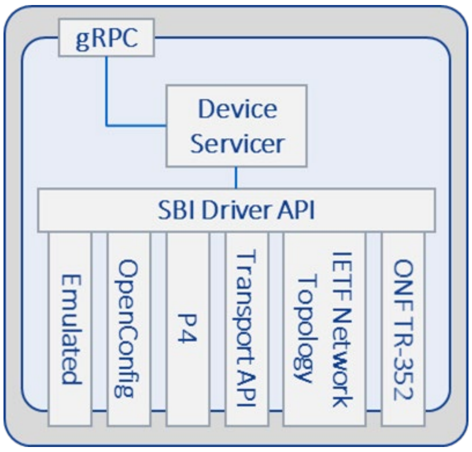


Figure 11: Architecture of the Device component.

4.1.2. Device Plugins

Depending on the device to be managed, the TeraFlow OS Device component loads and uses the respective device plugin to translate abstract device operations into device-specific operations. The implemented drivers follow the Driver API described in Section 4.1.3. In the rest of this section, a variety of device plugins are presented.

4.1.2.1. Emulated Device Driver Plugin

Introduction

This section describes the Emulated Device Driver Plugin. This driver is used only for testing purposes in order to avoid strict dependencies on actual (physical or virtual) devices connected to the Device component for testing.

Supported Function

The function of this driver is very simple; it operates like a normal driver but, instead of interacting with physical/virtual devices, it uses internal memory for storing the configuration values provided. The driver implements all the methods described in the Device Driver API (see Section 4.1.3):

- **Initialization** of the Emulated Driver provides a setting named as “endpoints” to define the endpoints that will be exposed by the emulated device.
- **Connect** and **Disconnect** methods do nothing, since the emulated driver does not need to connect to any external device.
- **GetInitialConfig** and **GetConfig** retrieve an initial configuration and the current configuration set for the emulated device. **GetConfig** supports filtering of the resources according to the parameters specified for the method.
- **SetConfig** and **DeleteConfig** update and delete the configured resources for the emulated driver.
- **SubscribeState** and **UnsubscribeState** activate and deactivate the monitoring of specific resources previously configured.
- **GetState** periodically retrieves synthetic randomly generated values according to the configured sampling durations and intervals.

4.1.2.2. OLS ONF Transport API Driver Plugin

Introduction

This subsection describes the implementation of the TAPI Device Driver that serves as an SBI for the TeraFlow OS inside the Device component. As depicted in Figure 12, we consider the use of the TAPI Driver to interact with an Open Line System (OLS) Controllers in charge of managing underlying optical transport networks. In that way, the entire optical domain managed by the OLS controller is exposed to the TeraFlow OS as a single component with endpoints corresponding to the border endpoints in the optical network.

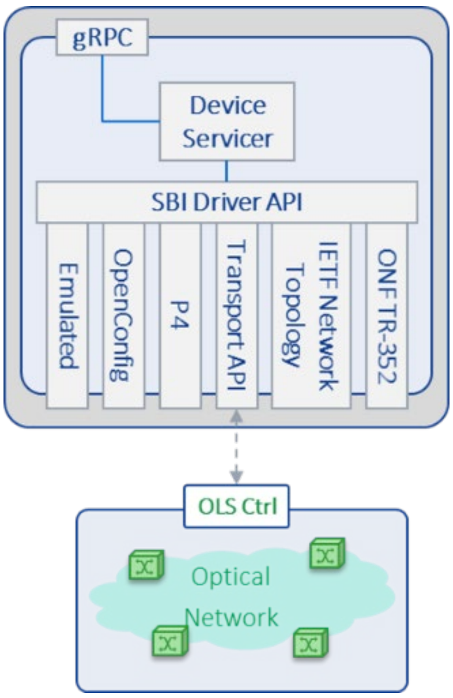


Figure 12: Architecture of the Device component's Transport API.

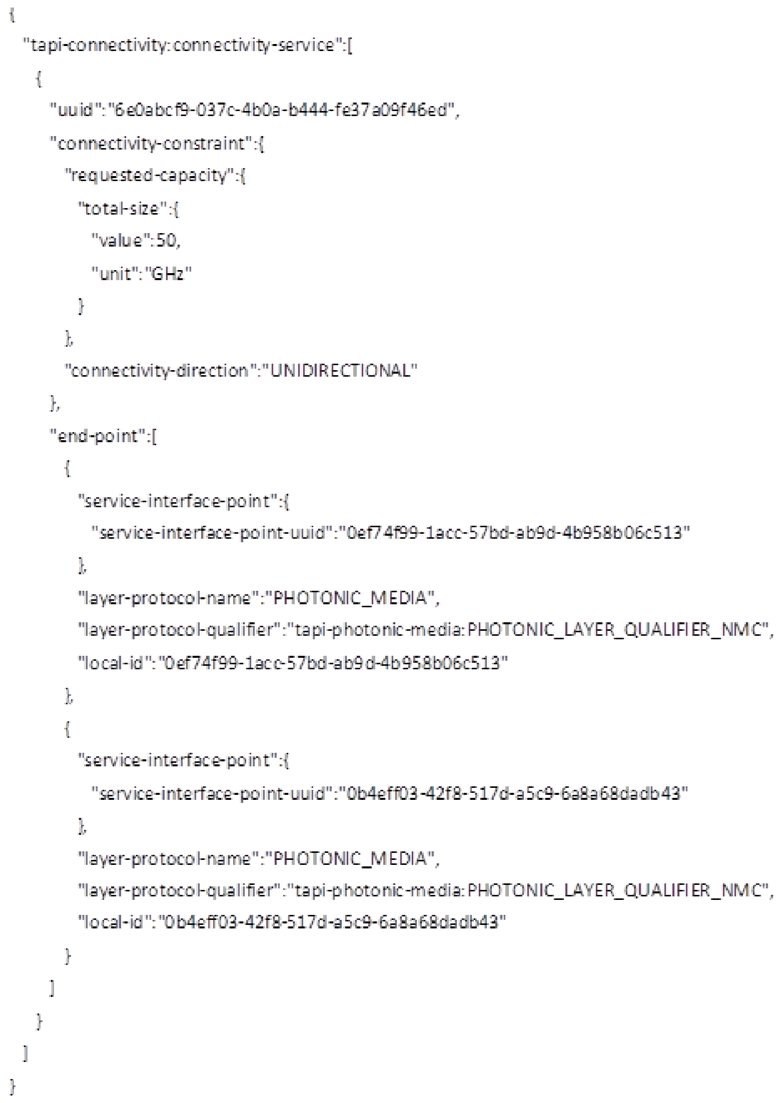
Supported Function

The goal of the TAPI Device Driver is to provide the function needed to establish basic communication with an OLS controller that will, in turn, manage the optical nodes. The interface to be implemented for TAPI-ready nodes is detailed in OpenAPI Specification (OAS) YAML files available in [17]. In particular, the REST API URIs supported for the basic functionality of TAPI are described below:

- **GET /restconf/data/tapi-common:context**
  This URI provides the client with the TAPI context of the server, i.e., the topology, connectivity services, Service Interface Points (SIPs), as well as information about the name of the context and its Universal Unique Identifier (UUID). This URI is used to retrieve information about the TAPI server and to check effective connectivity with it.
- **GET /restconf/data/tapi-common:context/tapi-connectivity:connectivity-context/connectivity-service**
  This URI retrieves the connectivity services present in the TAPI server as well as the underlying connections that support them.
- **DELETE /restconf/data/tapi-common:context/tapi-connectivity:connectivity-context/connectivity-service={uuid}**

This URI enables deletion of the connectivity service associated with a certain UUID.

- **<u>POST /restconf/config/context/connectivity-service/{uuid}</u>**
  This URI provides an endpoint for the creation of connectivity services. The data embedded in the body of the POST request is formatted as a JSON message. The JSON snippet in Figure 13 shows an example of creating a unidirectional 50 GHz connectivity service between two optical network endpoints.



```
{
  "tapi-connectivity:connectivity-service":[
    {
      "uuid":"6e0abcf9-037c-4b0a-b444-fe37a09f46ed",
      "connectivity-constraint":{
        "requested-capacity":{
          "total-size":{
            "value":50,
            "unit":"GHz"
          }
        },
        "connectivity-direction":"UNIDIRECTIONAL"
      },
      "end-point":[
        {
          "service-interface-point":{
            "service-interface-point-uuid":"0ef74f99-1acc-57bd-ab9d-4b958b06c513"
          },
          "layer-protocol-name":"PHOTONIC_MEDIA",
          "layer-protocol-qualifier":"tapi-photonic-media:PHOTONIC_LAYER_QUALIFIER_NMC",
          "local-id":"0ef74f99-1acc-57bd-ab9d-4b958b06c513"
        },
        {
          "service-interface-point":{
            "service-interface-point-uuid":"0b4eff03-42f8-517d-a5c9-6a8a68dadb43"
          },
          "layer-protocol-name":"PHOTONIC_MEDIA",
          "layer-protocol-qualifier":"tapi-photonic-media:PHOTONIC_LAYER_QUALIFIER_NMC",
          "local-id":"0b4eff03-42f8-517d-a5c9-6a8a68dadb43"
        }
      ]
    }
  ]
}
```

Figure 13: Example TAPI Create Connectivity Service.

4.1.2.3. ONF TR-532 Microwave Driver Plugin

Introduction

This subsection gives a high-level description of the data models implemented to manage the following MW device types through an external SDN Controller NBI:

- IETF Network (RFC 8345)
- IETF Network Topology (RFC 8345)
- IETF Eth-Tran Service

- ONF TR-532 Microwave Information Model (as supported by SIAE ALCplus2e MWV equipment)

List of Supported ONF TR-532 Models

The ONF TR-532 Microwave Information Model is a collection of YANG modules managed as augmentation of the main “core-model” module as shown in Table 10.

Table 10: YANG data models corresponding to ONF TR-532 device models.

ONF TR532 Model	YANG model	Revision	Version	Reference
air-interface	air-interface-2-0.yang	2020-01-21	2.0	**Section 8**
co-channel-profile	co-channel-profile-1-0.yang	2020-01-27	1.0	**Section 8**
core-model	core-model-1-4.yang	2019-11-27	1.4	**Section 8**
ethernet-container	ethernet-container-2-0.yang	2020-01-21	2.0	**Section 8**
Firmware	firmware-1-0.yang	2021-04-01	1.0	**Section 8**
hybrid-mw-structure	hybrid-mw-structure-2-0.yang	2020-01-22	2.0	**Section 8**
ltp-augment	ltp-augment-1-0.yang	2020-08-26	1.0	**Section 8**
mac-interface	mac-interface-1-0.yang	2020-01-23	1.0	**Section 8**
pure-ethernet-structure	pure-ethernet-structure-2-0.yang	2020-01-22	2.0	**Section 8**
tdm-container	tdm-container-2-0.yang	2020-01-23	2.0	**Section 8**
vlan-fc	vlan-fc-1-0.yang	2021-02-07	1.0	**Section 8**
vlan-fd	vlan-fd-1-0.yang	2021-01-26	1.0	**Section 8**
vlan-interface	vlan-interface-1-0.yang	2021-01-04	1.0	**Section 8**
wire-interface	wire-interface-2-0.yang	2020-01-23	2.0	**Section 8**
wred-profile	wred-profile-1-0.yang	2020-01-24	1.0	**Section 8**

The detailed information for ONF TR532 modules is publicly available [10]. Section 8 reports a list of parameters available in the models and supported by SIAE ALCplus2e MWV equipment. In the SIAE SDN Controller implementation, the TR-532 model is grafted under the entity “node” of the IETF-Network-Topology to provide a complete view of the microwave network in case a discovery of topology is needed by an E2E service manager. The Driver instance for MW devices does not interact directly with managed network elements, but with the SIAE SDN MW-Domain Controller as a network-level view of the underlying MW domain. This architecture, shown in Figure 14, has two consequences:

1) The MW devices can be still managed individually, but device driver function interfaces must be slightly modified to make explicit the specific device to be managed (via the extra parameter ‘ne_instance’).

2) Additionally, the Device component manages the state and configuration of MW-Domain related attributes, namely: MW Network Topology, MW Link Inventory, and MW End-to-End Ethernet (VLAN) Services. The related data can be exported to/from other TeraFlow OS Core components via suitable data models.

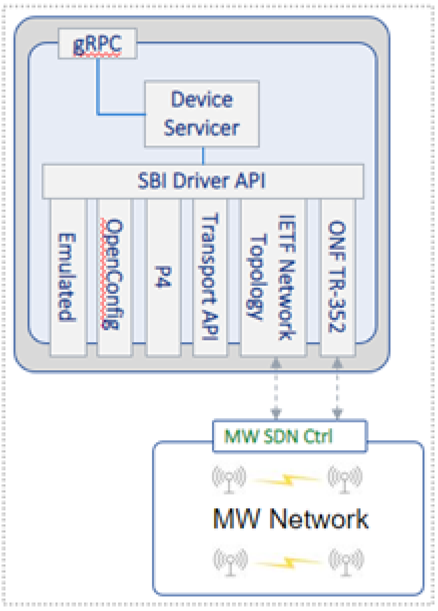


Figure 14: Overview of the interaction between TeraFlow and the MW SDN Controller.

In the following, we describe the list of MW Device Driver APIs supported by the SIAE implementation:

```
def Connect(self, ne_instance : str ) -> bool:
        " Connect to the Device indicated by the argument ne_instance,
        which contains the IP address of the Device. Create the NE instance
        in the SIAE SDN Controller, read configuration data and alarm
        status from device and store them internally.
            Returns:
                succeeded : bool
                  Boolean variable indicating if connection succeeded.

        "

def Disconnect(self, ne_instance : str ) -> bool:
        " Disconnect from the Device indicated by the argument ne_instance
        (IP address). Disconnect and delete NE instance from SIAE SDN
        Controller, removing all stored information.
            Returns:
                succeeded : bool
                  Boolean variable indicating if disconnection succeeded.
        "

def GetConfig(self, resource_keys : List[str], ne_instance : str) ->
List[Union[Any, None, Exception]]:
      " Retrieve running configuration of a Device (either entire or
      selected resource keys), or retrieve the MW-domain configuration as
      network topology, link inventory, or Ethernet (VLAN) services.
            Parameters:
                resource_keys : List[str]
                  List of keys pointing to the resources to be retrieved.
                  Each key can be a configuration attribute to be retrieved
                  from a specific Device (see 8.ANNEX for the list of
                  available attributes) or one of the following keywords:
```

```
                  'NetworkTopology'
                        Return the list of all MWV devices managed by
                        SIAE SDN Controller,
                  'LinkInventory'
                        Return the list of all the physical links
                        (Radio and Ethernet wired) connecting managed
                        MWV devices,
                  'VLAN'
                        Return the list of all the Ethernet services
                        (VLAN) configured into MWV device network
          ne_instance : str
            Specific Device from which the configuration must be
            retrieved.
      Returns:
          values : List[Union[Any, None, Exception]]
            List of values for resource keys requested. Return values
            must be in the same order that the resource keys were
            requested. If a resource is found, the appropriate value
            type must be retrieved, if a resource is not found, None
            must be retrieved in the List for that resource.
        "

def SetConfig(self, resources : List[Tuple[str, Any]], ne_instance : str) -
> List[Union[bool, Exception]]:
      " Create/Update configuration for a list of resources of a specific
      Device, or for MW-domain Ethernet (VLAN) services.
        Parameters:
          resources : List[Tuple[str, Any]]
            List of tuples, each containing either:
                 a resource_key pointing the attribute to be modified
                 or created in the specific Device (see 8.ANNEX for
                 the list of available attributes), and a
                 resource_value containing the new value to be set;
            or:
                 the keyword 'VLAN' to configure an Ethernet service,
                 with configuration parameters encoded in the second
                 element of the tuple.
          ne_instance : str
            Specific Device which must be configured.
        Returns:
          results : List[Union[bool, Exception]]
            List of results for resource key changes requested.
            Return values must be in the same order that the resource
            keys were requested.
        "

def DeleteConfig(self, resource_keys : List[str], ne_instance : str) ->
List[Union[bool, Exception]]:
      " Delete configuration for a list of resource keys of a specific
      Device, or for a MW-domain Ethernet (VLAN) service.
        Parameters:
          resource_keys : List[str]
            List of keys pointing to the resources to be deleted in
            the specific Device, or the keyword 'VLAN:<service_id>'
            indicating the deletion of the Ethernet service
            identified by <service_id>.
          ne_instance : str
            Specific Device from which the listed resources must be
            deleted.
        Returns:
                results : List[bool]
                  List of results for resource key and/or Ethernet service
                  deletions requested. Return values must be in the same
                  order that the keys were requested.
        "

def SubscribeState(self, subscriptions : List[Tuple[str, float, float]],
ne_instance : str) -> List[Union[bool, Exception]]:
      " Subscribe to state information of an entire Device, or selected
      resources. Subscriptions are incremental. Driver should keep track of
      requested resources.
        Parameters:
                subscriptions : List[Tuple[str, float, float]]
                  List of tuples, each containing a resource_key pointing
                  to the resource to be subscribed, a sampling_duration,
                  and a sampling_interval (both in seconds with float
                  representation) defining, respectively, for how long
                  monitoring should last, and the desired monitoring
                  interval for the resource specified.
                ne_instance : str
                  Specific Device from which the state subscription must be
                  set (see 8.ANNEX for the list of state parameters that
                  can be monitored).
        Returns:
                results : List[bool]
                  List of results for resource key subscriptions requested.
                  Return values must be in the same order than resource
                  keys requested.
        "

def UnsubscribeState(self, subscriptions : List[Tuple[str, float, float]],
ne_instance : str) -> List[Union[bool, Exception]]:
      " Unsubscribe from state information for an entire Device, or
      selected resources. Subscriptions are incremental. Driver should keep
      track of requested resources.
        Parameters:
                subscriptions : List[str]
                  List of tuples, each containing a resource_key pointing
                  to the resource to be subscribed, a sampling_duration,
                  and a sampling_interval (both in seconds with float
                  representation) defining, respectively, for how long
                  monitoring should last, and the desired monitoring
                  interval for the resource specified.
                ne_instance : str
                  Specific Device from which the state subscription must be
                  unset (see 8.ANNEX for the list of state parameters that
                  can be monitored).
        Returns:
                results : List[Union[bool, Exception]]
                  List of results for resource key unsubscriptions
                  requested. Return values must be in the same order that
                  the resource keys were requested.
        "

def GetState(self, blocking=False, ne_instance : str) ->
Iterator[Tuple[float, str, Any]]:
      " Retrieve last collected values for subscribed resources of a
      specific Device.
        Parameters:
                blocking : bool
```

```
                        Select the driver behaviour. In both cases, the driver
                        will first retrieve the samples accumulated and available
                        in the internal queue. Then, if blocking, the driver does
                        not terminate the loop and waits for additional samples
                        to come, thus behaving as a generator. If non-blocking,
                        the driver terminates the loop and returns. Non-blocking
                        behaviour can be used for periodically polling the
                        driver, while blocking can be used when a separate thread
                        is in charge of collecting the samples produced by the
                        driver.
                    ne_instance : str
                        Specific Device from which the last collected values must
                        be retrieved.
            Returns:
                    results : Iterator[Tuple[float, str, Any]]
                        Sequences of state samples. Each State sample contains a
                        float Unix-like timestamp of the sample in seconds with
                        up to microsecond resolution, the resource_key of the
                        sample, and its resource_value. Only resources with an
                        active subscription may be retrieved. Interval and
                        duration of the sampling process are specified when
                        creating the subscription using method SubscribeState().
                        Order of values yielded is arbitrary.
            "
```

4.1.2.4. OpenConfig Driver Plugin

Introduction

This subsection includes a high-level introduction of the OpenConfig parameters that are implemented by the Infinera NETCONF server and supported by DRX-30 IP/MPLS routers. The goal of this chapter is to provide a Device Driver interface for populating the configuration templates used to create the requests and generate OC messages that are issued to the routers. The overall setup is depicted in Figure 15, where the NETCONF server described in this section is part of DRX-30 router implementation.

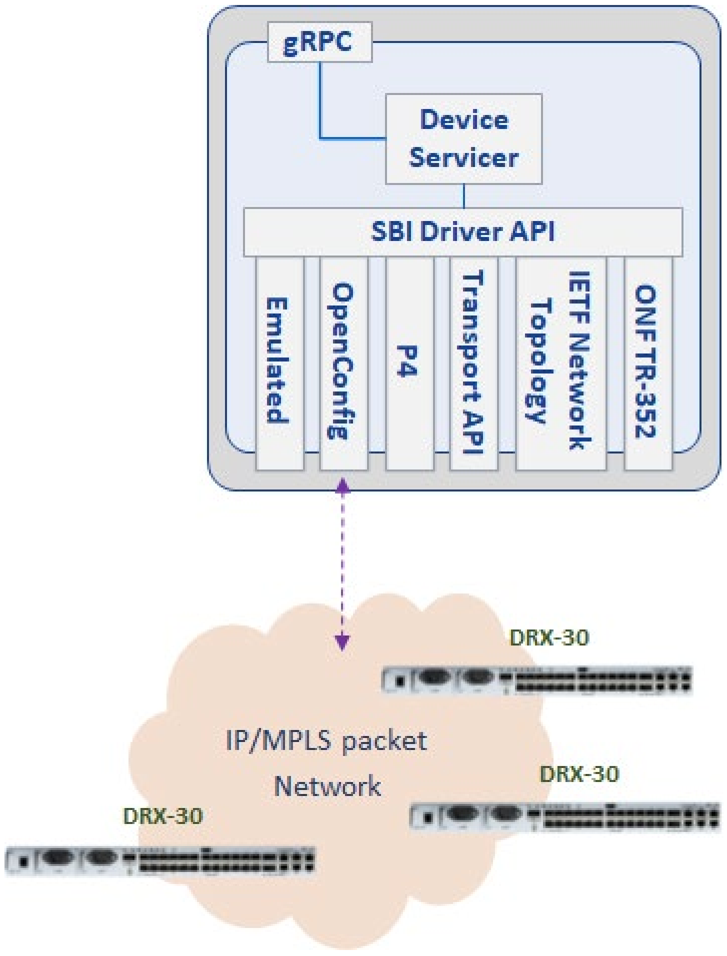


Figure 15: Architecture of the Device component and how it interacts with DRX-30 IP/MPLS routers.

Supported OC functionality

The following OC functionality is supported in the CNOS2.4-SP2 release, which was released November 26, 2021.

- Configure HW inventory
- Read HW-version, serial number, and manufacturing date
- Control fan speed
- Activate/deactivate power supplies
- Monitor power supply state and statistics (capacity, input/output current/voltage, and output power)
- Control interface admin state, MTU, and loopback.
- Enable/disable optical transceiver
- Control interface speed and auto-negotiation mode
- Configure interface IP-address and prefix length
- Configure Link Aggregation (LAG)
- Configure sub-interfaces: VLAN ID
- Configure routing protocols (BGP, OSPF)
- Configure static routes.
- Port statistics

List of OC supported parameters

To provide further details on supported functionality, the OC parameters implemented in the Infinera NETCONF server are listed in Table 11.

Table 11: OC data model parameters exposed by the Infinera NETCONF server.

OpenConfig path	OpenConfig base model	Revision	Version
/components/component/config/name	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/fan/state/oc-fan:speed	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/name	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/oc-linecard:linecard/oc-linecard:config/oc-linecard:power-admin-state	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/oc-linecard:linecard/oc-linecard:state/oc-linecard:power-admin-state	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/oc-linecard:linecard/oc-linecard:state/oc-linecard:slot-id	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/oc-transceiver:transceiver/oc-transceiver:config/oc-transceiver:enabled	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/oc-transceiver:transceiver/oc-transceiver:state/oc-transceiver:enabled	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/port/oc-port:breakout-mode/oc-port:config/oc-port:channel-speed	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/port/oc-port:breakout-mode/oc-port:config/oc-port:num-channels	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/port/oc-port:breakout-mode/oc-port:state/oc-port:channel-speed	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/port/oc-port:breakout-mode/oc-port:state/oc-port:num-channels	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/power-supply/config/oc-platform-psu:enabled	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/power-supply/state/oc-platform-psu:capacity	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/power-supply/state/oc-platform-psu:enabled	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/power-supply/state/oc-platform-psu:input-current	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/power-supply/state/oc-platform-psu:input-voltage	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/power-supply/state/oc-platform-psu:output-current	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/power-supply/state/oc-platform-psu:output-power	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/power-supply/state/oc-platform-psu:output-voltage	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/state/description	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/state/hardware-version	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/state/id	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/state/location	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/state/mfg-date	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/state/oper-status	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/state/parent	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/state/serial-no	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/state/type	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/subcomponents/subcomponent/config/name	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/subcomponents/subcomponent/name	openconfig-platform.yang	16/04/2019	0.12.2
/components/component/subcomponents/subcomponent/state/name	openconfig-platform.yang	16/04/2019	0.12.2
/interface/config/loopback-mode	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/config/description	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/config/enabled	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/config/loopback-mode	openconfig-interfaces.yang	16/04/2019	0.12.2

/interfaces/interface/config/mtu	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/config/name	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/config/oc-vlan:tpid	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/config/type	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/oc-eth:ethernet/oc-eth:config/oc-eth:auto-negotiate	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/oc-eth:ethernet/oc-eth:config/oc-eth:duplex-mode	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/oc-eth:ethernet/oc-eth:config/oc-eth:port-speed	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/oc-eth:ethernet/oc-eth:config/oc-lag:aggregate-id	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/oc-eth:ethernet/oc-eth:state/oc-eth:port-speed	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/oc-lag:aggregation/oc-lag:config/oc-lag:lag-type	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/oc-lag:aggregation/oc-lag:config/oc-lag:min-links	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/state/admin-status	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/state/enabled	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/state/name	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/state/oper-status	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/state/type	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/subinterfaces/subinterface/config/description	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/subinterfaces/subinterface/config/enabled	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/subinterfaces/subinterface/config/index	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/subinterfaces/subinterface/oc-ip:ipv4/oc-ip:addresses/oc-ip:address/oc-ip:config/oc-ip:ip	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/subinterfaces/subinterface/oc-ip:ipv4/oc-ip:addresses/oc-ip:address/oc-ip:state/oc-ip:origin	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/subinterfaces/subinterface/oc-ip:ipv4/oc-ip:addresses/oc-ip:address/oc-ip:config/oc-ip:prefix-length	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/subinterfaces/subinterface/oc-ip:ipv4/oc-ip:config/oc-ip:mtu	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/subinterfaces/subinterface/oc-vlan:vlan/oc-vlan:config/oc-vlan:vlan-id	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/subinterfaces/subinterface/oc-vlan:vlan/oc-vlan:match/oc-vlan:double-tagged/oc-vlan:config/oc-vlan:inner-vlan-id	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/subinterfaces/subinterface/oc-vlan:vlan/oc-vlan:match/oc-vlan:double-tagged/oc-vlan:config/oc-vlan:outer-vlan-id	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/subinterfaces/subinterface/oc-vlan:vlan/oc-vlan:match/oc-vlan:single-tagged/oc-vlan:config/oc-vlan:vlan-id	openconfig-platform	16/04/2019	0.12.2
/interfaces/interface/state/counters/in-octets	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/state/counters/in-pkts	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/state/counters/in-errors	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/state/counters/out-octets	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/state/counters/out-pkts	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/state/counters/out-discards	openconfig-interfaces.yang	16/04/2019	0.12.2
/interfaces/interface/state/counters/out-errors	openconfig-interfaces.yang	16/04/2019	0.12.2
/network-instances/network-instance/config/description	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/config/enabled	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/config/name	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/config/route-distinguisher	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/config/type	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/encapsulation/config/encapsulation-type	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/inter-instance-policies/apply-policy/config/export-policy	openconfig-network-instance	28/11/2019	0.13.2
/network-instances/network-instance/inter-instance-policies/apply-policy/config/import-policy	openconfig-network-instance	28/11/2019	0.13.2

/network-instances/network-instance/interfaces/interface/config/id	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/interfaces/interface/config/interface	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/interfaces/interface/config/subinterface	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/bgp/global/config/as	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/bgp/global/config/router-id	openconfig-network-instance	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/bgp/neighbors/neighbor/apply-policy/config/export-policy	openconfig-network-instance	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/bgp/neighbors/neighbor/apply-policy/config/import-policy	openconfig-network-instance	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/bgp/neighbors/neighbor/config/description	openconfig-network-instance	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/bgp/neighbors/neighbor/config/local-as	openconfig-network-instance	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/bgp/neighbors/neighbor/config/neighbor-address	openconfig-network-instance	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/bgp/neighbors/neighbor/config/peer-as	openconfig-network-instance	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/bgp/neighbors/neighbor/config/peer-type	openconfig-network-instance	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/bgp/neighbors/neighbor/config/remove-private-as	openconfig-network-instance	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/bgp/neighbors/neighbor/ebgp-multihop/config/enabled	openconfig-network-instance	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/bgp/neighbors/neighbor/ebgp-multihop/config/multihop-ttl	openconfig-network-instance	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/config/enabled	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/config/identifier	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/config/name	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/ospfv2/areas/area/config/identifier	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/ospfv2/areas/area/interfaces/interface/config/authentication-	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/ospfv2/areas/area/interfaces/interface/config/id	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/ospfv2/areas/area/interfaces/interface/config/network-type	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/ospfv2/areas/area/interfaces/interface/config/passive	openconfig-network-instance	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/ospfv2/areas/area/interfaces/interface/interface-ref/config/interface	openconfig-network-instance	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/ospfv2/areas/area/interfaces/interface/interface-ref/config/subinterface	openconfig-network-instance	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/ospfv2/global/config/router-id	openconfig-network-instance.yang	28/11/2019	0.13.2

/network-instances/network-instance/protocols/protocol/static-routes/static/config/prefix	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/static-routes/static/next-hops/next-hop/config/metric	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/protocols/protocol/static-routes/static/next-hops/next-hop/config/next-hop	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/table-connections/table-connection/config/address-family	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/table-connections/table-connection/config/dst-protocol	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/table-connections/table-connection/config/import-policy	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/table-connections/table-connection/config/src-protocol	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/tables/table/config/address-family	openconfig-network-instance.yang	28/11/2019	0.13.2
/network-instances/network-instance/tables/table/config/protocol	openconfig-network-instance.yang	28/11/2019	0.13.2
/routing-policy/defined-sets/oc-bgp-pol:bgp-defined-sets/oc-bgp-pol:ext-community-sets/oc-bgp-pol:ext-community-set/oc-bgp-pol:config/oc-bgp-pol:ext-community-member	openconfig-routing-policy.yang	21/11/2018	3.1.1
/routing-policy/defined-sets/oc-bgp-pol:bgp-defined-sets/oc-bgp-pol:ext-community-sets/oc-bgp-pol:ext-community-set/oc-bgp-pol:config/oc-bgp-pol:ext-community-set-name	openconfig-routing-policy.yang	21/11/2018	3.1.1
/routing-policy/policy-definitions/policy-definition/config/name	openconfig-routing-policy.yang	21/11/2018	3.1.1
/routing-policy/policy-definitions/policy-definition/statements/statement/actions/config/policy-result	openconfig-routing-policy.yang	21/11/2018	3.1.1
/routing-policy/policy-definitions/policy-definition/statements/statement/conditions/oc-bgp-pol:bgp-conditions/oc-bgp-pol:match-ext-community-set/oc-bgp-pol:config/oc-bgp-pol:ext-community-set	openconfig-routing-policy.yang	21/11/2018	3.1.1
/routing-policy/policy-definitions/policy-definition/statements/statement/conditions/oc-bgp-pol:bgp-conditions/oc-bgp-pol:match-ext-community-set/oc-bgp-pol:config/oc-bgp-pol:match-set-options	openconfig-routing-policy.yang	21/11/2018	3.1.1
/routing-policy/policy-definitions/policy-definition/statements/statement/config/name	openconfig-routing-policy.yang	21/11/2018	3.1.1
/vlan-tpid-config/tpid (equal to /interfaces/interface/config/oc-vlan:tpid)	openconfig-vlan	16/04/2019	3.2.0

Node discovery

Node discovery defines a data model for representing the system inventory. The model reflects every replaceable unit of the node. Every element in the inventory is termed a “component” with each component having a unique name and type. The uniqueness is guaranteed by the system within the node. Components have properties defined by the system that are modelled as a list of key-value pairs. These may or may not be user configurable.

Each component also has a list of “subcomponents” which are references to other components. Appearance in a list of subcomponents indicates a containment relationship as described above. For example, a line card component is having a list of references to port components that reside on the line card. The properties for each component may include dynamic values, e.g., in the “state” part of the schema. Further details are provided in Section 8, as follows:

- OC query to get the inventory details for all the components
- OC query to get details for all the interfaces.

L3-VPN use case: Service provisioning scenarios to configure a L3-VPN using OC models

The following scenario lists the required steps to configure a L3-VPN in Infinera DRX-30 stacked node:

- Create L3-VPN network-instance
  - <edit-config> - Create L3-VPN network-instance
- Define BGP Route Target for a L3-VPN network-instance
  - <edit-config> "ext-community-set-name"
  - <edit-config> "ext-community-member"
- Create BGP Routing Policies Import/Export for a L3-VPN
  - <edit-config> create a routing-policy
  - <editc-config> create BGP match conditions and action
- Apply BGP Import/export Policy (Route Target) to L3-VPN network instance
  - <edit-config> Apply BGP policy to network-instance
- Configure interfaces/subinterfaces L1/L2 parameters (Ethernet)
  - <edit-config> - Configure interfaces/subinterfaces L1/L2 parameters (Ethernet)
- Configure interfaces/subinterfaces L3 parameters (IP address & IP mask)
  - <edit-config> - Configure interfaces/subinterfaces L3 parameters (IP address & IP mask)
- Add interfaces (endpoint) to L3-VPN network instance
  - <edit-config> - Add interfaces (endpoint) to L3-VPN network instance
- Define routing protocols used within L3-VPN network instance
  - <edit-config> - Define routing protocols for network-instance
  - <edit-config> - Defining OSPF routing protocol for L3 VRF network-instance
- Assign a routing table (RIB) for each protocol within a L3-VPN network instance
  - <edit-config> network-instance routing tables
- Create protocol redistribution policies within a L3-VPN network instance
  - <edit-config> Creating protocol redistribution from STATIC to BGP
  - <edit-config> Creating protocol redistribution from OSPF to BGP
  - <edit-config> Creating protocol redistribution DIRECTLY_CONNECTED to BGP

4.1.2.5. P4 Whitebox Switches Driver Plugin

Introduction

Network devices are typically designed “bottom-up”, with fixed-function chips (i.e., with no run-time reconfigurability) being the heart of the system, thus determining how the device OS is realized and what functionality it can offer. Adding a new feature set to a fixed-function switch is a complex process that takes several months or even years as it requires hardware redesign.

The OpenFlow protocol [3] made a step forward by introducing an open interface to populate the forwarding tables (i.e., hash tables for Ethernet address lookup, longest-prefix match tables for IPv4/IPv6 and wildcard lookups for ACLs) in network switches, thus enabling software-based control planes to control switches from a variety of different vendors. However, OpenFlow still assumes the switches have a fixed behaviour (i.e., a fixed set of tables), which is typically described in the datasheet of a switch ASIC. This means that OpenFlow is unable to change the switch behaviour, e.g., by adding new protocols.

P4 [4] was introduced in 2014 with the purpose of addressing the limitations of the OpenFlow SDN protocol as well as legacy networking paradigms. P4 is an open source, domain-specific programming language for next-generation network devices, also known as whiteboxes, which focuses on describing a “top-down” forwarding plane of programmable (non-fixed-function) switches. With programmable switches, there is no need for fixed protocols, such as OpenFlow. Instead, P4 treats programmable switches just like general purpose processors (e.g., CPUs or GPUs), allowing them to execute code written in a specific programming language (i.e., the P4 language). The code is first compiled by a P4 compiler and then loaded into the processor of the whitebox switch. This way, P4 lets network developers define what headers a switch should be able to parse (including custom or new headers), how to match on each header, and what actions the switch may perform on each header. In P4, OpenFlow is just another program, i.e., one of many possible ways to describe what a forwarding plane does.

Stratum OS

Since the introduction of P4 in 2014, a large community has been established, initially around the P4.org consortium, and since 2019 under the ONF umbrella [5]. In the same year (i.e., 2019), ONF announced the release of the Stratum [6] project as an open-source silicon-independent switch OS for SDN that runs on a variety of switching silicon and whitebox switch platforms. Stratum exposes a set of next-generation SDN interfaces, including P4Runtime, OpenConfig, gRPC, gNMI, and gNOI, enabling greater programmability of forwarding behaviours in interchangeable forwarding devices, thus avoiding the vendor lock-in of today’s data planes through proprietary silicon interfaces and closed software APIs.

ONOS SDN Controller

ONF has implemented several P4 device drivers on top of the Stratum OS (i.e., Barefoot, Mellanox, BMv2) with P4Runtime support within the ONOS SDN controller [7] ecosystem. This way, ONOS uses gRPC messages to deploy a compiled P4 program on a target switch and manage its forwarding tables at run-time. ONOS, Stratum, and P4 are parts of a larger ONF project, titled Software-Defined Fabric (SD-Fabric), which is a fully-programmable cloud-managed network fabric for the emerging edge cloud systems (e.g., Industry 4.0 powered by 5G).

TeraFlow P4 device driver plugin

Rather than re-inventing the wheel, TeraFlow embraces the ONF initiatives around P4 by integrating ONOS, Stratum, and P4 into the TeraFlow ecosystem. In the rest of this section, we describe how the TeraFlow SDN controller will cooperate with ONOS to manage next-generation SDN whiteboxes based on the P4 paradigm. Figure 16 shows a high-level overview of the P4 TeraFlow device driver plugin and the way it interacts with P4 devices through ONOS and Stratum.

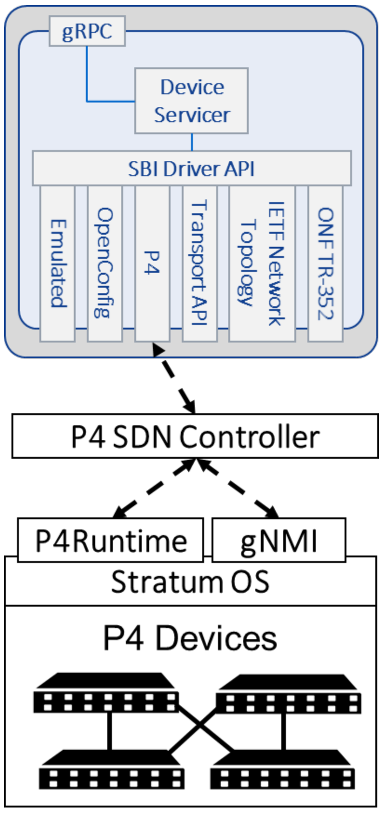


Figure 16: Architecture of the Device component's P4 driver plugin.

The ONOS P4 pipeline

The key interface for the TeraFlow P4 driver plugin is the ONOS NBI. For ONOS to manage P4 devices, the process shown in Figure 17 must be realized.

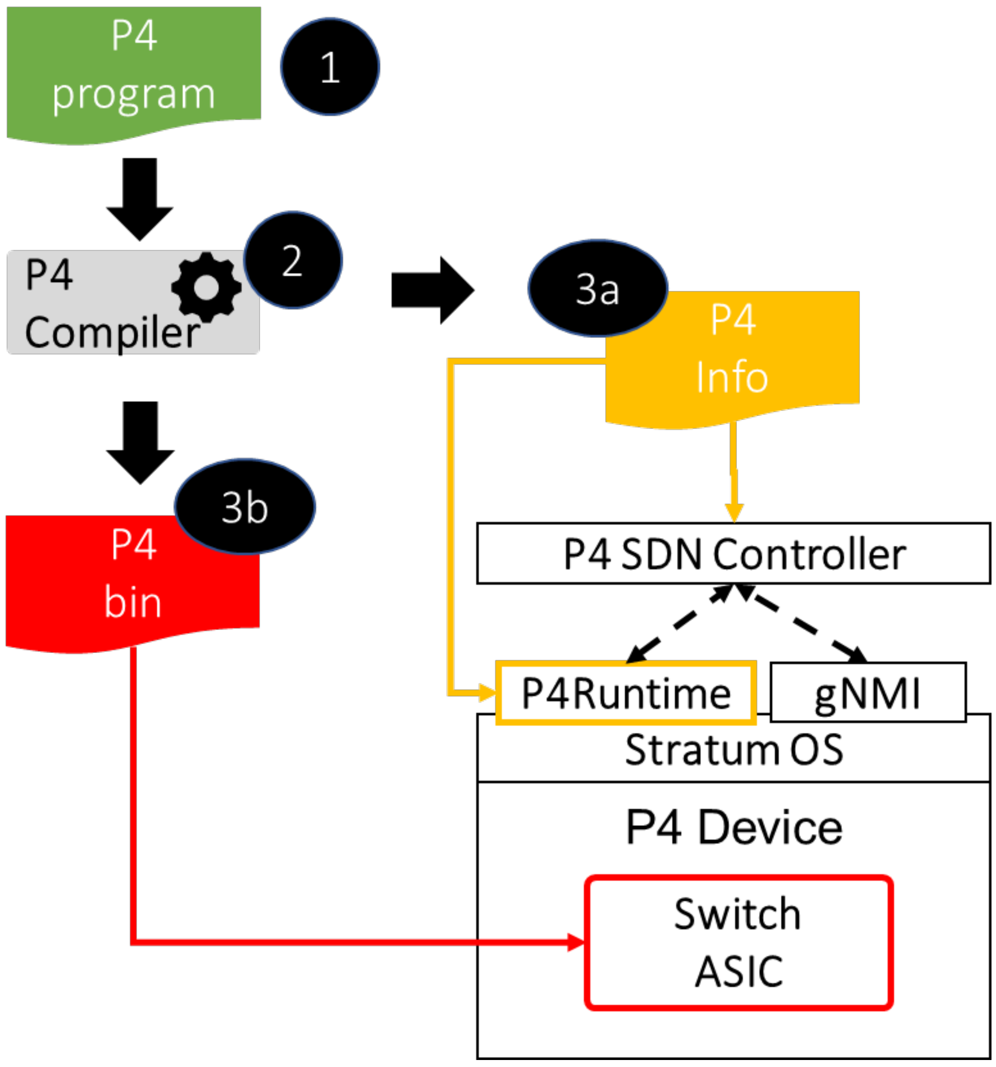


Figure 17: Required steps for a P4 SDN controller to install a P4 program on a P4 device.

Specifically, a desired P4 program needs to be written (step 1) and compiled (step 2) by a P4 compiler. The P4 compiler generates two outputs:

(i) A "P4 Info" file (step 3a) which describes the "schema" of the P4 pipeline for runtime control. This schema captures P4 program attributes such as tables, actions, parameters, etc, in a target-independent format (I.e., same P4Info for a software switch, ASIC, etc.).

(ii) A target-specific "P4 bin" binary (step 3b) used to realize a switch pipeline, such as a binary configuration for an application-specific integrated circuit (ASIC), a bitstream for a field-programmable gate array (FPGA), etc. At runtime the P4 SDN controller uses the gRPC-based P4Runtime interface to manage the match-action pipelines specified in the P4 program.

Early TeraFlow P4 Prototype

To begin this challenging integration with ONOS and Stratum, a minimum but essential subset of RPCs of the TeraFlow device driver is required in order to "detect" a P4 device. This subset includes:

- *Connect()* RPC to initiate a connection with a given set of P4 devices. Since the Connect() RPC requires an IP address and a port (and, optionally, some additional settings) to connect to the target device, the TeraFlow P4 device driver plugin uses the ONOS IP address and port to fetch the active connections with all the P4 devices under the ONOS umbrella. For ONOS to initiate a connection with a P4 device, two actions are required:

  - To register a new pipeconf.oar with a pipeconf ID to the Pipeconf Service through the ONOS NBI.
  - To register a new P4 device event using the device ID, the P4Runtime server address and port, a pipeconf ID (from the previous step), and the P4 driver name (e.g., bmv2 for software P4 switches or Tofino/Mellanox for supported hardware P4 switches).

- *GetConfig()* RPC to fetch device configuration from a given P4 device.
- *Disconnect()* RPC to terminate a connection with a given (set of) P4 device(s). Similar to the connect RPC, the disconnect method terminates the connection with ONOS, thus gets the TeraFlow controller disconnected from the underlying P4 device(s).

As also noted in Section 7, most of the development effort for the P4 device driver plugin will be in the second year of the TeraFlow project. The target set for this deliverable (D3.1) is to establish a working P4 testbed using existing tools, such as Mininet (with software-based bmv2 P4 switches), Stratum OS, and ONOS, and to use a primitive version of the TeraFlow P4 device driver to establish connection with this testbed.

4.1.3. Interfaces

In this section, we specify the relevant interfaces for the Device component, which are the gRPC interface and the SBI Driver interface. Moreover, we use sequence diagrams to describe how other components interact with the Device component and the underlying network equipment.

- **gRPC Interface**

The gRPC interface is offered to the rest of components to enable them to interact with the Device component. The RPC methods available in this interface are summarized in Table 12; these methods enable the addition and removal of devices, for instance, through a user interface, or automatically through future device auto-discovery methods. In addition, these methods enable to configure the network devices. Besides, a method to retrieve the initial configuration of devices has been added for components such as Automation that might need to know the initial configuration settings to be provided for activating device bootstrapping procedures.

Table 12: gRPC interface definition for Device component.

RPC Method Name	**Parameters**	**Results**
AddDevice	Device	DeviceId
ConfigureDevice	Device	DeviceId
DeleteDevice	DeviceId	---
GetInitialConfig	DeviceId	DeviceConfig

- **SBI Driver Interface**

The SBI Driver interface provides a list of methods to be implemented in case a developer wants to extend the Device component to support new network equipment. The methods required by the SBI Driver API are listed in Table 13.

Table 13: SBI Driver API Interface definition for Device component.

Method	**Description**
Connect() -> bool	Connect to the Device. Returns a Boolean value indicating if the connection succeeded. The driver should keep the connection alive, when possible.
Disconnect() -> bool	Disconnect from the Device. Returns a Boolean value indicating if the disconnection succeeded.
GetInitialConfig() -> List[Tuple[str, Any]]	Retrieve initial configuration of an entire device. Returns a list of tuples each containing the pair resource_key and resource_value.
GetConfig(resource_keys : List[str]) -> List[Tuple[str, Union[Any, None]]	Retrieve running configuration of an entire device, or selected resource keys. Returns a list of tuples each containing the pair resource_key and resource_value. If a key is not found, either the key can be ignored in the reply or retrieved with a None value.
SetConfig(resources : List[Tuple[str, Any]]) -> List[bool]	Create/Update a list of resources with new values. Each resource change requested is a tuple containing the pair resource_key and resource_value. Returns a list of Boolean values indicating if the update for each specified key succeeded.
DeleteConfig(resource_keys : List[str]) -> List[bool]	Delete a list of resource keys. Returns a list of Boolean values indicating if the delete for each specified key succeeded.
SubscribeState(subscriptions : List[Tuple[str, float, float]]) -> List[bool]	Subscribe to/Unsubscribe from state information of an entire device, or selected resources. Each subscription is a tuple containing the key pointing to the resource to be subscribed, a sampling_duration, and a sampling_interval (both in seconds with float representation). The methods return a Boolean value for each subscription indicating the success in subscribing or unsubscribing.
UnsubscribeState(subscriptions : List[Tuple[str, float, float]]) -> List[bool]	
GetState(blocking=False) -> Iterator[Tuple[float, str, Any]]	Retrieve the last collected values for subscribed resources. The method should be called once and will block until values are available. When values are available, it should yield each of them and block again until new values are available. When the driver is destroyed, the method returns. The blocking parameter enables selection of the driver behaviour. The driver first retrieves the samples accumulated on its internal queue. Then, if blocking is set, the driver does not terminate the loop and waits for additional samples to come; otherwise, the driver terminates the loop and returns. Each returned sample is a tuple containing a float Unix-like timestamp in seconds with up to microsecond resolution, the resource_key of the sample, and its resource_value.

- **Operational Workflows**

In this subsection, we describe two relevant workflows related to the Device component. The first one, illustrated in Figure 18, describes the process of adding a new network device into the Device component. When a TeraFlow OS component issues an AddDevice request to the Device component, the Device component interrogates the Context Management component and retrieves the current configuration of the device (if any). That way, we ensure there is no other instance of the Device component managing the new network equipment. Then, if the device is not being managed by another instance of the Device component, it selects the appropriate driver for the equipment depending on the supported drivers and the device type. Note that other filter fields, such as the vendor, part number, or the serial number, could be considered in the future to improve the driver selection procedure. Next, the driver is instantiated within the Device component, and a request to connect to the SDN agent running within the network device is issued. If the connection succeeds, the Device component requests to gather the current device configuration through the Driver. Finally, the database provided by the Context Management component is updated with the state of the device and the up-to-date configuration present on the device.

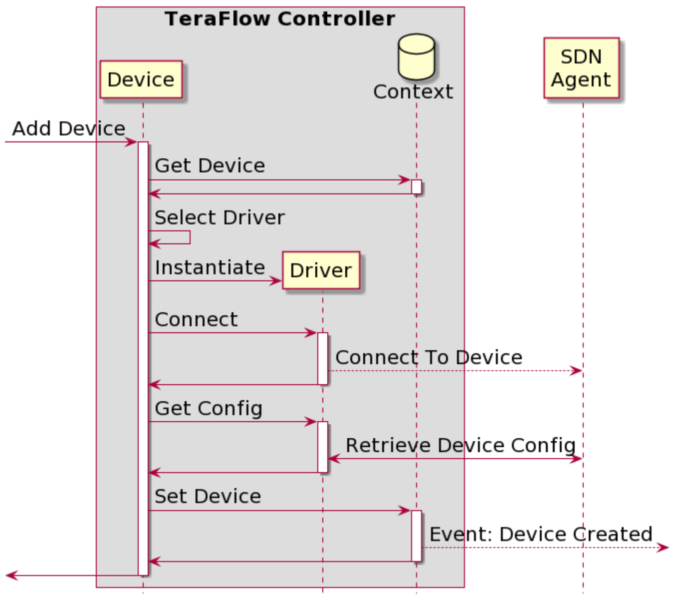


Figure 18: Sequence Diagram for Device component's AddDevice operation.

The second workflow, depicted in Figure 19, emphasizes how the device configuration is applied to the network devices. When a TeraFlow OS component issues a ConfigureDevice request to the Device component, the later interrogates the Context Management component and retrieves the current configuration of the Device to gather the most up-to-date configuration applied to the device. With the target configuration received in the ConfigureDevice request, and the current configuration from the Context Management component, the Device component composes a list of changes to be implemented in the configuration (entries to be set and/or deleted from the configuration, and subscriptions to be created and/or removed). Next, the Device component calls the SetConfig/DeleteConfig/SubscribeState/Unsubscribe state methods with the appropriate parameters to implement the configuration changes in the target network device. Finally, the database at the Context Management component is updated with the up-to-date configuration present on the device.

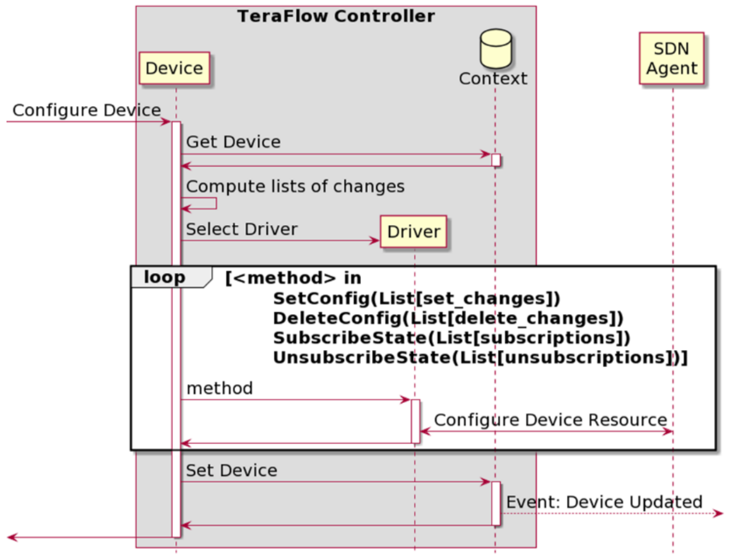


Figure 19: Sequence Diagram for Device component's ConfigureDevice operation.

4.1.4. Preliminary Results

A set of unit tests has been defined for the Device component. In this section, we report the results of the tests illustrating those operations and features for the released components that operate correctly. Note that some of the drivers are work in progress, thus only the tests for the Device Driver API, the Emulated Device Driver, and the OpenConfig Device Driver are reported. Note also that the monitoring functionality of the Device Drivers is work in progress. For the tests, the Device component makes use of an "in-memory" instance of Context Management component.

The results of the tests passed are summarized in Table 14. The tests for the Driver API consist of validating the different methods by means of the Emulated Driver directly without using the gRPC interface of the Device component. Then, the gRPC interface is added when testing the Device component with the Emulated Driver. Finally, the last implemented test focuses on testing the OpenConfig Device Driver, also through the gRPC interface of the Device component. In all cases, the appropriate constraints (existence of dependencies, correctness of data types, confirmation of retrieved values in the test cases, etc.) are checked.

Table 14: Unit tests passed for the Device component.

```
$ docker exec -i $IMAGE_NAME bash -c "pytest --log-level=DEBUG --verbose
$IMAGE_NAME/tests/test_unitary_driverapi.py $IMAGE_NAME/tests/test_unitary.py"
============================= test session starts ==============================
platform linux -- Python 3.9.6, pytest-6.2.5, py-1.11.0, pluggy-1.0.0 -- /usr/local/bin/python3
cachedir: .pytest_cache
benchmark: 3.4.1 (defaults: timer=time.perf_counter disable_gc=False min_rounds=5 min_time=0.000005
max_time=1.0 calibration_precision=10 warmup=False warmup_iterations=100000)
rootdir: /var/teraflow
plugins: benchmark-3.4.1
collecting ... collected 17 items
device/tests/test_unitary_driverapi.py::test_device_driverapi_emulated_setconfig PASSED [  5%]
device/tests/test_unitary_driverapi.py::test_device_driverapi_emulated_getconfig PASSED [ 11%]
device/tests/test_unitary_driverapi.py::test_device_driverapi_emulated_deleteconfig PASSED [ 17%]
device/tests/test_unitary_driverapi.py::test_device_driverapi_emulated_subscriptions PASSED [ 23%]
device/tests/test_unitary.py::test_prepare_environment[all_inmemory] PASSED [ 29%]
device/tests/test_unitary.py::test_device_emulated_add_error_cases[all_inmemory] PASSED [ 35%]
device/tests/test_unitary.py::test_device_emulated_add_correct[all_inmemory] PASSED [ 41%]
device/tests/test_unitary.py::test_device_emulated_get[all_inmemory] PASSED [ 47%]
```

```
device/tests/test_unitary.py::test_device_emulated_configure[all_inmemory] PASSED [ 52%]
device/tests/test_unitary.py::test_device_emulated_deconfigure[all_inmemory] PASSED [ 58%]
device/tests/test_unitary.py::test_device_emulated_delete[all_inmemory] PASSED [ 64%]
device/tests/test_unitary.py::test_device_openconfig_add_error_cases[all_inmemory] PASSED [ 70%]
device/tests/test_unitary.py::test_device_openconfig_add_correct[all_inmemory] PASSED [ 76%]
device/tests/test_unitary.py::test_device_openconfig_get[all_inmemory] PASSED [ 82%]
device/tests/test_unitary.py::test_device_openconfig_configure[all_inmemory] PASSED [ 88%]
device/tests/test_unitary.py::test_device_openconfig_deconfigure[all_inmemory] PASSED [ 94%]
device/tests/test_unitary.py::test_device_openconfig_delete[all_inmemory] PASSED [100%]
============================ 17 passed in 11.69s =============================
```

4.2. Service Component

In this section, we describe the Service component in charge of managing the life-cycle of the connectivity services established in the network. Different service types could be requested and different protocols and data models might be used to configure the network equipment. For this reason, the Service component implements a Service Handler API that enables network operators to precisely define the service types they need to support and the behaviour for each of them. We describe Service component's architectural design, the Service Handler API, and the interface it exposes to the rest of the TeraFlow OS components, and we provide some preliminary results of its operation.

4.2.1. Design Overview

The architectural design of the Service component is depicted in Figure 20. The component consists of a gRPC-based NBI exposed to the rest of TeraFlow OS components, a Service Servicer block that dispatches the incoming requests and interacts with the Service Handler API to choose the appropriate handler for each service type requested. Given that the Service component needs to know about the state and details of the existing connectivity services and the devices supporting them, it makes use of the Context Management component to store and retrieve up-to-date details about the devices and the services using the Context Management gRPC interface. The Service Handler interface enables network operators to extend the Service component to support different service types and use different protocols and data models to configure the devices. Currently, a L3-VPN service using OpenConfig (L3NM-OC) is under development.

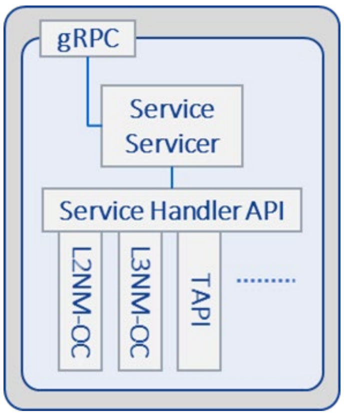


Figure 20: Architecture of the Service component.

4.2.2. Interfaces

In this section, we specify the relevant interfaces for the Service component, which are the gRPC interface and the Service Handler interface. Moreover, we use sequence diagrams to describe how other components interact with the Service component.

- **gRPC Interface**

The gRPC interface is offered to the rest of the components to enable them to interact with the Service component. The RPC methods available in this interface are summarized in Table 15; these methods enable the creation, modification, and removal of services. A method to retrieve the list of established connections for a service is also provided. To discover the available services, the Context Management Database should be interrogated.

Table 15: gRPC interface definition for Service component.

RPC Method Name	Parameters	Results
CreateService	Service	ServiceId
UpdateService	Service	ServiceId
DeleteService	ServiceId	---
GetConnectionList	ServiceId	ConnectionList

- **Service Handler Interface**

The Service Handler interface provides a list of methods to be implemented in case a developer wants to extend the Service component to support new types of services or use different sets of network device configuration protocols and data models. The methods required by the Service Handler interface are listed in Table 16.

Table 16: Service Handler Interface definition for Service component.

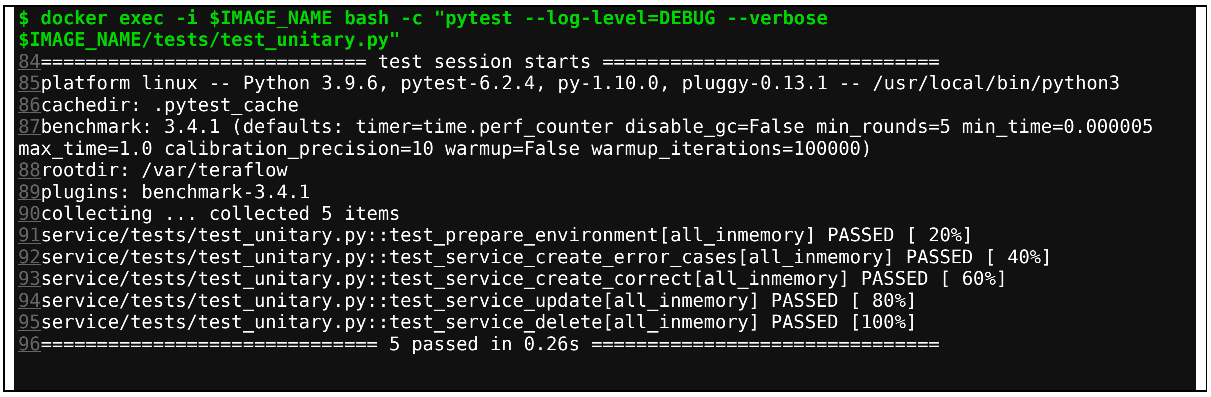
Method	Description
SetEndpoint(endpoints : List[Tuple[str, str, Optional[str]]]) -> List[bool] DeleteEndpoint(endpoints : List[Tuple[str, str, Optional[str]]]) -> List[bool]	Sets/Deletes the service endpoints. Each endpoint to be changed/deleted is defined as a tuple containing a device_uuid, an endpoint_uuid, and, optionally, a topology_uuid. Returns a list of Boolean values indicating whether each addition/removal was successful. Return values must be in the same order as the endpoints requested.
SetConstraint(constraints : List[Tuple[str, Any]]) -> List[bool] DeleteConstraint(constraints : List[Tuple[str, Any]]) -> List[bool]	Sets/Deletes the service constraints. Each constraint to be changed/deleted is defined as a tuple containing a constraint_type and a constraint_value. In Set, constraint_value is the new value to be set for the constraint. In Delete, constraint_value contains optional values that might be needed to locate the constraint to be deleted. Returns a list of Boolean values indicating whether each addition/removal was successful. Return values must be in the same order as the constraints requested.
SetConfig(	

resources : List[Tuple[str, Any]]) -> List[bool]	Sets/Deletes the service configuration resources. Each resource to be changed/deleted is defined as a tuple containing a resource_key and a resource_value. In Set, resource_value is the new value to be set for the resource. In Delete, resource_value contains optional values that might be needed to locate the resource to be deleted. Returns a list of Boolean values indicating whether each addition/removal was successful. Return values must be in the same order as the resources requested.
DeleteConfig(resources : List[Tuple[str, Any]]) -> List[bool]	

4.2.3. Preliminary Results

A set of unit tests has been defined for the Service component. In this section, we report preliminary results for the Service component skeleton. Note that Service Handlers are work in progress, so no specific service is being tested right now.

The results of the tests passed are summarized in Figure 21. The tests for the Service Handler interface consist of validating the different methods offered by the gRPC interface to create, update, and delete the services. A reduced set of constraints is being checked at the moment.



```
$ docker exec -i $IMAGE_NAME bash -c "pytest --log-level=DEBUG --verbose
$IMAGE_NAME/tests/test_unitary.py"
============================== test session starts ==============================
platform linux -- Python 3.9.6, pytest-6.2.4, py-1.10.0, pluggy-0.13.1 -- /usr/local/bin/python3
cachedir: .pytest_cache
benchmark: 3.4.1 (defaults: timer=time.perf_counter disable_gc=False min_rounds=5 min_time=0.000005
max_time=1.0 calibration_precision=10 warmup=False warmup_iterations=100000)
rootdir: /var/teraflow
plugins: benchmark-3.4.1
collecting ... collected 5 items
service/tests/test_unitary.py::test_prepare_environment[all_inmemory] PASSED [ 20%]
service/tests/test_unitary.py::test_service_create_error_cases[all_inmemory] PASSED [ 40%]
service/tests/test_unitary.py::test_service_create_correct[all_inmemory] PASSED [ 60%]
service/tests/test_unitary.py::test_service_update[all_inmemory] PASSED [ 80%]
service/tests/test_unitary.py::test_service_delete[all_inmemory] PASSED [100%]
============================== 5 passed in 0.26s ===============================
```

Figure 21: Unit tests passed for the Service component.

5. SDN Automation

This section provides a design overview, the northbound and southbound interfaces, and preliminary results of the core TeraFlow OS components of T3.3, i.e., the Automation Component (see Section 5.1) and the Policy Management Component (see Section 5.2).

5.1. Automation (ZTP) Component

This section introduces the design overview, interfaces, and preliminary results of the TeraFlow OS Automation component.

5.1.1. Design Overview

The goal of the Automation component, also abbreviated as ZTP, is to provide zero-touch device onboarding, reconfiguration, and deletion functions to the TeraFlow OS as well as to similar SDN controllers or overlay network management tools. To meet this objective, the Automation component is designed according to Figure 22.

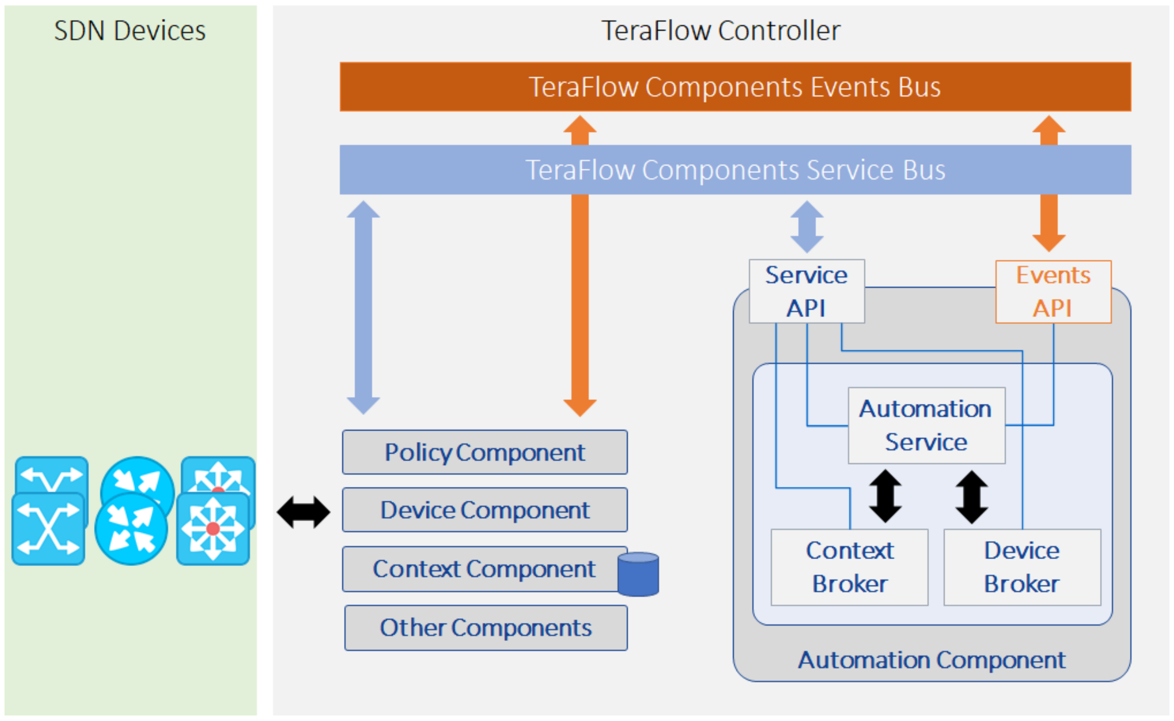


Figure 22: Automation (ZTP) component overview.

Northbound Interfaces: The Automation component presents two NBIs. The first interface, titled “Service API” in Figure 22, exposes basic automation functions to the rest of the TeraFlow OS components. The second interface, titled “Events API” in Figure 22, allows the Automation component to register to, receive, thus react upon relevant events from key TeraFlow OS components. Both interfaces are described in Section 5.1.2.

Automation Core: Upon receiving an incoming request from either of the northbound interfaces, the Automation service (shown in Figure 22) realizes the core logic of the Automation component. As the role of the Automation component is to provision initial or updated configuration to the underlying devices in an automated fashion, it mainly interacts with the device and context TeraFlow OS components. To abstract the communication with these components, the Automation component relies on two internal blocks, namely the “Device Broker” and the “Context Broker”. These blocks take instructions related to the underlying topology, devices, links, etc. and translate them to relevant service calls towards the Device and Context components. This way, the Automation component consumes certain services from these components, without dealing with device and/or context-specific implementation details.

For the Automation component to provide truly zero-touch services, it requires a way to trigger its control loops (i.e., automatically invoke its RPCs) without human intervention or external calls. To do so, the automation services leverage the Event Condition Action (ECA) policy management model [2]: ECA allows delegation of network management functions to the Automation component’s server, which can take instant action when a trigger condition on the managed objects is met. To do so, the Automation component subscribes to key events (e.g., an event that announces the association of a new device with the TeraFlow OS) published by the Context component. In response to such an event, the Automation component’s server automatically launches a relevant RPC to provision/update/delete a device.

5.1.2. Interfaces

In this section, the interfaces (i.e., the “Service API” and the “Events API”) of the Automation component are explained in detail.

- **Automation Service API**

  Table 17 displays an overview of the RPC methods exposed by the Automation component. Specifically, the main RPC methods provide a way to automatically (i) onboard a new device (i.e., ztpAdd), (ii) reconfigure an already onboarded device (i.e., ztpUpdate), and (iii) remove an onboarded device (i.e., ztpDelete) or all of the onboarded devices (i.e., ztpDeleteAll). In addition to those key functions, the Automation component also exposes two read-only RPCs that allow other TeraFlow OS components to access the current state of device roles associated to the various devices. Specifically, the Automation component allows querying a device role using a device role ID as input (i.e., ztpGetDeviceRole), or querying a list of device roles associated with a specific device ID (i.e., ZtpGetDeviceRolesByDeviceId).

Table 17: Service interface definition for the Automation component.

RPC Method Name	Parameters	Results
ztpGetDeviceRole	DeviceRoleId	DeviceRole
ZtpGetDeviceRolesByDeviceId	DeviceId	List<DeviceRole>
ztpAdd	DeviceRole	DeviceRoleState
ztpUpdate	DeviceRole, DeviceConfig	DeviceRoleState
ztpDelete	DeviceRole	DeviceRoleState
ZtpDeleteAll	-	List<DeviceDeletionResult>

- **Automation Events API**

Apart from the main Automation services, the Automation component exploits a publish-subscribe TeraFlow OS mechanism to dynamically associate components with relevant events that require immediate actions. This is the role of the "Events' API". The Automation component relates its services with three basic events, as shown in Table 18.

Table 18: Events' publish-subscribe interface for the Automation component.

Event Name	**Triggered by**	**Triggers**	**Results**
DEVICE_ADD	Context component	Automation component ztpAdd RPC	New Device object with an associated: • DeviceStatus = ENABLED • DeviceRoleState = ZTP_DEV_STATE_CREATED
DEVICE_UPDATE	Policy component (or external entity)	Automation component ztpUpdate RPC	Updated Device object with an associated: • DeviceRoleState = ZTP_DEV_STATE_UPDATED
DEVICE_DELETE	Policy component (or external entity)	Automation component ztpDelete RPC	Updated Device object with an associated: • DeviceStatus = DISABLED • DeviceRoleState = ZTP_DEV_STATE_DELETED

5.1.3. Operational Workflows

In this section, detailed sequence diagrams are provided for the most important RPCs of the Automation component, also highlighting the interaction of the Automation component with other core TeraFlow OS components or external entities.

Automated device onboarding workflow

Upon issuing a policy to add a new device, the Policy component calls the Device component's AddDevice method. This method initiates a connection with the requested device, and if successfully connected, the device driver obtains the device configuration. This configuration is turned into a DeviceConfig object and pushed to the Context database. Upon success, a "DEVICE_ADD" event is generated, notifying that a new device is associated with a TeraFlow OS device driver plugin. As a result, the Context component generates a notification through the "Events API". This event is received by the Automation component, thus the ztpAdd RPC is automatically triggered as shown in Figure 23. First, the Automation component requests the new Device object from the Context database, which in turn results in a "getDevice" call to the Device component. Then, if this device is not already configured, the Automation component requests this device's initial configuration parameters by issuing a "getInitialConfig" RPC to the Device component through the Context component. Upon the receipt of an updated DeviceConfig object, the Automation component loops through the configuration entries of its local object and updates the relevant entries according to the newly fetched DeviceConfig object. Next, the updated device object is pushed to the Device component and stored to the Context database via a "configureDevice" RPC. Upon success, the Automation component flips the DeviceStatus bit of the Device object to "ENABLED" and the respective ZTPDeviceState to "CREATED", while also generating relevant events. Finally, these events can be consumed by other TeraFlow OS components e.g., to begin monitoring routines for this device.

Note that, if the DeviceStatus of the newly arrived device is already ENABLED, the device is already provisioned, thus no action is taken by the ztpAdd RPC, while a relevant warning is issued.

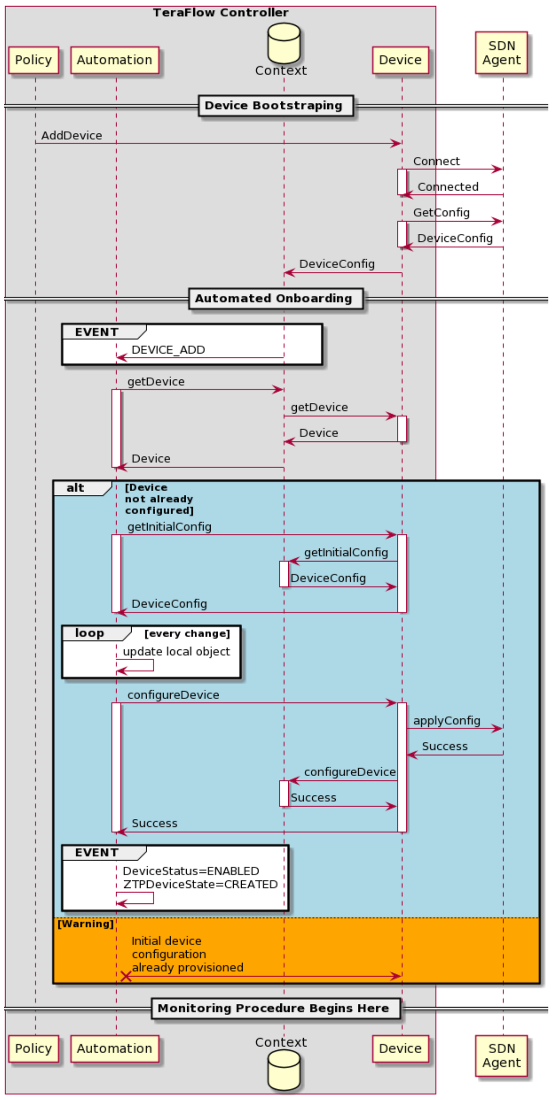


Figure 23: Zero-Touch Provisioning of a new device into TeraFlow OS (ztpAdd RPC).

Automated device update workflow

Upon a "DEVICE_UPDATE" event, e.g., received by the Policy component or an external entity, the Automation component receives a new candidate DeviceConfig object for a certain device, thus the ztpUpdate RPC is automatically triggered as shown in Figure 24. First, the Automation component requests the respective Device object from the Context database, which in turn results in a "getDevice" call to the Device component. Then, if this device is enabled, the Automation component loops through the configuration entries of its local object and updates the relevant entries according to the newly fetched DeviceConfig object. Next, the updated Device object is pushed to the Device component and stored to the Context database via a "configureDevice" RPC. Upon success, the Automation component flips the ZTPDeviceState to "UPDATED", while also generating a relevant event. Note that, if the device to be updated is not enabled, the ztpUpdate RPC outputs an error message indicating inability to update a disabled device.

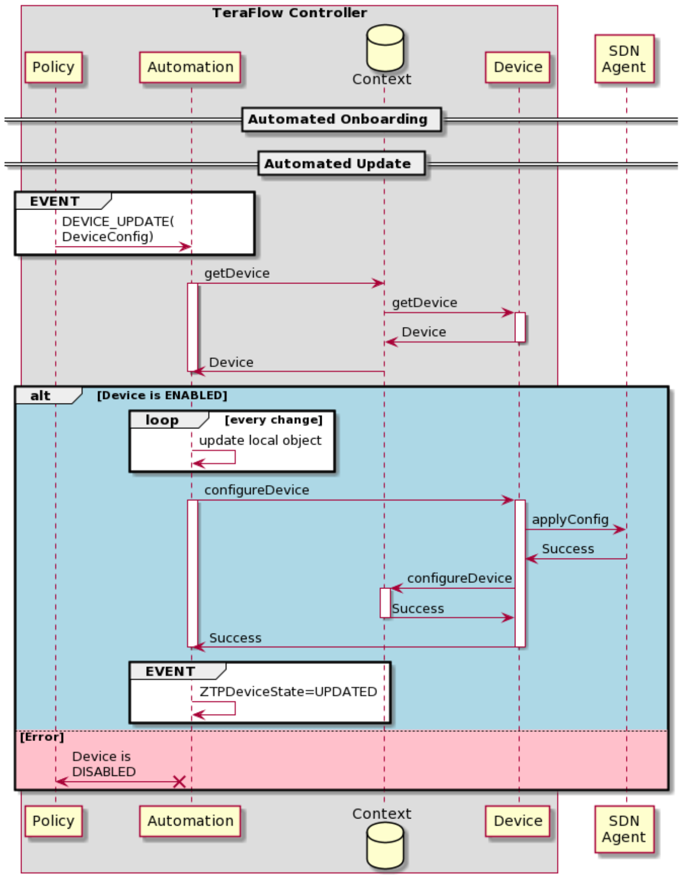


Figure 24: Zero-Touch Update of a device into TeraFlow OS (ztpUpdate RPC).

Automated device deletion workflow

Upon a “DEVICE_DELETE” event, e.g., received by the Policy component or an external entity, the Automation component receives a DeviceId object for provisional deletion, thus the ztpDelete RPC is automatically triggered as shown in Figure 25. First, the Automation component requests the respective Device object from the Context database, which in turn results in a “getDevice” call to the Device component. Then, if this device is enabled, the Device object is deleted from the Context database via a “deleteDevice” RPC issued to the Device and Context components in turn. Upon success, the Automation component flips the DeviceStatus bit of the Device object to “DISABLED” and the respective ZTPDeviceState to “DELETED”, while also generating relevant events. Note that, if the device to be deleted is not enabled, the ztpDelete RPC outputs an error message indicating inability to delete an already disabled device.

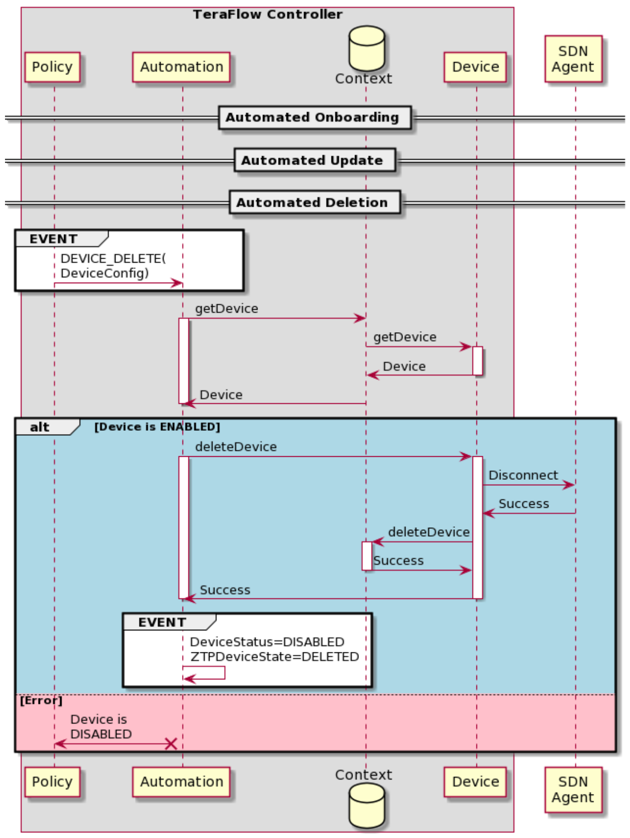
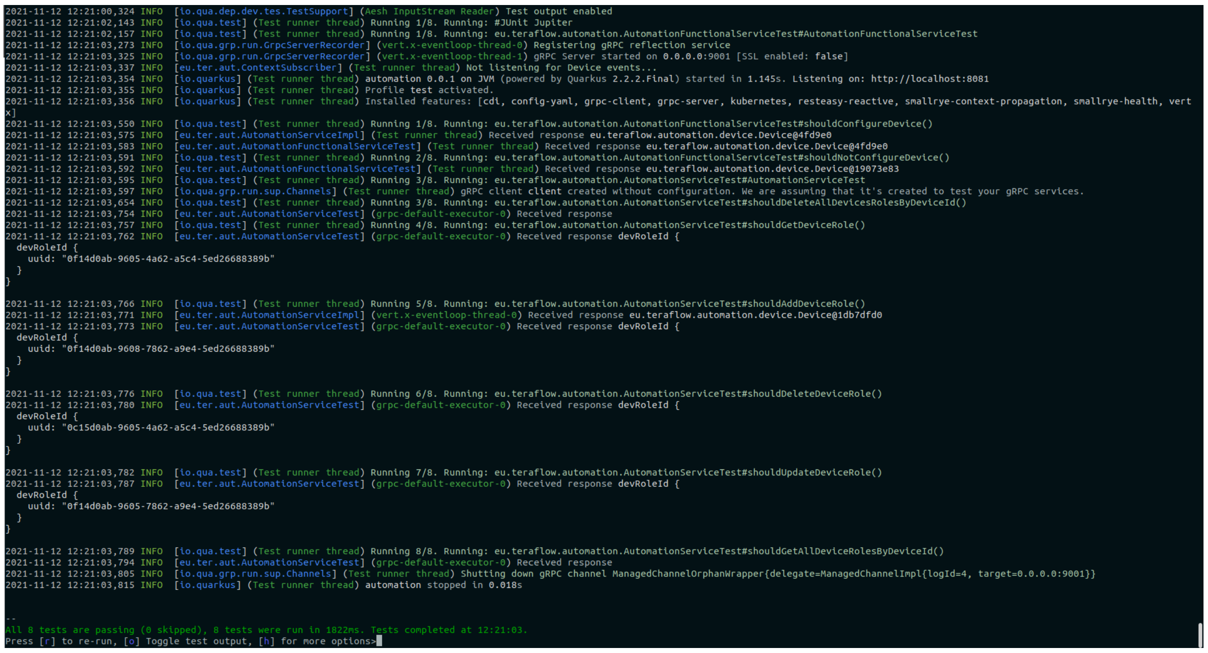


Figure 25: Zero-Touch Deletion of a device from TeraFlow OS (ztpDelete RPC).

5.1.4. Preliminary Results

In this section we show preliminary results stemming from the development of the Automation component. These results focus on three aspects, namely (i) the realization of unit tests for testing internal processes of the Automation component, (ii) the correct spawning of the Automation server offering automation services to the TeraFlow OS, and (iii) an example invocation of a key automation service (i.e., ztpAdd) which automatically adds a new device in the network.

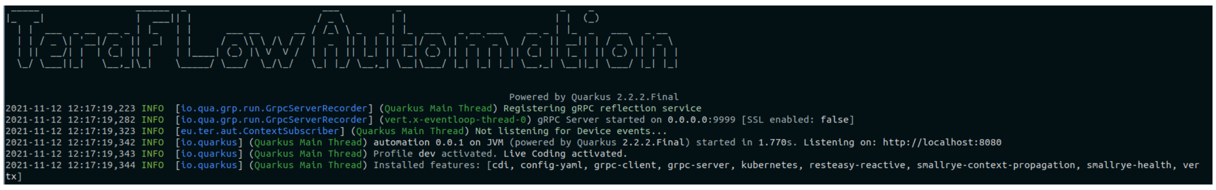
Figure 26 shows the successful outcome of eight unit tests that were developed for the Automation component in the course of the first TeraFlow OS release.



```
2021-11-12 12:21:00,324 INFO  [io.qua.dep.dev.tes.TestSupport] (Aesh InputStream Reader) Test output enabled
2021-11-12 12:21:02,143 INFO  [io.qua.test] (Test runner thread) Running 1/8. Running: #JUnit Jupiter
2021-11-12 12:21:02,157 INFO  [io.qua.test] (Test runner thread) Running 1/8. Running: eu.teraflow.automation.AutomationFunctionalServiceTest#AutomationFunctionalServiceTest
2021-11-12 12:21:03,273 INFO  [io.qua.grp.run.GrpcServerRecorder] (vert.x-eventloop-thread-0) Registering gRPC reflection service
2021-11-12 12:21:03,325 INFO  [io.qua.grp.run.GrpcServerRecorder] (vert.x-eventloop-thread-1) gRPC Server started on 0.0.0.0:9001 [SSL enabled: false]
2021-11-12 12:21:03,337 INFO  [eu.ter.aut.ContextSubscriber] (Test runner thread) Not listening for Device events...
2021-11-12 12:21:03,354 INFO  [io.quarkus] (Test runner thread) automation 0.0.1 on JVM (powered by Quarkus 2.2.2.Final) started in 1.145s. Listening on: http://localhost:8081
2021-11-12 12:21:03,355 INFO  [io.quarkus] (Test runner thread) Profile test activated.
2021-11-12 12:21:03,356 INFO  [io.quarkus] (Test runner thread) Installed features: [cdi, config-yaml, grpc-client, grpc-server, kubernetes, resteasy-reactive, smallrye-context-propagation, smallrye-health, vert
x]
2021-11-12 12:21:03,550 INFO  [io.qua.test] (Test runner thread) Running 1/8. Running: eu.teraflow.automation.AutomationFunctionalServiceTest#shouldConfigureDevice()
2021-11-12 12:21:03,575 INFO  [eu.ter.aut.AutomationServiceImpl] (Test runner thread) Received response eu.teraflow.automation.device.Device@4fd9e0
2021-11-12 12:21:03,583 INFO  [eu.ter.aut.AutomationFunctionalServiceTest] (Test runner thread) Received response eu.teraflow.automation.device.Device@4fd9e0
2021-11-12 12:21:03,591 INFO  [io.qua.test] (Test runner thread) Running 2/8. Running: eu.teraflow.automation.AutomationFunctionalServiceTest#shouldNotConfigureDevice()
2021-11-12 12:21:03,592 INFO  [eu.ter.aut.AutomationFunctionalServiceTest] (Test runner thread) Received response eu.teraflow.automation.device.Device@19073e83
2021-11-12 12:21:03,595 INFO  [io.qua.test] (Test runner thread) Running 3/8. Running: eu.teraflow.automation.AutomationServiceTest#AutomationServiceTest
2021-11-12 12:21:03,597 INFO  [io.qua.grp.run.sup.Channels] (Test runner thread) gRPC client client created without configuration. We are assuming that it's created to test your gRPC services.
2021-11-12 12:21:03,654 INFO  [io.qua.test] (Test runner thread) Running 3/8. Running: eu.teraflow.automation.AutomationServiceTest#shouldDeleteAllDevicesRolesByDeviceId()
2021-11-12 12:21:03,754 INFO  [eu.ter.aut.AutomationServiceTest] (grpc-default-executor-0) Received response
2021-11-12 12:21:03,757 INFO  [io.qua.test] (Test runner thread) Running 4/8. Running: eu.teraflow.automation.AutomationServiceTest#shouldGetDeviceRole()
2021-11-12 12:21:03,762 INFO  [eu.ter.aut.AutomationServiceTest] (grpc-default-executor-0) Received response devRoleId {
  devRoleId {
    uuid: "0f14d0ab-9605-4a62-a5c4-5ed26688389b"
  }
}

2021-11-12 12:21:03,766 INFO  [io.qua.test] (Test runner thread) Running 5/8. Running: eu.teraflow.automation.AutomationServiceTest#shouldAddDeviceRole()
2021-11-12 12:21:03,771 INFO  [eu.ter.aut.AutomationServiceImpl] (vert.x-eventloop-thread-0) Received response eu.teraflow.automation.device.Device@1db7dfd0
2021-11-12 12:21:03,773 INFO  [eu.ter.aut.AutomationServiceTest] (grpc-default-executor-0) Received response devRoleId {
  devRoleId {
    uuid: "0f14d0ab-9608-7862-a9e4-5ed26688389b"
  }
}

2021-11-12 12:21:03,776 INFO  [io.qua.test] (Test runner thread) Running 6/8. Running: eu.teraflow.automation.AutomationServiceTest#shouldDeleteDeviceRole()
2021-11-12 12:21:03,780 INFO  [eu.ter.aut.AutomationServiceTest] (grpc-default-executor-0) Received response devRoleId {
  devRoleId {
    uuid: "0c15d0ab-9605-4a62-a5c4-5ed26688389b"
  }
}

2021-11-12 12:21:03,782 INFO  [io.qua.test] (Test runner thread) Running 7/8. Running: eu.teraflow.automation.AutomationServiceTest#shouldUpdateDeviceRole()
2021-11-12 12:21:03,787 INFO  [eu.ter.aut.AutomationServiceTest] (grpc-default-executor-0) Received response devRoleId {
  devRoleId {
    uuid: "0f14d0ab-9605-7862-a9e4-5ed26688389b"
  }
}

2021-11-12 12:21:03,789 INFO  [io.qua.test] (Test runner thread) Running 8/8. Running: eu.teraflow.automation.AutomationServiceTest#shouldGetAllDeviceRolesByDeviceId()
2021-11-12 12:21:03,794 INFO  [eu.ter.aut.AutomationServiceTest] (grpc-default-executor-0) Received response
2021-11-12 12:21:03,805 INFO  [io.qua.grp.run.sup.Channels] (Test runner thread) Shutting down gRPC channel ManagedChannelOrphanWrapper{delegate=ManagedChannelImpl{logId=4, target=0.0.0.0:9001}}
2021-11-12 12:21:03,815 INFO  [io.quarkus] (Test runner thread) automation stopped in 0.018s


--
All 8 tests are passing (0 skipped), 8 tests were run in 1822ms. Tests completed at 12:21:03.
Press [r] to re-run, [o] Toggle test output, [h] for more options>
```

Figure 26: Unit tests for the Automation TeraFlow OS component.

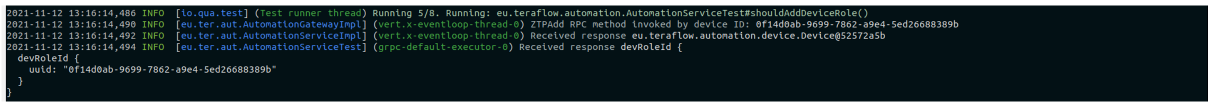
Once all unit tests have passed successfully, the Automation component can be deployed. Figure 27 shows the output of the gRPC Automation server upon the deployment of this component. The Automation component is implemented in Java using the Apache Quarkus framework. From the logs in Figure 27, we see that the Automation server is successfully started, listening on port 9999.



```
                                                                Powered by Quarkus 2.2.2.Final
2021-11-12 12:17:19,223 INFO  [io.qua.grp.run.GrpcServerRecorder] (Quarkus Main Thread) Registering gRPC reflection service
2021-11-12 12:17:19,282 INFO  [io.qua.grp.run.GrpcServerRecorder] (vert.x-eventloop-thread-0) gRPC Server started on 0.0.0.0:9999 [SSL enabled: false]
2021-11-12 12:17:19,323 INFO  [eu.ter.aut.ContextSubscriber] (Quarkus Main Thread) Not listening for Device events...
2021-11-12 12:17:19,342 INFO  [io.quarkus] (Quarkus Main Thread) automation 0.0.1 on JVM (powered by Quarkus 2.2.2.Final) started in 1.770s. Listening on: http://localhost:8080
2021-11-12 12:17:19,343 INFO  [io.quarkus] (Quarkus Main Thread) Profile dev activated. Live Coding activated.
2021-11-12 12:17:19,344 INFO  [io.quarkus] (Quarkus Main Thread) Installed features: [cdi, config-yaml, grpc-client, grpc-server, kubernetes, resteasy-reactive, smallrye-context-propagation, smallrye-health, ver
tx]
```

Figure 27: Instantiation of the Automation gRPC server.

To add a new device in the network, the ztpAdd method of the Automation component should be invoked. Figure 28 shows such an example invocation for a device with ID 0f14d0ab-9699-7862-a9e4-5ed26688389b.



```
2021-11-12 13:16:14,486 INFO  [io.qua.test] (Test runner thread) Running 5/8. Running: eu.teraflow.automation.AutomationServiceTest#shouldAddDeviceRole()
2021-11-12 13:16:14,490 INFO  [eu.ter.aut.AutomationGatewayImpl] (vert.x-eventloop-thread-0) ZTPAdd RPC method invoked by device ID: 0f14d0ab-9699-7862-a9e4-5ed26688389b
2021-11-12 13:16:14,492 INFO  [eu.ter.aut.AutomationServiceImpl] (vert.x-eventloop-thread-0) Received response eu.teraflow.automation.device.Device@52572a5b
2021-11-12 13:16:14,494 INFO  [eu.ter.aut.AutomationServiceTest] (grpc-default-executor-0) Received response devRoleId {
  devRoleId {
    uuid: "0f14d0ab-9699-7862-a9e4-5ed26688389b"
  }
}
```

Figure 28: Invocation of the ztpAdd gRPC method for adding a new device in the network.

5.2. Policy Management Component

Network policy may be defined as a collection of rules that dictate the behaviours of network resources, which may include devices (physical or virtual) and functional components. Within the TeraFlow project, we use policy rules for both:

- high-level objectives, which are often referred to as “intent” statements due to the declarative nature of the request.

- low-level objectives, applied to specific devices or when making network resource assignment decisions. With often use imperative statements when processing “network policy”.

Our focus on the initial design of the TeraFlow Policy Management is to use “event-driven management” [2]. Event-driven management provides a valuable method to monitor state change of managed objects and resources and enable automatic triggering of responses to events based on an established set of rules. The TeraFlow event-driven policy provides rapid autonomic responses to specific conditions, enabling self-management behaviours, such as self-configuration, self-healing, self-optimization, and self-protection.

The TeraFlow Policy Management Component utilises an emerging technical technique called “Event Condition Action” (ECA) to provide event-driven benefits [2]. ECA Policy enables actions to be automatically triggered based on when certain events in the network occur while certain conditions hold. Thus, ECA facilitates limited logic to be delegated to network devices and functional components for automating specific required behaviour.

5.2.1. Design Overview

The goal of the Policy Management component, also abbreviated as Policy component, is to translate a network operator’s high-level network policy statements into a correct set of low-level instructions that realize this policy across the various network elements. To meet this objective, the Policy component is designed according to Figure 29.

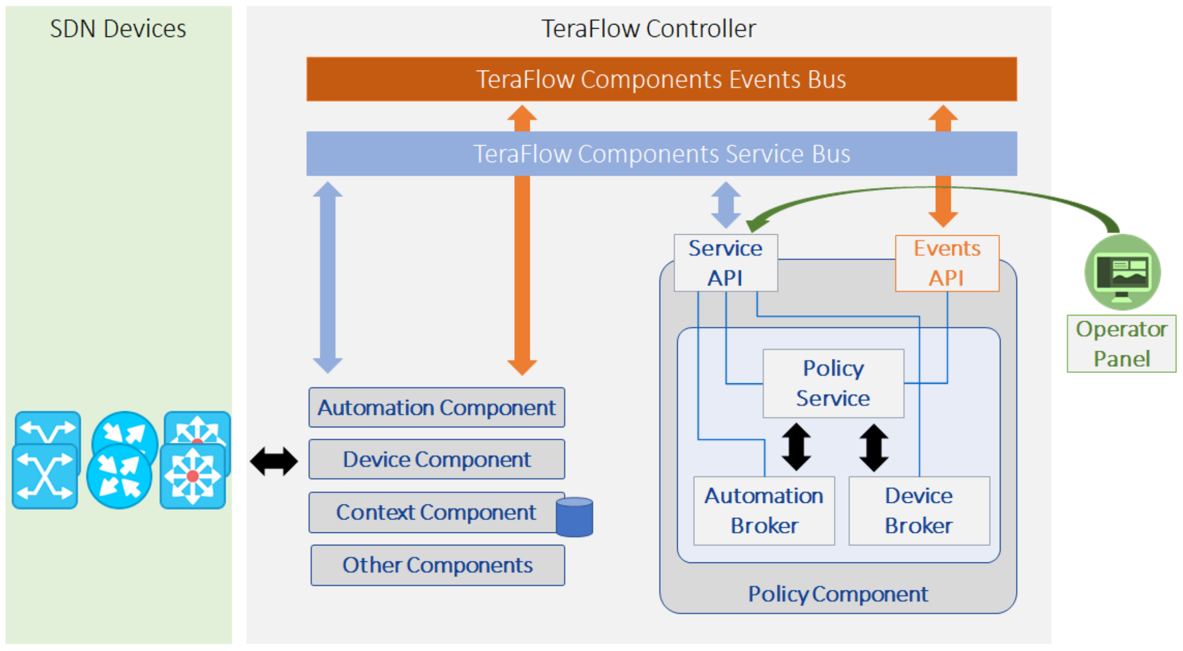


Figure 29: Policy component overview.

Northbound Interfaces: The policy component presents two NBIs. The first interface, titled “Service API” in Figure 29, exposes basic policy functions to the rest of the TeraFlow OS components as well as network operator panels connected to the TeraFlow OS. The second interface, titled “Events API” in Figure 29, allows the Policy component to register to receive, and thus react upon, relevant events from key TeraFlow OS components. Both interfaces are described in Section 5.2.2.

Core logic: Upon receiving an incoming request from either of the northbound interfaces, the Policy service (shown in Figure 29) realizes the core logic of the Policy component. First, for any new policies provided by an external entity, the Policy component's core loop goes through the respective policy events and subscribes to those events through the Events API. This way the Policy component will be automatically triggered when such an event occurs, which will in turn trigger the policy enforcement process. When such an event is triggered, the received notification (through the Event API) reaches the Policy Service block and the triggered policy is identified. To apply this policy though, the ECA model requires a policy condition to be met. Finally, once the condition is met, the Policy Service block will apply a set of actions associated with this policy. Such list of actions could affect either a single device (i.e., for device-level policy type) or a set of devices (i.e., network-wide policy type). To exemplify an ECA policy, one may consider the following example of a monitoring policy:

- Event: Device with ID X is enabled (i.e., successfully finished bootstrapping and initial configuration)
- Condition: Device with ID X is not monitored
- Action: Start monitoring of certain KPIs on device with ID X.

5.2.2. Interfaces

In this section, the interfaces (i.e., the "Service API" and the "Events API") of the Policy component are explained in detail.

- **Policy Service API**

Table 19 displays an overview of the RPC methods exposed by the Policy component. Specifically, the main RPC methods allow a client to (i) express and add a new policy (i.e., policyAdd), (ii) update an already applied policy (i.e., policyUpdate), and (iii) revoke an applied policy (i.e., policyDelete). In addition to those key functions, the Policy component also exposes three read-only RPCs that allow other TeraFlow OS components to access the current state of policies associated to the various devices or services. Specifically, the Policy component allows a client to query a single policy rule using a policy rule ID (GetPolicy), or to query a list of policy rules by device ID (i.e., GetPolicyByDeviceId) or service ID (i.e., GetPolicyByServiceId).

Table 19: Service interface definition for the Policy component.

RPC Method Name	**Parameters**	**Results**
PolicyAdd	PolicyRule	PolicyRuleState
PolicyUpdate	PolicyRule	PolicyRuleState
PolicyDelete	PolicyRule	PolicyRuleState
GetPolicy	PolicyRuleId	PolicyRule
GetPolicyByDeviceId	DeviceId	PolicyRuleList
GetPolicyByServiceId	ServiceId	PolicyRuleList

- **Policy Events API**

Apart from the main Policy services, the Policy component exploits a publish-subscribe TeraFlow OS mechanism to dynamically associate components with relevant events that require immediate actions. This is the role of the "Events API". The current implementation of the Policy component does not yet provide support for this API, this is work planned through the second year of the TeraFlow project.

5.2.3. Operational Workflows

In this section, a detailed sequence diagram is provided for the most important RPC of the Policy management component, i.e., the creation of a new policy. This highlights both the interaction of the Policy component with other core TeraFlow OS components or external entities, as well as the prominent effect of the ECA model in the design of the Policy component.

Policy add workflow

To apply a new policy to the network, a network operator needs to trigger the policyAdd RPC of the Policy management component as shown in Figure 30. This call will provide the Policy component with a Policy rule object which contains a triplet of internal objects, namely an event, a set of conditions, and a set of actions. First, the Policy component parses the received policy, extracts the policy event, and registers to this event via the Events API. When this event is triggered, a notification is received by the Policy component, thus the given policy should be processed. This involves a check on whether the given policy conditions are met or not. For this to happen, the Policy component executes a loop across all conditions and marks a flag (i.e., ready = True) when a condition is met, otherwise ready=False. If and only if all conditions are met, is the action part of the policy rule applied. For each action in a list of actions, the Policy component fetches the affected device from the Context component's database, applies the action locally, and then invokes the Automation component's ztpUpdate RPC to enforce this policy to the device (see Section 5.1.2 above). Upon a successful policy enforcement, a relevant event is generated, otherwise an error is sent back to the operator signifying the inability of the Policy component to apply this policy.

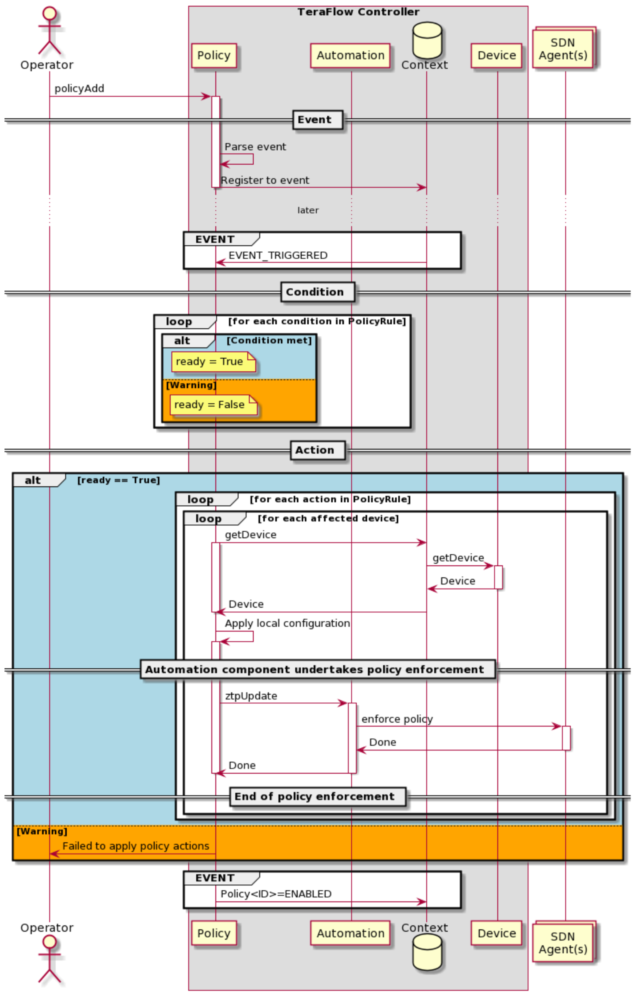
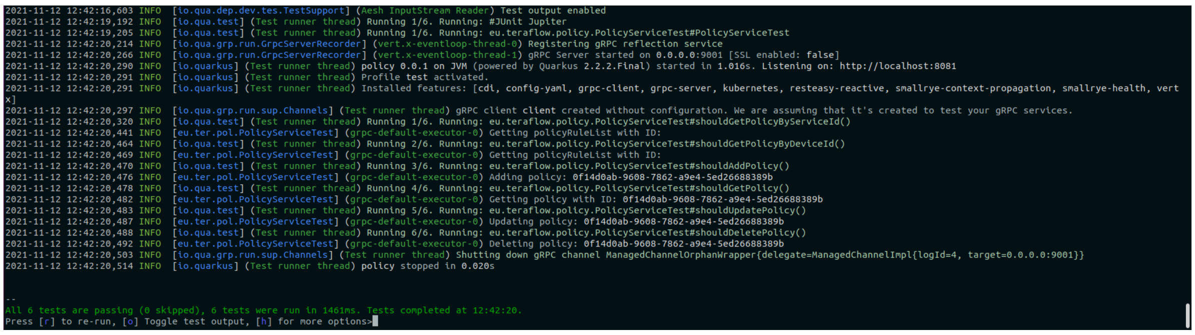


Figure 30: Generic Policy creation by an operator through the TeraFlow OS policyAdd RPC.

5.2.4. Preliminary Results

In this section we show preliminary results stemming from the preliminary development of the Policy Management component. These results focus on two aspects, namely (i) the realization of unit tests for testing internal processes of the Policy Management component, and (ii) the correct spawning of the Policy server offering policy management services to the TeraFlow OS.

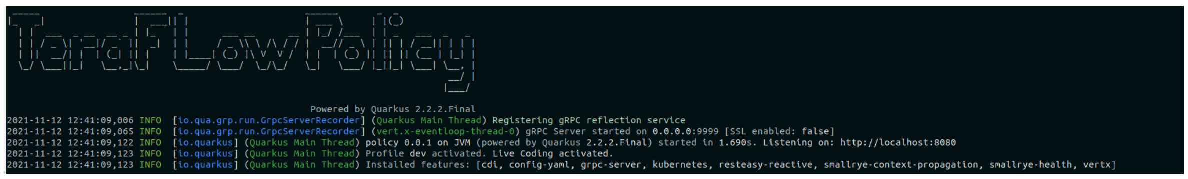
Figure 31 shows the successful outcome of six unit tests that were developed for the Policy Management component during the first TeraFlow OS release.



```
2021-11-12 12:42:16,603 INFO  [io.qua.dep.dev.tes.TestSupport] (Aesh InputStream Reader) Test output enabled
2021-11-12 12:42:19,192 INFO  [io.qua.test] (Test runner thread) Running 1/6. Running: #JUnit Jupiter
2021-11-12 12:42:19,205 INFO  [io.qua.test] (Test runner thread) Running 1/6. Running: eu.teraflow.policy.PolicyServiceTest#PolicyServiceTest
2021-11-12 12:42:20,214 INFO  [io.qua.grp.run.GrpcServerRecorder] (vert.x-eventloop-thread-0) Registering gRPC reflection service
2021-11-12 12:42:20,266 INFO  [io.qua.grp.run.GrpcServerRecorder] (vert.x-eventloop-thread-1) gRPC Server started on 0.0.0.0:9001 [SSL enabled: false]
2021-11-12 12:42:20,290 INFO  [io.quarkus] (Test runner thread) policy 0.0.1 on JVM (powered by Quarkus 2.2.2.Final) started in 1.016s. Listening on: http://localhost:8081
2021-11-12 12:42:20,291 INFO  [io.quarkus] (Test runner thread) Profile test activated.
2021-11-12 12:42:20,291 INFO  [io.quarkus] (Test runner thread) Installed features: [cdi, config-yaml, grpc-client, grpc-server, kubernetes, resteasy-reactive, smallrye-context-propagation, smallrye-health, vert
x]
2021-11-12 12:42:20,297 INFO  [io.qua.grp.run.sup.Channels] (Test runner thread) gRPC client client created without configuration. We are assuming that it's created to test your gRPC services.
2021-11-12 12:42:20,320 INFO  [io.qua.test] (Test runner thread) Running 1/6. Running: eu.teraflow.policy.PolicyServiceTest#shouldGetPolicyByServiceId()
2021-11-12 12:42:20,441 INFO  [eu.ter.pol.PolicyServiceTest] (grpc-default-executor-0) Getting policyRuleList with ID:
2021-11-12 12:42:20,464 INFO  [io.qua.test] (Test runner thread) Running 2/6. Running: eu.teraflow.policy.PolicyServiceTest#shouldGetPolicyByDeviceId()
2021-11-12 12:42:20,469 INFO  [eu.ter.pol.PolicyServiceTest] (grpc-default-executor-0) Getting policyRuleList with ID:
2021-11-12 12:42:20,470 INFO  [io.qua.test] (Test runner thread) Running 3/6. Running: eu.teraflow.policy.PolicyServiceTest#shouldAddPolicy()
2021-11-12 12:42:20,476 INFO  [eu.ter.pol.PolicyServiceTest] (grpc-default-executor-0) Adding policy: 0f14d0ab-9608-7862-a9e4-5ed26688389b
2021-11-12 12:42:20,478 INFO  [io.qua.test] (Test runner thread) Running 4/6. Running: eu.teraflow.policy.PolicyServiceTest#shouldGetPolicy()
2021-11-12 12:42:20,482 INFO  [eu.ter.pol.PolicyServiceTest] (grpc-default-executor-0) Getting policy with ID: 0f14d0ab-9608-7862-a9e4-5ed26688389b
2021-11-12 12:42:20,483 INFO  [io.qua.test] (Test runner thread) Running 5/6. Running: eu.teraflow.policy.PolicyServiceTest#shouldUpdatePolicy()
2021-11-12 12:42:20,487 INFO  [eu.ter.pol.PolicyServiceTest] (grpc-default-executor-0) Updating policy: 0f14d0ab-9608-7862-a9e4-5ed26688389b
2021-11-12 12:42:20,488 INFO  [io.qua.test] (Test runner thread) Running 6/6. Running: eu.teraflow.policy.PolicyServiceTest#shouldDeletePolicy()
2021-11-12 12:42:20,492 INFO  [eu.ter.pol.PolicyServiceTest] (grpc-default-executor-0) Deleting policy: 0f14d0ab-9608-7862-a9e4-5ed26688389b
2021-11-12 12:42:20,503 INFO  [io.qua.grp.run.sup.Channels] (Test runner thread) Shutting down gRPC channel ManagedChannelOrphanWrapper{delegate=ManagedChannelImpl{logId=4, target=0.0.0.0:9001}}
2021-11-12 12:42:20,514 INFO  [io.quarkus] (Test runner thread) policy stopped in 0.020s


--
All 6 tests are passing (0 skipped), 6 tests were run in 1461ms. Tests completed at 12:42:20.
Press [r] to re-run, [o] Toggle test output, [h] for more options>
```

Figure 31: Unit tests for the Policy Management TeraFlow OS component.

Once all unit tests are successfully passed, the Policy component can be deployed. Figure 32 shows the output of the gRPC Policy server upon the deployment of this component. Like the Automation component, the Policy component is also implemented in Java using the Apache Quarkus framework. From the logs in Figure 32, we see that the Policy server is successfully started, listening on port 8080.



```
                                        Powered by Quarkus 2.2.2.Final
2021-11-12 12:41:09,006 INFO  [io.qua.grp.run.GrpcServerRecorder] (Quarkus Main Thread) Registering gRPC reflection service
2021-11-12 12:41:09,065 INFO  [io.qua.grp.run.GrpcServerRecorder] (vert.x-eventloop-thread-0) gRPC Server started on 0.0.0.0:9999 [SSL enabled: false]
2021-11-12 12:41:09,122 INFO  [io.quarkus] (Quarkus Main Thread) policy 0.0.1 on JVM (powered by Quarkus 2.2.2.Final) started in 1.690s. Listening on: http://localhost:8080
2021-11-12 12:41:09,123 INFO  [io.quarkus] (Quarkus Main Thread) Profile dev activated. Live Coding activated.
2021-11-12 12:41:09,123 INFO  [io.quarkus] (Quarkus Main Thread) Installed features: [cdi, config-yaml, grpc-server, kubernetes, resteasy-reactive, smallrye-context-propagation, smallrye-health, vertx]
```

Figure 32: Instantiation of the Policy Management gRPC server.

6. Transport Network Slicing and Multi-tenancy

This section provides a design overview, the northbound & southbound interfaces, and preliminary results of the core TeraFlow OS components of T3.4, i.e., the Slice Management component (see Section 6.1). Prior to that, we clarify what a transport network slice means to TeraFlow.

Transport Network Slice definition

A transport network slice consists of a set of endpoints (e.g., CEs), a connectivity matrix between subsets of these endpoints, and service level behaviours requested for each sender on the connectivity matrix [11]. The connectivity between the endpoints might be point-to-point, point-to-multipoint, or multipoint-to-multipoint. Often these slices will be used to satisfy network behaviour defined in a Service Level Agreement (SLA) [11].

The SLA is an explicit or implicit contract between the customer and the provider of the slice. The terms of the SLA will be expressed in a set of Service Level Indicators (SLIs), Service Level Objectives (SLOs), and Service Level Expectations (SLEs), which are described below:

- Service Level Indicator
  A quantifiable measure of an aspect of the performance of a network. It might be a measure of throughput in bits per second or latency in milliseconds.
- Service Level Objective
  A target value or range for the measurements returned by observation of an SLI, expressed as "SLI <= target", or "lower bound <= SLI <= upper bound". For example, a customer may determine whether the provider is meeting the SLOs by measurements of the traffic.
- Service Level Expectation
  Often an expression of an unmeasurable service-related request that a customer makes of the provider. The customer has little way of determining whether the SLE is being met, but there is still a contract for a service that meets the expectation.

A customer may also be defined as a tenant, and each slice will be associated with a tenant and may control and operate the slices that have been provided. Furthermore, transport network slicing supports multi-tenancy so that lower-layer infrastructure (often referred to as the “underlay”) provides connectivity and resource guarantees to multiple tenants.

A transport slice may also be configured to be “soft” or “hard”. A “soft slice” might be a VLAN assigned to a tenant to packet data traffic – such as VoIP – usually transported by a layer-specific VPN (LxVPN) [12] over a shared underlay infrastructure. A “hard slice” provides a connectivity resource underpinned by dedicated resources that may be virtual or physical.

6.1. Slice Management Component

This section presents the design overview (Section 6.1.1), interfaces (Section 6.1.2), and preliminary results (Section 6.1.3) of the Slice Management component.

6.1.1. Design Overview

Transport network slicing is facilitated using an abstracted management architecture shown in Figure 33.

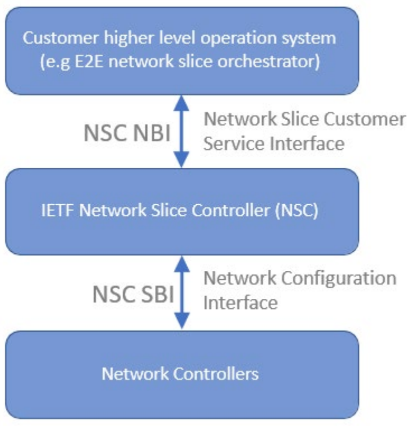

Customer higher level operation system
(e.g E2E network slice orchestrator)
NSC NBI
Network Slice Customer Service Interface
IETF Network Slice Controller (NSC)
NSC SBI
Network Configuration Interface
Network Controllers

Figure 33: Abstracted Management Architecture for Network Slicing

The TeraFlow slicing implementation uses the Network Slice Controller (NSC) to realise a transport network slice, using physical and virtual network resources provided by the underlying network controllers. In the TeraFlow project, these controllers manage both optical and packet resource domains.

A transport network slice request will be sent to the NSC by high-layer OSS components. The slice request will include a specific set of high-level requirements for transport connectivity parameters and behaviour; this may include bandwidth, latency and reliability, requirements.

The method of the slice request via the "Network Slice Customer Service Interface" (NBI) is not formally defined, but will typically use a model-driven protocol and slice template. For example, in the TeraFlow project, we use a YANG data model, which is encoded and sent to the NSC using the RESTCONF protocol.

Mapping the tenant transport network slice to the underlay resources and life-cycle management of the service is managed by the NSC. During the initial transport network slice request, numerous objectives may be requested, including:

- Disjoint paths to ensure slices are completely separated from other slices using Shared Risk Link Groups (SRLGs). Thus, slices do not share the same underlay network path, including physical or virtual resources.
- Specific SLA requirements for the service.

A further set of model-driven interfaces communicate between the NSC function and domain-specific network controllers via the "Network Configuration Interface" (SBI). Within the NSC, policy management plays an important role. For example, the NSC will instantiate slices based on the NBI request it receives, adhering to the SLA requirements and facilitating the transport network slice mapping to underlay network resources. Within TeraFlow additional YANG data model will be used to define this policy enforcement behaviour [2].

6.1.2. Interfaces

Several data models for IETF Transport Network Slices might be considered, but our first implementation and preliminary results will be obtained from [13].

An OSS/BSS may request that the deployed network slice is isolated from any other network slices or different services delivered to the same customers. Naturally, other network slices or services must not negatively impact the requested transport network slice's delivery. There are several possibilities to provide this isolation, which can be provided at several degrees, such as dedicated allocation of resources for a specific slice or sharing some network resources.

Figure 34 shows the multiple isolation options that range from a hard slice to a soft slice:

a) no-isolation, meaning that slices are not separated;
b) physical-isolation, where slices are completely physically separated, for example, in different locations;
c) logical-isolation, where slices are logically separated, only a certain degree of isolation is performed through QoS mechanisms;
d) process-isolation, where slices include process and threads isolation;
e) physical-network-isolation, where slices contain physically separated links;
f) virtual-resource-isolation, where slices have dedicated virtual resources;
g) network-functions-isolation, where Network Function (NF) are dedicated to a single network slice;
h) service-isolation, where virtual resources and NFs are shared.

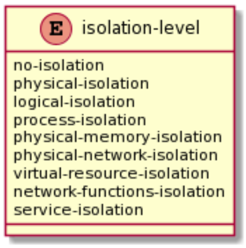

E
isolation-level
no-isolation
physical-isolation
logical-isolation
process-isolation
physical-memory-isolation
physical-network-isolation
virtual-resource-isolation
network-functions-isolation
service-isolation

Figure 34 IETF Transport Network Slice Isolation Levels

6.1.3. Preliminary Results

This section provides preliminary results obtained with the support of an ABNO orchestrator to trigger the multiple interfaces to deploy hard isolated slices. These preliminary results have allowed us to decide the which data models to use for the TeraFlow OS implementation. They have been published in [14].

Figure 35 shows a preliminary network architecture for network slicing in TeraFlow. It consists of three main domains: SDN domain, IP domain, and optical domain. The Network Slice Controller (NSC) realizes a transport network slice in the underlying transport infrastructure, and maintains and monitors the state of its resources. The NSC will delegate to SDN Domain controllers to configure the network resources. The NSC receives a transport network slice request from the Operation Support System and Business Support System (OSS/BSS). The request is modelled using the YANG data model defined in [13] by means of the RESTCONF protocol. Internal workflow for transport network slice lifecycle management is prepared on top of L2/L3 service management (SM) workflows to interact with underlying IP and Optical Domain controllers via a RESTCONF client.

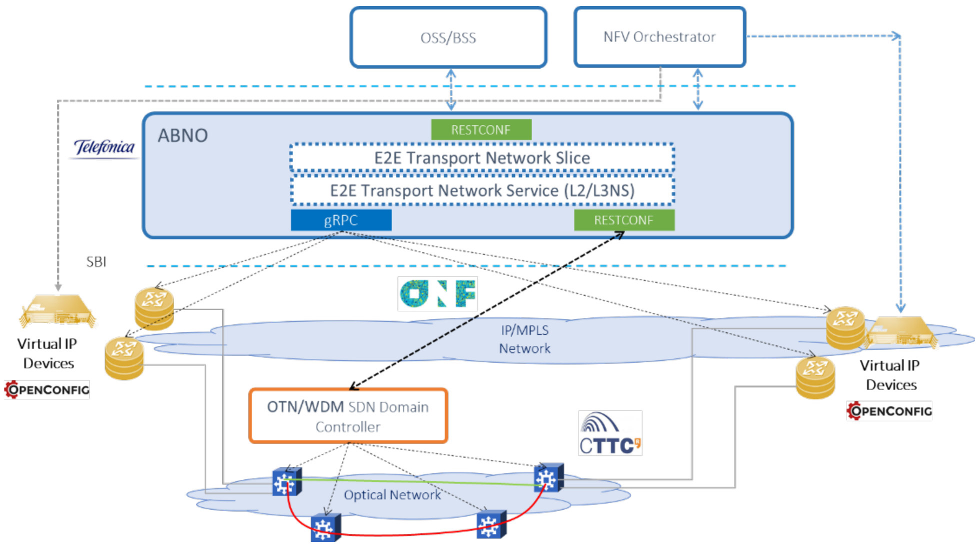


Figure 35 Slicing preliminary architecture

Figure 36 shows the workflow to deploy hard and soft transport network slices. In the workflow, two isolated network slices are deployed. The first one allocates a connectivity service to interconnect both IP layer domains. This triggers the necessary optical configuration mechanism to each of the underlying ROADMs. Once the connectivity service has been established, the NSC is responsible for requesting the IP SDN domain controller to provision the necessary virtual routers (in the proposed scenario two site-network-access are configured: one network instance on each site). Link Aggregation Control Protocol (LACP) is configured for each network instance. Then interfaces are aggregated and properly configured using dedicated VLAN or MPLS-TE mechanisms. When a new isolated slice is requested, the NSC can request a dedicated and isolated connectivity service from the underlying optical SDN controller. The connectivity constraint is provided using disjoint path selection with the ONF Transport API. It consists of including a diversity exclusion constraint with the previous connectivity service identifier. Later, at the IP layer, new virtual routers are deployed to provide the requested degree of isolation.

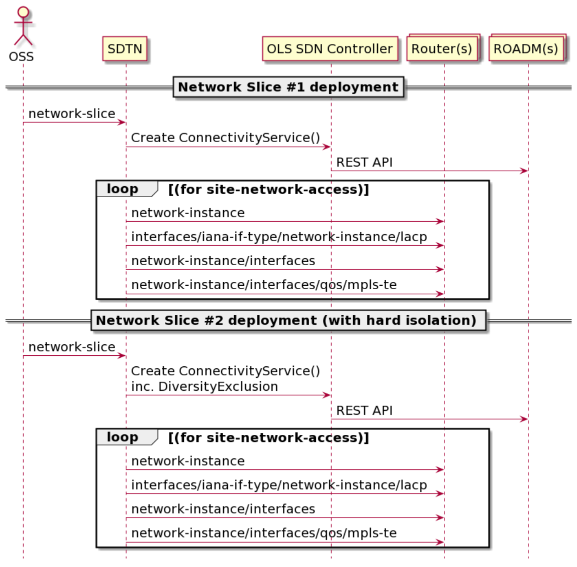
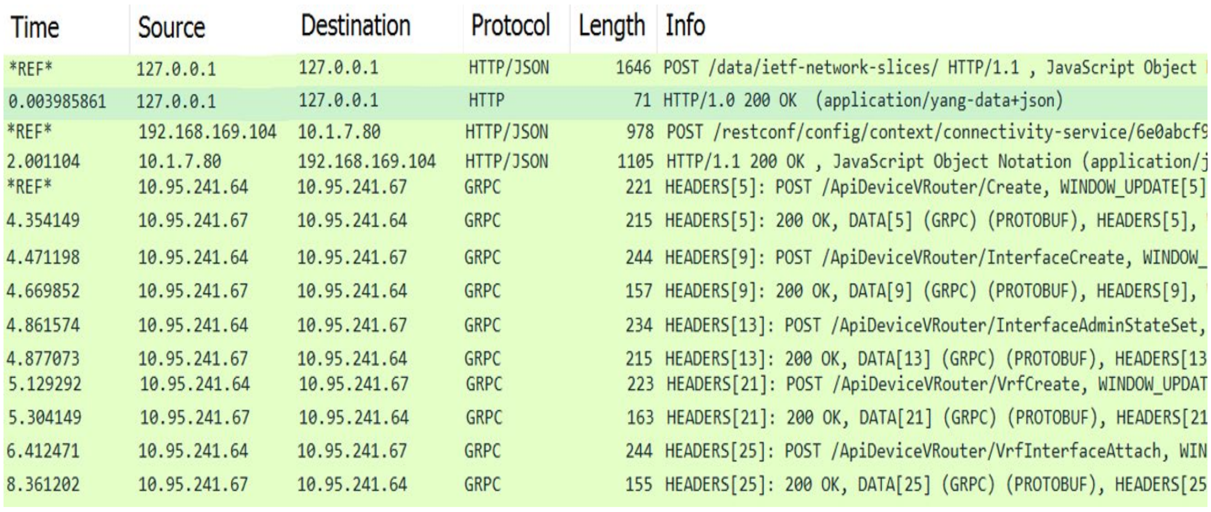

OSS
SDTN
OLS SDN Controller
Router(s)
ROADM(s)
Network Slice #1 deployment
network-slice
Create ConnectivityService()
REST API
loop [(for site-network-access)]
network-instance
interfaces/iana-if-type/network-instance/lacp
network-instance/interfaces
network-instance/interfaces/qos/mpls-te
Network Slice #2 deployment (with hard isolation)
network-slice
Create ConnectivityService()
inc. DiversityExclusion
REST API
loop [(for site-network-access)]
network-instance
interfaces/iana-if-type/network-instance/lacp
network-instance/interfaces
network-instance/interfaces/qos/mpls-te

Figure 36 Proposed sequency diagram

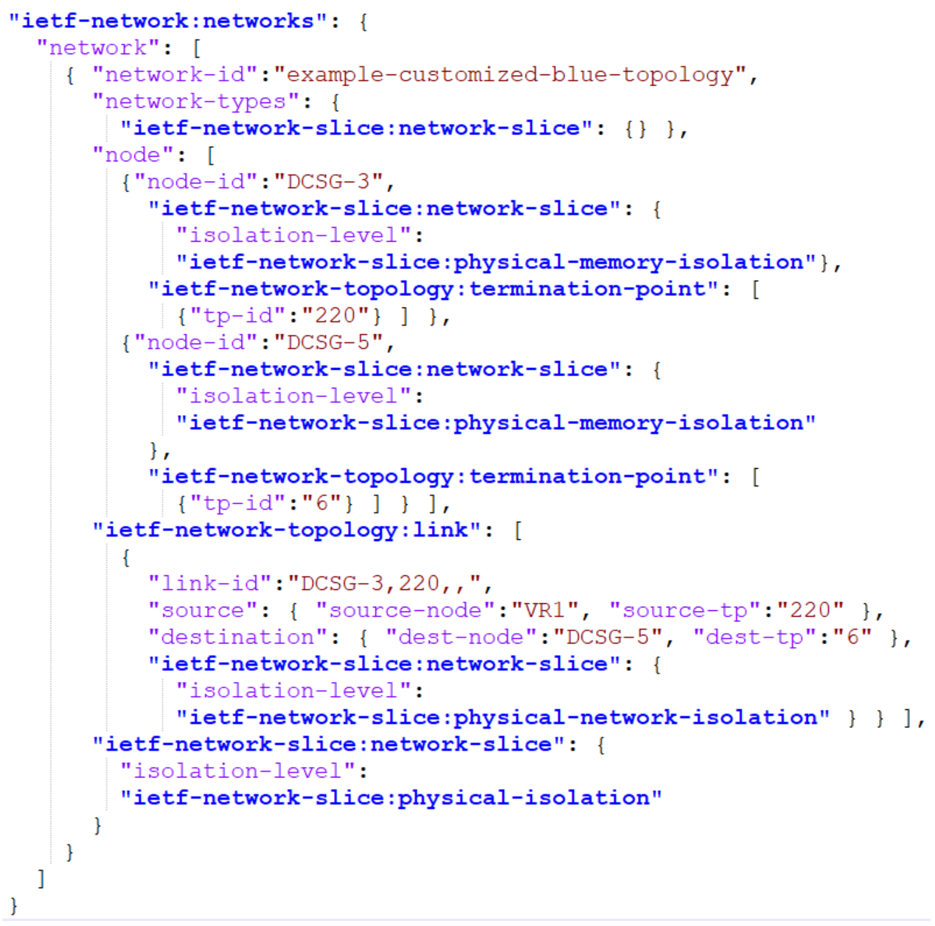
To validate the sequence diagram depicted in Figure 37, we provide the captured Wireshark traces from the complete system when deploying a complete hard slice request (the figure provides the selected significant traces). In this figure the different protocols can be observed. Firstly, it shows the request for the transport network slice. It is stored and answered at once, and it is processed after response has been sent (a later a status update might be requested).



Time	Source	Destination	Protocol	Length	Info
REF	127.0.0.1	127.0.0.1	HTTP/JSON	1646	POST /data/ietf-network-slices/ HTTP/1.1 , JavaScript Object
0.003985861	127.0.0.1	127.0.0.1	HTTP	71	HTTP/1.0 200 OK (application/yang-data+json)
REF	192.168.169.104	10.1.7.80	HTTP/JSON	978	POST /restconf/config/context/connectivity-service/6e0abcf9
2.001104	10.1.7.80	192.168.169.104	HTTP/JSON	1105	HTTP/1.1 200 OK , JavaScript Object Notation (application/j
REF	10.95.241.64	10.95.241.67	GRPC	221	HEADERS[5]: POST /ApiDeviceVRouter/Create, WINDOW_UPDATE[5]
4.354149	10.95.241.67	10.95.241.64	GRPC	215	HEADERS[5]: 200 OK, DATA[5] (GRPC) (PROTOBUF), HEADERS[5],
4.471198	10.95.241.64	10.95.241.67	GRPC	244	HEADERS[9]: POST /ApiDeviceVRouter/InterfaceCreate, WINDOW_
4.669852	10.95.241.67	10.95.241.64	GRPC	157	HEADERS[9]: 200 OK, DATA[9] (GRPC) (PROTOBUF), HEADERS[9],
4.861574	10.95.241.64	10.95.241.67	GRPC	234	HEADERS[13]: POST /ApiDeviceVRouter/InterfaceAdminStateSet,
4.877073	10.95.241.67	10.95.241.64	GRPC	215	HEADERS[13]: 200 OK, DATA[13] (GRPC) (PROTOBUF), HEADERS[13
5.129292	10.95.241.64	10.95.241.67	GRPC	223	HEADERS[21]: POST /ApiDeviceVRouter/VrfCreate, WINDOW_UPDAT
5.304149	10.95.241.67	10.95.241.64	GRPC	163	HEADERS[21]: 200 OK, DATA[21] (GRPC) (PROTOBUF), HEADERS[21
6.412471	10.95.241.64	10.95.241.67	GRPC	244	HEADERS[25]: POST /ApiDeviceVRouter/VrfInterfaceAttach, WIN
8.361202	10.95.241.67	10.95.241.64	GRPC	155	HEADERS[25]: 200 OK, DATA[25] (GRPC) (PROTOBUF), HEADERS[25

Figure 37 Wireshark captures for deployment of a hard slice

The transport network slice requests physical network isolation for the solicited link. Thus a T-API connectivity service is requested including as connectivity constraints the diversity exclusion option. For that, we provide the identifiers from the previously-established connectivity services, and we obtain a disjoint path for the new requested slice, resulting in the requested physical network isolation. It can be seen that the connectivity service setup time is 2.001s. Consider that the ADRENALINE testbed already has the paths equalized.



```
"ietf-network:networks": {
  "network": [
    { "network-id":"example-customized-blue-topology",
      "network-types": {
        "ietf-network-slice:network-slice": {} },
      "node": [
        {"node-id":"DCSG-3",
          "ietf-network-slice:network-slice": {
            "isolation-level":
            "ietf-network-slice:physical-memory-isolation"},
          "ietf-network-topology:termination-point": [
            {"tp-id":"220"} ] },
        {"node-id":"DCSG-5",
          "ietf-network-slice:network-slice": {
            "isolation-level":
            "ietf-network-slice:physical-memory-isolation"
          },
          "ietf-network-topology:termination-point": [
            {"tp-id":"6"} ] } ],
      "ietf-network-topology:link": [
        {
          "link-id":"DCSG-3,220,,",
          "source": { "source-node":"VR1", "source-tp":"220" },
          "destination": { "dest-node":"DCSG-5", "dest-tp":"6" },
          "ietf-network-slice:network-slice": {
            "isolation-level":
            "ietf-network-slice:physical-network-isolation" } } ],
      "ietf-network-slice:network-slice": {
        "isolation-level":
        "ietf-network-slice:physical-isolation"
      }
    }
  ]
}
```

Figure 38 Example of transport slice request

Finally, the virtual routers (vRouter) are created and properly set up. The Wireshark capture shows the internal gRPC protocol from Volta Networks, which is equivalent to OpenConfig calls, for the vRouter deployment and configuration. It includes a call to create a new vRouter on top of Edgecore hardware. Then, interfaces are incorporated to the vRouter and configured. Later, the virtual routing and forwarding (VRF) table is set up, and finally interfaces are attached to the VRF. This setup delay takes around 8.36s.

7. Conclusions and Next Steps

This deliverable serves as a reference document for the design, interface specification, and preliminary evaluation of core TeraFlow OS components, touching upon important areas of modern network operating systems, including (i) scalable high-performance SDN control plane, (ii) heterogeneous SDN hardware integration, (iii) service and OS lifecycle automation, and (iv) network slice management.

The documented components are engineered as parts of a fully disaggregated cloud-native network operating system, with the objective to address the above needs. The source code, mechanics, documentation, and installation guidelines of the core TeraFlow OS components are provided in MS3.2. The objective of this document is to provide additional context on the formal description, component architecture, and basic concepts of these core TeraFlow OS components, document their interactions (i.e., APIs) within the TeraFlow ecosystem but also externally, as well as provide a preliminary evaluation of their basic features.

Towards the final version of the lifecycle automation and high performance SDN components in WP3 (i.e., D3.2), a set of updates and extensions are planned for all the core TeraFlow OS components. Table 20 summarizes a list of expected extensions per component, while mapping the involvement of partners to the development activities that will fulfil these extensions.

Table 20: Future development plans for the core TeraFlow OS components and the contributing partners.

WP3 Task	Component Name	Component extensions	Involved Partners
T3.1	Context Management	• Experiment with other database backends. • Review and extend gRPC API, if required by other components.	CTTC
	Monitoring	• Upgrade management and metrics databases to support high data exchange and improve scalability • Extension of the event listening procedure to support all types of events • Complete information model and add new RPC methods accordingly.	ATOS
	Traffic Engineering	• Integration with Service, Device, and Context components. • TE workflow demonstration using emulated routers.	STR
	Auto Scaling	• Integration of the latest auto-scaling features from Kubernetes.	Features covered by Kubernetes

		• Export of KPI from multiple components using Prometheus API. HPA decision based on Prometheus obtained metrics.	Orchestrator (see MS3.2)
	Load balancing	• Integration of the latest load-balancing features from Kubernetes	
T3.2	Device	• Automated device discovery through interaction with the Automation component. • Complete implementation of device monitoring functions. • Improve OpenConfig driver templates. • Validate and improve TAPI driver. • P4 driver extensions with additional RPCs (e.g., support for P4 device configuration)	CTTC, TID, SIAE, INF, UBI
	Service	• Complete integration of Service with Device, Context, and Compute components. • Implement L3-VPN service.	CTTC
T3.3	Automation (ZTP)	• Automated device discovery in collaboration with the Device component. • Automated device-level policies (e.g., for monitoring) in collaboration with the Policy component.	UBI
	Policy Management	• ECA policy model implementation for device-level policies with a focus on monitoring cases.	UBI, ODC
T3.4	Slice Management	• Initial version of Slice Management component	ODC, CTTC

8. ANNEX

8.1. ONF TR-532 model parameters

8.1.1. ONF TR-532 - air-interface parameters

```
module: air-interface-2-0
  augment /core-model:control-construct/core-model:logical-termination-point/core-model:layer-
protocol:
    +--rw air-interface-pac
       +--ro air-interface-capability
       |  +--ro type-of-equipment?                               string
       |  +--ro tx-frequency-min?                                int32
       |  +--ro tx-frequency-max?                                int32
       |  +--ro rx-frequency-min?                                int32
       |  +--ro rx-frequency-max?                                int32
       |  +--ro transmission-mode-list* [transmission-mode-name]
       |  |  +--ro transmission-mode-name                 string
       |  |  +--ro transmission-mode-rank?                int32
       |  |  +--ro channel-bandwidth?                     int32
       |  |  +--ro modulation-scheme?                     int16
       |  |  +--ro code-rate?                             int8
       |  |  +--ro symbol-rate-reduction-factor?          int8
       |  |  +--ro tx-power-min?                          int8
       |  |  +--ro tx-power-max?                          int8
       |  |  +--ro rx-threshold?                          int16
       |  |  +--ro am-upshift-level?                      int8
       |  |  +--ro am-downshift-level?                    int8
       |  |  +--ro xpic-is-avail?                         boolean
       |  |  +--ro supported-as-fixed-configuration?      boolean
       |  +--ro duplex-distance-is-freely-configurable?          boolean
       |  +--ro duplex-distance-list*                            int32
       |  +--ro auto-freq-select-is-avail?                       boolean
       |  +--ro adaptive-modulation-is-avail?                    boolean
       |  +--ro atpc-is-avail?                                   boolean
       |  +--ro atpc-range?                                      int8
       |  +--ro supported-radio-signal-id-datatype?              radio-signal-id-datatype-type
       |  +--ro supported-radio-signal-id-length?                int16
       |  +--ro expected-equals-transmitted-radio-signal-id?     boolean
       |  +--ro encryption-is-avail?                             boolean
       |  +--ro supported-loop-back-kind-list*                   loop-back-type
       |  +--ro maintenance-timer-range?                         string
       |  +--ro supported-alarm-list*                            string
       |  +--ro performance-monitoring-is-avail?                 boolean
       |  +--ro direction-of-acm-performance-values?             direction-type
       +--rw air-interface-configuration
       |  +--rw air-interface-name?                        string
       |  +--rw remote-air-interface-name?                 string
       |  +--rw transmitted-radio-signal-id
       |  |  +--rw numeric-radio-signal-id?          uint16
       |  |  +--rw alphanumeric-radio-signal-id?     string
       |  +--rw expected-radio-signal-id
       |  |  +--rw numeric-radio-signal-id?          uint16
       |  |  +--rw alphanumeric-radio-signal-id?     string
       |  +--rw tx-frequency?                              int32
       |  +--rw rx-frequency?                              int32
       |  +--rw transmission-mode-min?                     -> /core-model:control-
construct/logical-termination-point/layer-protocol/air-interface:air-interface-pac/air-
interface-capability/transmission-mode-list/transmission-mode-name
       |  +--rw transmission-mode-max?                     -> /core-model:control-
construct/logical-termination-point/layer-protocol/air-interface:air-interface-pac/air-
interface-capability/transmission-mode-list/transmission-mode-name
       |  +--rw power-is-on?                               boolean
       |  +--rw transmitter-is-on?                         boolean
       |  +--rw receiver-is-on?                            boolean
       |  +--rw tx-power?                                  int8
       |  +--rw adaptive-modulation-is-on?                 boolean
       |  +--rw xpic-is-on?                                boolean
       |  +--rw mimo-is-on?                                boolean
       |  +--rw alic-is-on?                                boolean
       |  +--rw atpc-is-on?                            boolean
       |  +--rw atpc-thresh-upper?                     int16
       |  +--rw atpc-thresh-lower?                     int16
       |  +--rw atpc-tx-power-min?                     int8
       |  +--rw auto-freq-select-is-on?                boolean
       |  +--rw auto-freq-select-range?                int8
       |  +--rw modulation-is-on?                      boolean
       |  +--rw encryption-is-on?                      boolean
       |  +--rw cryptographic-key?                     string
       |  +--rw loop-back-kind-on?                     loop-back-type
       |  +--rw maintenance-timer?                     int32
       |  +--rw problem-kind-severity-list* [problem-kind-name]
       |  |  +--rw problem-kind-name        string
       |  |  +--rw problem-kind-severity?   severity-type
       |  +--rw g-826-threshold-cross-alarm-list* [g-826-value-kind granularity-period]
       |  |  +--rw g-826-value-kind            g-826-type
       |  |  +--rw alarm-raising-threshold?    int32
       |  |  +--rw alarm-clearing-threshold?   int32
       |  |  +--rw granularity-period          granularity-period-type
       |  +--rw xlts-threshold-cross-alarm-list* [level-threshold-second-kind granularity-
period xlts-threshold-cross-alarm-definition-number]
       |  |  +--rw level-threshold-second-kind                    xlevel-threshold-second-
kind-type
       |  |  +--rw xlts-level?                                    int8
       |  |  +--rw amount-of-seconds?                             int16
       |  |  +--rw granularity-period                             granularity-period-type
       |  |  +--rw xlts-threshold-cross-alarm-definition-number    int8
       |  +--rw acm-threshold-cross-alarm-list* [acm-threshold-cross-alarm-definition-number
granularity-period]
       |  |  +--rw acm-threshold-cross-alarm-definition-number    int8
       |  |  +--rw transmission-mode?                            -> /core-model:control-
construct/logical-termination-point/layer-protocol/air-interface:air-interface-pac/air-
interface-capability/transmission-mode-list/transmission-mode-name
       |  |  +--rw amount-of-seconds?                            int16
       |  |  +--rw granularity-period                            granularity-period-type
       |  +--rw clearing-threshold-cross-alarms-is-on?   boolean
       |  +--rw performance-monitoring-is-on?            boolean
       +--ro air-interface-status
       |  +--ro interface-status?              interface-status-type
       |  +--ro tx-frequency-cur?              int32
       |  +--ro rx-frequency-cur?              int32
       |  +--ro transmission-mode-cur?         -> /core-model:control-construct/logical-
termination-point/layer-protocol/air-interface:air-interface-pac/air-interface-
capability/transmission-mode-list/transmission-mode-name
       |  +--ro received-radio-signal-id
       |  |  +--ro numeric-radio-signal-id?        uint16
       |  |  +--ro alphanumeric-radio-signal-id?   string
       |  +--ro radio-power-is-up?             boolean
       |  +--ro link-is-up?                    boolean
       |  +--ro xpic-is-up?                    boolean
       |  +--ro mimo-is-up?                    boolean
       |  +--ro alic-is-up?                    boolean
       |  +--ro atpc-is-up?                    boolean
       |  +--ro auto-freq-select-is-up?        boolean
       |  +--ro local-end-point-id?            string
       |  +--ro remote-end-point-id?           string
       |  +--ro loop-back-kind-up?             loop-back-type
       |  +--ro performance-monitoring-is-up?  boolean
       |  +--ro rx-level-cur?                  int8
       |  +--ro tx-level-cur?                  int8
       |  +--ro snir-cur?                      int8
       |  +--ro xpd-cur?                       int8
       |  +--ro rf-temp-cur?                   int8
       +--ro air-interface-current-problems
       |  +--ro current-problem-list* [sequence-number]
       |  |  +--ro problem-name?       string
       |  |  +--ro sequence-number     int16
       |  |  +--ro timestamp?          yang:date-and-time
       |  |  +--ro problem-severity?   severity-type
       |  +--ro number-of-current-problems?   int8
       |  +--ro time-of-latest-change?        yang:date-and-time
       +--ro air-interface-current-performance
       |  +--ro current-performance-data-list* [granularity-period]
       |  |  +--ro performance-data
       |  |  |  +--ro es?                    int32
       |  |  |  +--ro ses?                 int32
       |  |  |  +--ro cses?                int32
       |  |  |  +--ro unavailability?      int32
       |  |  |  +--ro tx-level-min?        int8
       |  |  |  +--ro tx-level-max?        int8
       |  |  |  +--ro tx-level-avg?        int8
       |  |  |  +--ro rx-level-min?        int8
       |  |  |  +--ro rx-level-max?        int8
       |  |  |  +--ro rx-level-avg?        int8
       |  |  |  +--ro time-xstates-list* [time-xstate-sequence-number]
       |  |  |  |  +--ro time-xstate-sequence-number    int8
       |  |  |  |  +--ro transmission-mode?            -> /core-model:control-
construct/logical-termination-point/layer-protocol/air-interface:air-interface-pac/air-
interface-capability/transmission-mode-list/transmission-mode-name
       |  |  |  |  +--ro time?                          int32
       |  |  |  +--ro snir-min?            int8
       |  |  |  +--ro snir-max?            int8
       |  |  |  +--ro snir-avg?            int8
       |  |  |  +--ro xpd-min?             int8
       |  |  |  +--ro xpd-max?             int8
       |  |  |  +--ro xpd-avg?             int8
       |  |  |  +--ro rf-temp-min?         int8
       |  |  |  +--ro rf-temp-max?         int8
       |  |  |  +--ro rf-temp-avg?         int8
       |  |  |  +--ro defect-blocks-sum?   int16
       |  |  |  +--ro time-period?         int32
       |  |  +--ro timestamp?               yang:date-and-time
       |  |  +--ro suspect-interval-flag?   boolean
       |  |  +--ro elapsed-time?            int64
       |  |  +--ro scanner-id?              string
       |  |  +--ro granularity-period       granularity-period-type
       |  +--ro number-of-current-performance-sets?   int8
       +--ro air-interface-historical-performances
          +--ro historical-performance-data-list* [granularity-period period-end-time]
          |  +--ro performance-data
          |  |  +--ro es?                  int32
          |  |  +--ro ses?                 int32
          |  |  +--ro cses?                int32
          |  |  +--ro unavailability?      int32
          |  |  +--ro tx-level-min?        int8
          |  |  +--ro tx-level-max?        int8
          |  |  +--ro tx-level-avg?        int8
          |  |  +--ro rx-level-min?        int8
          |  |  +--ro rx-level-max?        int8
          |  |  +--ro rx-level-avg?        int8
          |  |  +--ro time-xstates-list* [time-xstate-sequence-number]
          |  |  |  +--ro time-xstate-sequence-number    int8
          |  |  |  +--ro transmission-mode?            -> /core-model:control-
construct/logical-termination-point/layer-protocol/air-interface:air-interface-pac/air-
interface-capability/transmission-mode-list/transmission-mode-name
          |  |  |  +--ro time?                          int32
          |  |  +--ro snir-min?            int8
          |  |  +--ro snir-max?            int8
          |  |  +--ro snir-avg?            int8
          |  |  +--ro xpd-min?             int8
          |  |  +--ro xpd-max?             int8
          |  |  +--ro xpd-avg?             int8
          |  |  +--ro rf-temp-min?         int8
          |  |  +--ro rf-temp-max?         int8
          |  |  +--ro rf-temp-avg?         int8
          |  |  +--ro defect-blocks-sum?   int16
          |  |  +--ro time-period?         int32
          |  +--ro suspect-interval-flag?   boolean
          |  +--ro history-data-id?         string
          |  +--ro granularity-period       granularity-period-type
          |  +--ro period-end-time          yang:date-and-time
          +--ro number-of-historical-performance-sets?   int16
          +--ro time-of-latest-change?                   yang:date-and-time
```

8.1.2. ONF TR-532 - co-channel-profile model parameters

```
module: co-channel-profile-1-0
```

```
  augment /core-model:control-construct/core-model:profile-collection/core-model:profile:
    +--rw co-channel-profile-pac
       +--ro co-channel-profile-capability
       |  +--ro xpic-is-avail?                  boolean
       |  +--ro mimo-is-avail?                  boolean
       |  +--ro number-of-mimo-channels-max?    int8
       |  +--ro alic-is-avail?                  boolean
       +--rw co-channel-profile-configuration
          +--rw profile-name?                      string
          +--rw kind-of-co-channel-group?          kind-of-co-channel-group-type
          +--rw logical-termination-point-list*    -> /core-model:control-construct/logical-
termination-point/uuid
```

8.1.3. ONF TR-532 - core-model model parameters

```
module: core-model-1-4
  +--rw control-construct!
     +--rw top-level-equipment*         -> /control-construct/equipment/uuid
     +--rw equipment* [uuid]
     |  +--rw connector* [local-id]
     |  |  +--rw connector?                string
     |  |  +--rw orientation?              connector-and-pin-orientation
     |  |  +--rw pin-layout?               string
     |  |  +--rw connector-type?           string
     |  |  +--rw role?                     string
     |  |  +--rw local-id                  string
     |  |  +--rw name* [value-name]
     |  |  |  +--rw value-name    string
     |  |  |  +--rw value?        string
     |  |  +--rw label* [value-name]
     |  |  |  +--rw value-name    string
     |  |  |  +--rw value?        string
     |  |  +--rw extension* [value-name]
     |  |  |  +--rw value-name    string
     |  |  |  +--rw value?        string
     |  |  +--ro operational-state?        operational-state
     |  |  +--rw administrative-control?   administrative-control
     |  |  +--ro administrative-state?     administrative-state
     |  |  +--rw lifecycle-state?          lifecycle-state
     |  |  +--rw address*                  dt-address
     |  +--rw contained-holder* [local-id]
     |  |  +--rw occupying-fru?            -> /control-construct/equipment/uuid
     |  |  +--rw holder-location?          dt-address
     |  |  +--rw expected-holder
     |  |  |  +--rw spatial-properties-of-type
     |  |  |  |  +--rw height?   string
     |  |  |  |  +--rw width?    string
     |  |  |  |  +--rw length?   string
     |  |  |  +--rw environmental-rating
     |  |  |  |  +--rw thermal-rating
     |  |  |  |  |  +--rw thermal-rating-name?   string
     |  |  |  |  |  +--rw maximum-temperature?   decimal64
     |  |  |  |  |  +--rw minimum-temperature?   decimal64
     |  |  |  |  +--rw power-rating
     |  |  |  |  |  +--rw power-rating-name?    string
     |  |  |  |  |  +--rw power-rating-value?   string
     |  |  |  |  +--rw humidity-rating?   string
     |  |  |  +--rw position
     |  |  |  |  +--rw relative-position?   string
     |  |  |  +--rw holder-structure
     |  |  |  |  +--rw holder-category?      holder-category
     |  |  |  |  +--rw is-captive?           boolean
     |  |  |  |  +--rw is-guided?            boolean
     |  |  |  |  +--rw is-quantised-space?   boolean
     |  |  |  +--rw local-id?                     string
     |  |  |  +--rw name* [value-name]
     |  |  |  |  +--rw value-name    string
     |  |  |  |  +--rw value?        string
     |  |  |  +--rw label* [value-name]
     |  |  |  |  +--rw value-name    string
     |  |  |  |  +--rw value?        string
     |  |  |  +--rw extension* [value-name]
     |  |  |  |  +--rw value-name    string
     |  |  |  |  +--rw value?        string
     |  |  |  +--ro operational-state?            operational-state
     |  |  |  +--rw administrative-control?       administrative-control
     |  |  |  +--ro administrative-state?         administrative-state
     |  |  |  +--rw lifecycle-state?              lifecycle-state
     |  |  |  +--rw address*                      dt-address
     |  |  +--rw actual-holder
     |  |  |  +--rw spatial-properties-of-type
     |  |  |  |  +--rw height?   string
     |  |  |  |  +--rw width?    string
     |  |  |  |  +--rw length?   string
     |  |  |  +--rw environmental-rating
     |  |  |  |  +--rw thermal-rating
     |  |  |  |  |  +--rw thermal-rating-name?   string
     |  |  |  |  |  +--rw maximum-temperature?   decimal64
     |  |  |  |  |  +--rw minimum-temperature?   decimal64
     |  |  |  |  +--rw power-rating
     |  |  |  |  |  +--rw power-rating-name?    string
     |  |  |  |  |  +--rw power-rating-value?   string
     |  |  |  |  +--rw humidity-rating?   string
     |  |  |  +--rw position
     |  |  |  |  +--rw relative-position?   string
     |  |  |  +--rw holder-structure
     |  |  |  |  +--rw holder-category?      holder-category
     |  |  |  |  +--rw is-captive?           boolean
     |  |  |  |  +--rw is-guided?            boolean
     |  |  |  |  +--rw is-quantised-space?   boolean
     |  |  |  +--rw local-id?                     string
     |  |  |  +--rw name* [value-name]
     |  |  |  |  +--rw value-name    string
     |  |  |  |  +--rw value?        string
     |  |  |  +--rw label* [value-name]
     |  |  |  |  +--rw value-name    string
     |  |  |  |  +--rw value?        string
     |  |  |  +--rw extension* [value-name]
     |  |  |  |  +--rw value-name    string
     |  |  |  |  +--rw value?        string
     |  |  |  +--ro operational-state?            operational-state
     |  |  |  +--rw administrative-control?       administrative-control
     |  |  |  +--ro administrative-state?         administrative-state
     |  |  |  +--rw lifecycle-state?              lifecycle-state
     |  |  |  +--rw address*                      dt-address
     |  |  +--rw local-id                  string
     |  |  +--rw name* [value-name]
     |  |  |  +--rw value-name    string
     |  |  |  +--rw value?        string
     |  |  +--rw label* [value-name]
     |  |  |  +--rw value-name    string
     |  |  |  +--rw value?        string
     |  |  +--rw extension* [value-name]
     |  |  |  +--rw value-name    string
     |  |  |  +--rw value?        string
     |  |  +--ro operational-state?        operational-state
     |  |  +--rw administrative-control?   administrative-control
     |  |  +--ro administrative-state?     administrative-state
     |  |  +--rw lifecycle-state?          lifecycle-state
     |  |  +--rw address*                  dt-address
     |  +--ro is-field-replaceable?            boolean
     |  +--rw function-block*                  string
     |  +--rw expected-equipment* [local-id]
     |  |  +--rw location
     |  |  |  +--rw equipment-location?      dt-address
     |  |  |  +--rw geographical-location?   dt-address
     |  |  +--rw structure
     |  |  |  +--rw category?   equipment-category
     |  |  +--rw swappability
     |  |  |  +--rw is-hot-swappable?   boolean
     |  |  +--rw physical-properties
     |  |  |  +--rw temperature?   string
     |  |  +--rw function-enablers
     |  |  |  +--rw power-state?   string
     |  |  +--rw mechanical-functions
     |  |  |  +--rw rotation-speed?   string
     |  |  +--rw physical-characteristics
     |  |  |  +--rw weight-characeristics?   string
     |  |  |  +--rw fire-characteristics?    string
     |  |  |  +--rw materials?               string
     |  |  +--rw mechanical-features
     |  |  +--rw spatial-properties-of-type
     |  |  |  +--rw height?   string
     |  |  |  +--rw width?    string
     |  |  |  +--rw length?   string
     |  |  +--rw environmental-rating
     |  |  |  +--rw thermal-rating
     |  |  |  |  +--rw thermal-rating-name?   string
     |  |  |  |  +--rw maximum-temperature?   decimal64
     |  |  |  |  +--rw minimum-temperature?   decimal64
     |  |  |  +--rw power-rating
     |  |  |  |  +--rw power-rating-name?    string
     |  |  |  |  +--rw power-rating-value?   string
     |  |  |  +--rw humidity-rating?   string
     |  |  +--rw manufactured-thing
     |  |  |  +--rw manufacturer-properties
     |  |  |  |  +--rw manufacturer-identifier?   string
     |  |  |  |  +--rw manufacturer-name?         string
     |  |  |  +--rw equipment-type
     |  |  |  |  +--rw description?            string
     |  |  |  |  +--rw model-identifier?       string
     |  |  |  |  +--rw part-type-identifier?   string
     |  |  |  |  +--rw type-name?              string
     |  |  |  |  +--rw version?                string
     |  |  |  +--rw equipment-instance
     |  |  |  |  +--rw manufacture-date?            yang:date-and-time
     |  |  |  |  +--rw serial-number?               string
     |  |  |  |  +--rw asset-instance-identifier?   string
     |  |  |  +--rw operator-augmented-equipment-type
     |  |  |  |  +--rw asset-type-identifier?   string
     |  |  |  +--rw operator-augmented-equipment-instance
     |  |  |     +--rw asset-instance-identifier?   string
     |  |  +--rw local-id                      string
     |  |  +--rw name* [value-name]
     |  |  |  +--rw value-name    string
     |  |  |  +--rw value?        string
     |  |  +--rw label* [value-name]
     |  |  |  +--rw value-name    string
     |  |  |  +--rw value?        string
     |  |  +--rw extension* [value-name]
     |  |  |  +--rw value-name    string
     |  |  |  +--rw value?        string
     |  |  +--ro operational-state?           operational-state
     |  |  +--rw administrative-control?      administrative-control
     |  |  +--ro administrative-state?        administrative-state
     |  |  +--rw lifecycle-state?             lifecycle-state
     |  |  +--rw address*                     dt-address
     |  +--rw actual-equipment
     |  |  +--rw location
     |  |  |  +--rw equipment-location?      dt-address
     |  |  |  +--rw geographical-location?   dt-address
     |  |  +--rw structure
     |  |  |  +--rw category?   equipment-category
     |  |  +--rw swappability
     |  |  |  +--rw is-hot-swappable?   boolean
     |  |  +--rw physical-properties
     |  |  |  +--rw temperature?   string
     |  |  +--rw function-enablers
     |  |  |  +--rw power-state?   string
     |  |  +--rw mechanical-functions
     |  |  |  +--rw rotation-speed?   string
     |  |  +--rw physical-characteristics
     |  |  |  +--rw weight-characeristics?   string
     |  |  |  +--rw fire-characteristics?    string
     |  |  |  +--rw materials?               string
     |  |  +--rw mechanical-features
     |  |  +--rw spatial-properties-of-type
     |  |  |  +--rw height?   string
     |  |  |  +--rw width?    string
     |  |  |  +--rw length?   string
     |  |  +--rw environmental-rating
     |  |  |  +--rw thermal-rating
     |  |  |  |  +--rw thermal-rating-name?   string
```

```
   |  |  |  |  +--rw maximum-temperature?   decimal64
   |  |  |  |  +--rw minimum-temperature?   decimal64
   |  |  |  +--rw power-rating
   |  |  |  |  +--rw power-rating-name?    string
   |  |  |  |  +--rw power-rating-value?   string
   |  |  |  +--rw humidity-rating?   string
   |  |  +--rw manufactured-thing
   |  |  |  +--rw manufacturer-properties
   |  |  |  |  +--rw manufacturer-identifier?   string
   |  |  |  |  +--rw manufacturer-name?         string
   |  |  |  +--rw equipment-type
   |  |  |  |  +--rw description?            string
   |  |  |  |  +--rw model-identifier?       string
   |  |  |  |  +--rw part-type-identifier?   string
   |  |  |  |  +--rw type-name?              string
   |  |  |  |  +--rw version?                string
   |  |  |  +--rw equipment-instance
   |  |  |  |  +--rw manufacture-date?            yang:date-and-time
   |  |  |  |  +--rw serial-number?               string
   |  |  |  |  +--rw asset-instance-identifier?   string
   |  |  |  +--rw operator-augmented-equipment-type
   |  |  |  |  +--rw asset-type-identifier?   string
   |  |  |  +--rw operator-augmented-equipment-instance
   |  |  |     +--rw asset-instance-identifier?   string
   |  |  +--rw local-id?                     string
   |  |  +--rw name* [value-name]
   |  |  |  +--rw value-name    string
   |  |  |  +--rw value?        string
   |  |  +--rw label* [value-name]
   |  |  |  +--rw value-name    string
   |  |  |  +--rw value?        string
   |  |  +--rw extension* [value-name]
   |  |  |  +--rw value-name    string
   |  |  |  +--rw value?        string
   |  |  +--ro operational-state?          operational-state
   |  |  +--rw administrative-control?     administrative-control
   |  |  +--ro administrative-state?       administrative-state
   |  |  +--rw lifecycle-state?            lifecycle-state
   |  |  +--rw address*                    dt-address
   |  +--rw equipment-functional-boundary?   string
   |  +--rw external-managed-id
   |  |  +--rw manager-identifier?      string
   |  |  +--rw external-managed-uuid?   string
   |  +--rw local-id?                        string
   |  +--rw uuid                             universal-id
   |  +--rw name* [value-name]
   |  |  +--rw value-name    string
   |  |  +--rw value?        string
   |  +--rw label* [value-name]
   |  |  +--rw value-name    string
   |  |  +--rw value?        string
   |  +--rw extension* [value-name]
   |  |  +--rw value-name    string
   |  |  +--rw value?        string
   |  +--ro operational-state?             operational-state
   |  +--rw administrative-control?        administrative-control
   |  +--ro administrative-state?          administrative-state
   |  +--rw lifecycle-state?               lifecycle-state
   |  +--rw address*                       dt-address
   +--rw logical-termination-point* [uuid]
   |  +--rw server-ltp*               -> /control-construct/logical-termination-point/uuid
   |  +--rw client-ltp*               -> /control-construct/logical-termination-point/uuid
   |  +--rw layer-protocol* [local-id]
   |  |  +--rw layer-protocol-name?             layer-protocol-name-type
   |  |  +--rw configured-client-capacity?      string
   |  |  +--rw lp-direction?                    termination-direction
   |  |  +--rw termination-state?               termination-state
   |  |  +--rw configuration-and-switch-control*   string
   |  |  +--rw is-protection-lock-out?          boolean
   |  |  +--rw fc-blocks-signal-to-lp?          string
   |  |  +--rw local-id                         string
   |  |  +--rw name* [value-name]
   |  |  |  +--rw value-name    string
   |  |  |  +--rw value?        string
   |  |  +--rw label* [value-name]
```

```
     |  |  |  +--rw value-name    string
     |  |  |  +--rw value?        string
     |  |  +--rw extension* [value-name]
     |  |  |  +--rw value-name    string
     |  |  |  +--rw value?        string
     |  |  +--ro operational-state?                operational-state
     |  |  +--rw administrative-control?           administrative-control
     |  |  +--ro administrative-state?             administrative-state
     |  |  +--rw lifecycle-state?                  lifecycle-state
     |  |  +--rw address*                          dt-address
     |  +--rw connected-ltp?           -> /control-construct/logical-termination-point/uuid
     |  +--rw peer-ltp?                -> /control-construct/logical-termination-point/uuid
     |  +--rw physical-port-reference*   -> /control-construct/equipment/uuid
     |  +--rw ltp-in-other-view*       -> /control-construct/logical-termination-point/uuid
     |  +--rw ltp-direction?           termination-direction
     |  +--rw transfer-capacity-pac?    string
     |  +--rw fd-rule-group*           -> /control-construct/forwarding-domain/uuid
     |  +--rw embedded-clock* [local-id]
     |  |  +--ro run-mode?                  run-mode
     |  |  +--rw encompassed-clock*         -> /control-construct/logical-termination-
point/embedded-clock/local-id
     |  |  +--rw encapsulated-fc* [uuid]
     |  |  |  +--rw layer-protocol-name?               layer-protocol-name-type
     |  |  |  +--rw lower-level-fc*                    -> /control-construct/forwarding-
domain/fc/uuid
     |  |  |  +--rw fc-route?                          string
     |  |  |  +--rw fc-port* [local-id]
     |  |  |  |  +--rw logical-termination-point*       -> /control-construct/logical-
termination-point/uuid
     |  |  |  |  +--rw role?                           port-role
     |  |  |  |  +--rw fc-port-direction?              port-direction
     |  |  |  |  +--rw is-protection-lock-out?         boolean {fcportisprotectionlockout}?
     |  |  |  |  +--rw selection-priority?             int64
     |  |  |  |  +--ro is-internal-port?               boolean
     |  |  |  |  +--rw fc-route-feeds-fc-port-egress*   string
     |  |  |  |  +--rw fc-port*                        -> /control-construct/forwarding-
domain/fc/fc-port/local-id
     |  |  |  |  +--rw port-of-internal-fc*            -> /control-construct/forwarding-
domain/fc/fc-port/local-id
     |  |  |  |  +--rw local-id                        string
     |  |  |  |  +--rw name* [value-name]
     |  |  |  |  |  +--rw value-name    string
     |  |  |  |  |  +--rw value?        string
     |  |  |  |  +--rw label* [value-name]
     |  |  |  |  |  +--rw value-name    string
     |  |  |  |  |  +--rw value?        string
     |  |  |  |  +--rw extension* [value-name]
     |  |  |  |  |  +--rw value-name    string
     |  |  |  |  |  +--rw value?        string
     |  |  |  |  +--ro operational-state?              operational-state
     |  |  |  |  +--rw administrative-control?         administrative-control
     |  |  |  |  +--ro administrative-state?           administrative-state
     |  |  |  |  +--rw lifecycle-state?                lifecycle-state
     |  |  |  |  +--rw address*                        dt-address
     |  |  |  +--rw fc-switch* [local-id]
     |  |  |  |  +--rw hold-off-time?                              int64
     |  |  |  |  +--rw prot-type?                                  protection-type
     |  |  |  |  +--rw reversion-mode?                             reversion-mode
     |  |  |  |  +--rw selected-fc-port* [local-id]
     |  |  |  |  |  +--rw logical-termination-point*       -> /control-construct/logical-
termination-point/uuid
     |  |  |  |  |  +--rw role?                           port-role
     |  |  |  |  |  +--rw fc-port-direction?              port-direction
     |  |  |  |  |  +--rw is-protection-lock-out?         boolean
{fcportisprotectionlockout}?
     |  |  |  |  |  +--rw selection-priority?             int64
     |  |  |  |  |  +--ro is-internal-port?               boolean
     |  |  |  |  |  +--rw fc-route-feeds-fc-port-egress*   string
     |  |  |  |  |  +--rw fc-port*                        -> /control-construct/forwarding-
domain/fc/fc-port/local-id
     |  |  |  |  |  +--rw port-of-internal-fc*            -> /control-construct/forwarding-
domain/fc/fc-port/local-id
     |  |  |  |  |  +--rw local-id                        string
     |  |  |  |  |  +--rw name* [value-name]
     |  |  |  |  |  |  +--rw value-name    string
     |  |  |  |  |  |  +--rw value?        string
     |  |  |  |  |  +--rw label* [value-name]
     |  |  |  |  |  |  +--rw value-name    string
     |  |  |  |  |  |  +--rw value?        string
     |  |  |  |  |  +--rw extension* [value-name]
     |  |  |  |  |  |  +--rw value-name    string
     |  |  |  |  |  |  +--rw value?        string
     |  |  |  |  |  +--ro operational-state?              operational-state
     |  |  |  |  |  +--rw administrative-control?         administrative-control
     |  |  |  |  |  +--ro administrative-state?           administrative-state
     |  |  |  |  |  +--rw lifecycle-state?                lifecycle-state
     |  |  |  |  |  +--rw address*                        dt-address
     |  |  |  |  +--rw profile-proxy*                              string
     |  |  |  |  +--rw configuration-and-switch-control?           string
     |  |  |  |  +--rw internal-configuration-and-switch-control?  string
     |  |  |  |  +--rw switch-control?                             switch-control
     |  |  |  |  +--rw switch-selects-ports?                       port-direction
     |  |  |  |  +--ro switch-selection-reason?                    switch-state-reason
     |  |  |  |  +--rw control-parameters
     |  |  |  |  |  +--rw reversion-mode?                 reversion-mode
     |  |  |  |  |  +--rw wait-to-revert-time?            int64
     |  |  |  |  |  +--rw prot-type?                      protection-type
     |  |  |  |  |  +--rw hold-off-time?                  int64
     |  |  |  |  |  +--rw network-scheme-specification?   string
     |  |  |  |  +--rw wait-to-restore-time?                       int64
     |  |  |  |  +--rw local-id                                    string
     |  |  |  |  +--rw name* [value-name]
     |  |  |  |  |  +--rw value-name    string
     |  |  |  |  |  +--rw value?        string
     |  |  |  |  +--rw label* [value-name]
     |  |  |  |  |  +--rw value-name    string
     |  |  |  |  |  +--rw value?        string
     |  |  |  |  +--rw extension* [value-name]
     |  |  |  |  |  +--rw value-name    string
     |  |  |  |  |  +--rw value?        string
     |  |  |  |  +--ro operational-state?                          operational-state
     |  |  |  |  +--rw administrative-control?                     administrative-control
     |  |  |  |  +--ro administrative-state?                       administrative-state
     |  |  |  |  +--rw lifecycle-state?                            lifecycle-state
     |  |  |  |  +--rw address*                                    dt-address
     |  |  |  +--rw configuration-and-switch-control*   string
     |  |  |  +--rw forwarding-direction?              forwarding-direction
     |  |  |  +--rw is-protection-lock-out?            boolean
{forwardingconstructisprotectionlockout}?
     |  |  |  +--rw service-priority?                  int64
     |  |  |  +--rw supported-link*                    string
     |  |  |  +--rw supporting-pc?                     string
     |  |  |  +--rw external-managed-id
     |  |  |  |  +--rw manager-identifier?      string
     |  |  |  |  +--rw external-managed-uuid?   string
     |  |  |  +--rw local-id?                          string
     |  |  |  +--rw uuid                               universal-id
     |  |  |  +--rw name* [value-name]
     |  |  |  |  +--rw value-name    string
     |  |  |  |  +--rw value?        string
     |  |  |  +--rw label* [value-name]
     |  |  |  |  +--rw value-name    string
     |  |  |  |  +--rw value?        string
     |  |  |  +--rw extension* [value-name]
     |  |  |  |  +--rw value-name    string
     |  |  |  |  +--rw value?        string
     |  |  |  +--ro operational-state?                 operational-state
     |  |  |  +--rw administrative-control?            administrative-control
     |  |  |  +--ro administrative-state?              administrative-state
     |  |  |  +--rw lifecycle-state?                   lifecycle-state
     |  |  |  +--rw address*                           dt-address
     |  |  +--rw sync-ltp*                 -> /control-construct/logical-termination-
point/uuid
     |  |  +--rw encapsulated-casc*        string
     |  |  +--rw phase-aligned-clock*      -> /control-construct/logical-termination-
point/embedded-clock/local-id
     |  |  +--rw local-id                  string
     |  |  +--rw name* [value-name]
     |  |  |  +--rw value-name    string
     |  |  |  +--rw value?        string
     |  |  +--rw label* [value-name]
     |  |  |  +--rw value-name    string
     |  |  |  +--rw value?        string
     |  |  +--rw extension* [value-name]
     |  |  |  +--rw value-name    string
     |  |  |  +--rw value?        string
     |  |  +--ro operational-state?        operational-state
     |  |  +--rw administrative-control?   administrative-control
     |  |  +--ro administrative-state?     administrative-state
     |  |  +--rw lifecycle-state?          lifecycle-state
     |  |  +--rw address*                  dt-address
     |  +--rw supporting-pc?             string
     |  +--rw external-managed-id
     |  |  +--rw manager-identifier?      string
     |  |  +--rw external-managed-uuid?   string
     |  +--rw local-id?                  string
     |  +--rw uuid                       universal-id
     |  +--rw name* [value-name]
     |  |  +--rw value-name    string
     |  |  +--rw value?        string
     |  +--rw label* [value-name]
     |  |  +--rw value-name    string
     |  |  +--rw value?        string
     |  +--rw extension* [value-name]
     |  |  +--rw value-name    string
     |  |  +--rw value?        string
     |  +--ro operational-state?         operational-state
     |  +--rw administrative-control?    administrative-control
     |  +--ro administrative-state?      administrative-state
     |  +--rw lifecycle-state?           lifecycle-state
     |  +--rw address*                   dt-address
     +--rw forwarding-domain* [uuid]
     |  +--rw layer-protocol-name*         layer-protocol-name-type
     |  +--rw lower-level-fd*              -> /control-construct/forwarding-domain/uuid
     |  +--rw fc* [uuid]
     |  |  +--rw layer-protocol-name?                layer-protocol-name-type
     |  |  +--rw lower-level-fc*                     -> /control-construct/forwarding-
domain/fc/uuid
     |  |  +--rw fc-route?                           string
     |  |  +--rw fc-port* [local-id]
     |  |  |  +--rw logical-termination-point*       -> /control-construct/logical-
termination-point/uuid
     |  |  |  +--rw role?                            port-role
     |  |  |  +--rw fc-port-direction?               port-direction
     |  |  |  +--rw is-protection-lock-out?          boolean {fcportisprotectionlockout}?
     |  |  |  +--rw selection-priority?              int64
     |  |  |  +--ro is-internal-port?                boolean
     |  |  |  +--rw fc-route-feeds-fc-port-egress*   string
     |  |  |  +--rw fc-port*                         -> /control-construct/forwarding-
domain/fc/fc-port/local-id
     |  |  |  +--rw port-of-internal-fc*             -> /control-construct/forwarding-
domain/fc/fc-port/local-id
     |  |  |  +--rw local-id                         string
     |  |  |  +--rw name* [value-name]
     |  |  |  |  +--rw value-name    string
     |  |  |  |  +--rw value?        string
     |  |  |  +--rw label* [value-name]
     |  |  |  |  +--rw value-name    string
     |  |  |  |  +--rw value?        string
     |  |  |  +--rw extension* [value-name]
     |  |  |  |  +--rw value-name    string
     |  |  |  |  +--rw value?        string
     |  |  |  +--ro operational-state?               operational-state
     |  |  |  +--rw administrative-control?          administrative-control
     |  |  |  +--ro administrative-state?            administrative-state
     |  |  |  +--rw lifecycle-state?                 lifecycle-state
     |  |  |  +--rw address*                         dt-address
     |  |  +--rw fc-switch* [local-id]
     |  |  |  +--rw hold-off-time?                               int64
     |  |  |  +--rw prot-type?                                   protection-type
     |  |  |  +--rw reversion-mode?                              reversion-mode
     |  |  |  +--rw selected-fc-port* [local-id]
     |  |  |  |  +--rw logical-termination-point*       -> /control-construct/logical-
termination-point/uuid
     |  |  |  |  +--rw role?                            port-role
     |  |  |  |  +--rw fc-port-direction?               port-direction
     |  |  |  |  +--rw is-protection-lock-out?          boolean {fcportisprotectionlockout}?
     |  |  |  |  +--rw selection-priority?              int64
     |  |  |  |  +--ro is-internal-port?                boolean
     |  |  |  |  +--rw fc-route-feeds-fc-port-egress*   string
     |  |  |  |  +--rw fc-port*                         -> /control-construct/forwarding-
domain/fc/fc-port/local-id
     |  |  |  |  +--rw port-of-internal-fc*             -> /control-construct/forwarding-
domain/fc/fc-port/local-id
     |  |  |  |  +--rw local-id                         string
     |  |  |  |  +--rw name* [value-name]
     |  |  |  |  |  +--rw value-name    string
     |  |  |  |  |  +--rw value?        string
     |  |  |  |  +--rw label* [value-name]
     |  |  |  |  |  +--rw value-name    string
     |  |  |  |  |  +--rw value?        string
     |  |  |  |  +--rw extension* [value-name]
     |  |  |  |  |  +--rw value-name    string
     |  |  |  |  |  +--rw value?        string
     |  |  |  |  +--ro operational-state?               operational-state
     |  |  |  |  +--rw administrative-control?          administrative-control
     |  |  |  |  +--ro administrative-state?            administrative-state
     |  |  |  |  +--rw lifecycle-state?                 lifecycle-state
     |  |  |  |  +--rw address*                         dt-address
     |  |  |  +--rw profile-proxy*                               string
     |  |  |  +--rw configuration-and-switch-control?            string
     |  |  |  +--rw internal-configuration-and-switch-control?   string
     |  |  |  +--rw switch-control?                              switch-control
     |  |  |  +--rw switch-selects-ports?                        port-direction
     |  |  |  +--ro switch-selection-reason?                     switch-state-reason
     |  |  |  +--rw control-parameters
     |  |  |  |  +--rw reversion-mode?                  reversion-mode
     |  |  |  |  +--rw wait-to-revert-time?             int64
     |  |  |  |  +--rw prot-type?                       protection-type
     |  |  |  |  +--rw hold-off-time?                   int64
     |  |  |  |  +--rw network-scheme-specification?    string
     |  |  |  +--rw wait-to-restore-time?                        int64
     |  |  |  +--rw local-id                                     string
     |  |  |  +--rw name* [value-name]
     |  |  |  |  +--rw value-name    string
     |  |  |  |  +--rw value?        string
     |  |  |  +--rw label* [value-name]
     |  |  |  |  +--rw value-name    string
     |  |  |  |  +--rw value?        string
     |  |  |  +--rw extension* [value-name]
     |  |  |  |  +--rw value-name    string
     |  |  |  |  +--rw value?        string
     |  |  |  +--ro operational-state?                           operational-state
     |  |  |  +--rw administrative-control?                      administrative-control
     |  |  |  +--ro administrative-state?                        administrative-state
     |  |  |  +--rw lifecycle-state?                             lifecycle-state
     |  |  |  +--rw address*                                     dt-address
     |  |  +--rw configuration-and-switch-control*   string
     |  |  +--rw forwarding-direction?               forwarding-direction
     |  |  +--rw is-protection-lock-out?             boolean
{forwardingconstructisprotectionlockout}?
     |  |  +--rw service-priority?                   int64
     |  |  +--rw supported-link*                     string
     |  |  +--rw supporting-pc?                      string
     |  |  +--rw external-managed-id
     |  |  |  +--rw manager-identifier?      string
     |  |  |  +--rw external-managed-uuid?   string
     |  |  +--rw local-id?                           string
     |  |  +--rw uuid                                universal-id
     |  |  +--rw name* [value-name]
     |  |  |  +--rw value-name    string
     |  |  |  +--rw value?        string
     |  |  +--rw label* [value-name]
     |  |  |  +--rw value-name    string
     |  |  |  +--rw value?        string
     |  |  +--rw extension* [value-name]
     |  |  |  +--rw value-name    string
     |  |  |  +--rw value?        string
     |  |  +--ro operational-state?                  operational-state
     |  |  +--rw administrative-control?             administrative-control
     |  |  +--ro administrative-state?               administrative-state
     |  |  +--rw lifecycle-state?                    lifecycle-state
     |  |  +--rw address*                            dt-address
     |  +--rw logical-termination-point*   -> /control-construct/logical-termination-
point/uuid
     |  +--rw fd-port* [local-id]
     |  |  +--rw logical-termination-point*   -> /control-construct/logical-termination-
point/uuid
     |  |  +--rw role?                       port-role
     |  |  +--rw fd-port-direction?          port-direction
     |  |  +--rw fc-port*                    -> /control-construct/forwarding-domain/fc/fc-
port/local-id
     |  |  +--rw fd-port*                    -> /control-construct/forwarding-domain/fd-
port/local-id
     |  |  +--rw local-id                    string
     |  |  +--rw name* [value-name]
     |  |  |  +--rw value-name    string
     |  |  |  +--rw value?        string
     |  |  +--rw label* [value-name]
     |  |  |  +--rw value-name    string
     |  |  |  +--rw value?        string
     |  |  +--rw extension* [value-name]
     |  |  |  +--rw value-name    string
     |  |  |  +--rw value?        string
     |  |  +--ro operational-state?          operational-state
     |  |  +--rw administrative-control?     administrative-control
     |  |  +--ro administrative-state?       administrative-state
     |  |  +--rw lifecycle-state?            lifecycle-state
     |  |  +--rw address*                    dt-address
     |  +--rw external-managed-id
     |  |  +--rw manager-identifier?      string
     |  |  +--rw external-managed-uuid?   string
     |  +--rw local-id?                    string
     |  +--rw uuid                         universal-id
     |  +--rw name* [value-name]
     |  |  +--rw value-name    string
     |  |  +--rw value?        string
     |  +--rw label* [value-name]
     |  |  +--rw value-name    string
     |  |  +--rw value?        string
     |  +--rw extension* [value-name]
     |  |  +--rw value-name    string
     |  |  +--rw value?        string
     |  +--ro operational-state?           operational-state
     |  +--rw administrative-control?      administrative-control
     |  +--ro administrative-state?        administrative-state
     |  +--rw lifecycle-state?             lifecycle-state
     |  +--rw address*                     dt-address
     +--rw profile-collection
     |  +--rw profile* [uuid]
     |     +--rw profile-name?             profile-name-type
     |     +--rw external-managed-id
     |     |  +--rw manager-identifier?      string
     |     |  +--rw external-managed-uuid?   string
     |     +--rw local-id?                 string
     |     +--rw uuid                      universal-id
     |     +--rw name* [value-name]
     |     |  +--rw value-name    string
     |     |  +--rw value?        string
     |     +--rw label* [value-name]
     |     |  +--rw value-name    string
     |     |  +--rw value?        string
     |     +--rw extension* [value-name]
     |     |  +--rw value-name    string
     |     |  +--rw value?        string
     |     +--ro operational-state?        operational-state
     |     +--rw administrative-control?   administrative-control
     |     +--ro administrative-state?     administrative-state
     |     +--rw lifecycle-state?          lifecycle-state
     |     +--rw address*                  dt-address
     +--rw external-managed-id
     |  +--rw manager-identifier?      string
     |  +--rw external-managed-uuid?   string
     +--rw local-id?                    string
     +--rw uuid?                        universal-id
     +--rw name* [value-name]
     |  +--rw value-name    string
     |  +--rw value?        string
     +--rw label* [value-name]
     |  +--rw value-name    string
     |  +--rw value?        string
     +--rw extension* [value-name]
     |  +--rw value-name    string
     |  +--rw value?        string
     +--ro operational-state?          operational-state
     +--rw administrative-control?     administrative-control
     +--ro administrative-state?       administrative-state
     +--rw lifecycle-state?            lifecycle-state
     +--rw address*                    dt-address
```

8.1.4. ONF TR-532 - ethernet-container model parameters

```
module: ethernet-container-2-0
  augment /core-model:control-construct/core-model:logical-termination-point/core-model:layer-
protocol:
    +--rw ethernet-container-pac
       +--ro ethernet-container-capability
       |  +--ro available-queue-list* [queue-name]
       |  |  +--ro queue-name                                        queue-name-type
       |  |  +--ro max-queue-depth?                                  int32
       |  |  +--ro queue-depth-configuration-is-avail?               boolean
       |  |  +--ro available-dropping-behavior-kind-list*            dropping-behavior-kind-
type
       |  |  +--ro available-drop-precedence-kind-list*              drop-precedence-type
       |  |  +--ro wred-profile-per-drop-precedence-is-available?   boolean
       |  |  +--ro available-scheduling-kind-list*                   scheduler-kind-type
       |  +--ro explicit-congestion-notification-is-avail?                 boolean
       |  +--ro ingress-policing-is-avail?                                  boolean
       |  +--ro egress-shaping-is-avail?                                    boolean
       |  +--ro information-rate-min?                                       int32
       |  +--ro information-rate-max?                                       int32
       |  +--ro burst-size-min?                                             int16
       |  +--ro burst-size-max?                                             int16
       |  +--ro bundling-is-avail?                                          boolean
       |  +--ro bundling-group-size-max?                                    int8
       |  +--ro support-of-management-frames-without-preamble-is-avail?   boolean
       |  +--ro supported-header-compression-kind-list* [header-compression-name]
       |  |  +--ro header-compression-name           string
       |  |  +--ro header-compression-mode?           header-compression-mode-type
       |  |  +--ro compressed-protocol-layer-list*   protocol-layer-type
       |  |  +--ro mpls-payload-kind-list*           mpls-payload-kind-type
       |  |  +--ro compressed-header-length?         int16
       |  +--ro fec-is-avail?                                               boolean
       |  +--ro fec-word-size-max?                                          int16
       |  +--ro supported-fec-redundancy-size-list*                         fec-redundancy-size-
type
       |  +--ro supported-fec-interleaver-kind-list*                        fec-interleaver-
kind-type
       |  +--ro supported-fec-interleaver-depth-list*                       fec-interleaver-
depth-type
       |  +--ro encryption-is-avail?                                        boolean
       |  +--ro admin-shut-down-is-avail?                                   boolean
       |  +--ro supported-loop-back-kind-list*                              loop-back-type
       |  +--ro maintenance-timer-range?                                    string
       |  +--ro statistics-is-avail?                                        boolean
       |  +--ro supported-alarm-list*                                       string
       |  +--ro performance-monitoring-is-avail?                            boolean
       +--rw ethernet-container-configuration
       |  +--rw interface-name?                             string
       |  +--rw interface-is-on?                            boolean
       |  +--rw queue-behavior-list* [queue-name]
       |  |  +--rw queue-name                  queue-name-type
       |  |  +--rw queue-depth?                int32
       |  |  +--rw dropping-behavior-kind?    dropping-behavior-kind-type
       |  |  +--rw wred-behavior-list* [affected-drop-precedence affected-protocol]
       |  |  |  +--rw affected-drop-precedence     drop-precedence-type
       |  |  |  +--rw affected-protocol            protocol-layer-type
       |  |  |  +--rw wred-profile?               -> /core-model:control-construct/profile-
collection/profile/uuid
       |  |  +--rw scheduler-kind?           scheduler-kind-type
       |  |  +--rw weighting?                int8
       |  +--rw explicit-congestion-notification-is-on?   boolean
       |  +--rw ingress-policing-profile?                 -> /core-model:control-
construct/profile-collection/profile/uuid
       |  +--rw egress-shaping-is-on?                     boolean
       |  +--rw maximum-information-rate?                 int32
       |  +--rw maximum-burst-size?                       int16
       |  +--rw bundling-is-on?                           boolean
       |  +--rw header-compression-kind?                  -> /core-model:control-
construct/logical-termination-point/layer-protocol/ethernet-container:ethernet-container-
pac/ethernet-container-capability/supported-header-compression-kind-list/header-compression-
name
       |  +--rw fec-is-on?                                boolean
       |  +--rw fec-word-size?                            int16
       |  +--rw fec-redundancy-size?                      fec-redundancy-size-type
       |  +--rw fec-interleaver-kind?                     fec-interleaver-kind-type
       |  +--rw fec-interleaver-depth?                    fec-interleaver-depth-type
       |  +--rw encryption-is-on?                         boolean
       |  +--rw cryptographic-key?                        string
       |  +--rw loop-back-kind-on?                        loop-back-type
       |  +--rw maintenance-timer?                        int32
       |  +--rw statistics-is-on?                         boolean
       |  +--rw problem-kind-severity-list* [problem-kind-name]
       |  |  +--rw problem-kind-name         string
       |  |  +--rw problem-kind-severity?   severity-type
       |  +--rw performance-monitoring-is-on?             boolean
       +--ro ethernet-container-status
       |  +--ro interface-status?                interface-status-type
       |  +--ro bundling-is-up?                  boolean
       |  +--ro remote-site-is-faulty?           boolean
       |  +--ro loop-back-kind-up?               loop-back-type
       |  +--ro statistics-is-up?                boolean
       |  +--ro performance-monitoring-is-up?   boolean
       |  +--ro timestamp?                       yang:date-and-time
       |  +--ro last-10-sec-data-input-rate?    int32
       |  +--ro last-10-sec-data-output-rate?   int32
       |  +--ro total-bytes-input?               uint64
       |  +--ro total-bytes-output?              uint64
       |  +--ro forwarded-bytes-input?           uint64
       |  +--ro forwarded-bytes-output?          uint64
       +--ro ethernet-container-current-problems
       |  +--ro current-problem-list* [sequence-number]
       |  |  +--ro problem-name?       string
       |  |  +--ro sequence-number      int16
       |  |  +--ro timestamp?          yang:date-and-time
       |  |  +--ro problem-severity?   severity-type
       |  +--ro number-of-current-problems?   int8
       |  +--ro time-of-latest-change?        yang:date-and-time
       +--ro ethernet-container-current-performance
       |  +--ro current-performance-data-list* [granularity-period]
       |  |  +--ro performance-data
       |  |  |  +--ro tx-ethernet-bytes-max-s?    int32
       |  |  |  +--ro tx-ethernet-bytes-max-m?    int64
       |  |  |  +--ro tx-ethernet-bytes-sum?      int64
       |  |  |  +--ro queue-utilization-list* [queue-name]
       |  |  |  |  +--ro queue-name           queue-name-type
       |  |  |  |  +--ro max-queue-length?   int32
       |  |  |  |  +--ro avg-queue-length?   int32
       |  |  |  +--ro fec-corrected-blocks?       int32
       |  |  |  +--ro fec-uncorrectable-blocks?   int32
       |  |  |  +--ro time-period?                int32
       |  |  +--ro timestamp?               yang:date-and-time
       |  |  +--ro suspect-interval-flag?   boolean
       |  |  +--ro elapsed-time?            int64
       |  |  +--ro scanner-id?              string
       |  |  +--ro granularity-period       granularity-period-type
       |  +--ro number-of-current-performance-sets?   int8
       +--ro ethernet-container-historical-performances
          +--ro historical-performance-data-list* [granularity-period period-end-time]
          |  +--ro performance-data
          |  |  +--ro tx-ethernet-bytes-max-s?    int32
          |  |  +--ro tx-ethernet-bytes-max-m?    int64
          |  |  +--ro tx-ethernet-bytes-sum?      int64
          |  |  +--ro queue-utilization-list* [queue-name]
          |  |  |  +--ro queue-name          queue-name-type
          |  |  |  +--ro max-queue-length?   int32
          |  |  |  +--ro avg-queue-length?   int32
          |  |  +--ro fec-corrected-blocks?       int32
          |  |  +--ro fec-uncorrectable-blocks?   int32
          |  |  +--ro time-period?                int32
          |  +--ro suspect-interval-flag?   boolean
          |  +--ro history-data-id?         string
          |  +--ro granularity-period       granularity-period-type
          |  +--ro period-end-time          yang:date-and-time
          +--ro number-of-historical-performance-sets?   int16
          +--ro time-of-latest-change?                   yang:date-and-time
```

8.1.5. ONF TR-532 - firmware model parameters

```
module: firmware-1-0
  augment /core-model:control-construct:
    +--rw firmware-collection
       +--ro firmware-component-list* [local-id]
       |  +--ro firmware-component-pac
       |  |  +--ro firmware-component-capability
       |  |  |  +--ro firmware-component-name?            string
       |  |  |  +--ro firmware-component-version?         string
       |  |  |  +--ro firmware-component-class?           firmware-component-class-type
       |  |  |  +--ro individual-activation-is-avail?     boolean
       |  |  |  +--ro related-kinds-of-equipment-list*    string
       |  |  +--ro firmware-component-status
       |  |     +--ro firmware-component-status?             firmware-component-status-type
       |  |     +--ro firmware-component-activation-date?    yang:date-and-time
       |  |     +--ro is-active-on-equipment-list*           string
       |  +--ro subordinate-firmware-component-list*    string
       |  +--ro local-id                                string
       |  +--ro name* [value-name]
       |  |  +--ro value-name    string
       |  |  +--ro value?        string
       |  +--ro label* [value-name]
       |  |  +--ro value-name    string
       |  |  +--ro value?        string
       |  +--ro extension* [value-name]
       |  |  +--ro value-name    string
       |  |  +--ro value?        string
       |  +--ro operational-state?                      operational-state
       |  +--ro administrative-control?                 administrative-control
       |  +--ro administrative-state?                   administrative-state
       |  +--ro lifecycle-state?                        lifecycle-state
       |  +--ro address*                                dt-address
       +--ro download
          +--ro filename?                      string
          +--ro download-status?               download-status-type
          +--ro download-status-description?   string

  rpcs:
    +---x download-firmware-component
    |  +---w input
    |     +---w source-uri                 string
    |     +---w filename                   string
    |     +---w username-at-file-server?   string
    |     +---w password-at-file-server?   string
    |     +---w ssh-key?                   string
    |     +---w force-download             boolean
    +---x abort-firmware-download
    +---x activate-firmware-component
       +---w input
          +---w firmware-component         -> /core-model:control-construct/firmware:firmware-
collection/firmware-component-list/local-id
          +---w activation-delay-period    uint64
```

8.1.6. ONF TR532 - hybrid-mw-structure model parameters

```
module: hybrid-mw-structure-2-0
  augment /core-model:control-construct/core-model:logical-termination-point/core-model:layer-
protocol:
    +--rw hybrid-mw-structure-pac
       +--ro hybrid-mw-structure-capability
       |  +--ro supported-tdm-structure-kind-list* [tdm-structure-name]
       |  |  +--ro tdm-structure-name                   string
       |  |  +--ro tdm-segment-size?                    int32
       |  |  +--ro max-number-of-segments-reservable?   int8
       |  +--ro supported-alarm-list*                string
       |  +--ro performance-monitoring-is-avail?     boolean
       +--rw hybrid-mw-structure-configuration
       |  +--rw tdm-structure-kind?                      -> /core-model:control-
construct/logical-termination-point/layer-protocol/hybrid-mw-structure:hybrid-mw-structure-
pac/hybrid-mw-structure-capability/supported-tdm-structure-kind-list/tdm-structure-name
       |  +--rw number-of-tdm-segments-to-be-reserved?   int8
       |  +--rw problem-kind-severity-list* [problem-kind-name]
       |  |  +--rw problem-kind-name        string
       |  |  +--rw problem-kind-severity?   severity-type
       |  +--rw g-826-threshold-cross-alarm-list* [g-826-value-kind granularity-period]
       |  |  +--rw g-826-value-kind           g-826-type
       |  |  +--rw alarm-raising-threshold?    int32
       |  |  +--rw alarm-clearing-threshold?   int32
       |  |  +--rw granularity-period          granularity-period-type
       |  +--rw clearing-threshold-cross-alarms-is-on?   boolean
       |  +--rw performance-monitoring-is-on?            boolean
       +--ro hybrid-mw-structure-status
       |  +--ro segment-status-list* [segment-status-type-id]
       |  |  +--ro segment-status-type-id         int16
       |  |  +--ro segment-is-reserved-for-tdm?   boolean
       |  |  +--ro operational-status?            operational-state-type
       |  +--ro performance-monitoring-is-up?   boolean
       +--ro hybrid-mw-structure-current-problems
       |  +--ro current-problem-list* [sequence-number]
       |  |  +--ro problem-name?       string
       |  |  +--ro sequence-number     int16
       |  |  +--ro timestamp?          yang:date-and-time
       |  |  +--ro problem-severity?   severity-type
       |  +--ro number-of-current-problems?   int8
       |  +--ro time-of-latest-change?        yang:date-and-time
       +--ro hybrid-mw-structure-current-performance
       |  +--ro current-performance-data-list* [granularity-period]
       |  |  +--ro performance-data
       |  |  |  +--ro time-period?      int32
       |  |  |  +--ro es?               int32
       |  |  |  +--ro ses?              int32
       |  |  |  +--ro cses?             int32
       |  |  |  +--ro unavailability?   int32
       |  |  |  +--ro rx-level-min?     int8
       |  |  |  +--ro rx-level-max?     int8
       |  |  |  +--ro rx-level-avg?     int8
       |  |  +--ro timestamp?              yang:date-and-time
       |  |  +--ro suspect-interval-flag?   boolean
       |  |  +--ro elapsed-time?           int64
       |  |  +--ro scanner-id?             string
       |  |  +--ro granularity-period      granularity-period-type
       |  +--ro number-of-current-performance-sets?   int8
       +--ro hybrid-mw-structure-historical-performances
          +--ro historical-performance-data-list* [granularity-period period-end-time]
          |  +--ro performance-data
          |  |  +--ro time-period?      int32
          |  |  +--ro es?               int32
          |  |  +--ro ses?              int32
          |  |  +--ro cses?             int32
          |  |  +--ro unavailability?   int32
          |  |  +--ro rx-level-min?     int8
          |  |  +--ro rx-level-max?     int8
          |  |  +--ro rx-level-avg?     int8
          |  +--ro suspect-interval-flag?   boolean
          |  +--ro history-data-id?         string
          |  +--ro granularity-period       granularity-period-type
          |  +--ro period-end-time          yang:date-and-time
          +--ro number-of-historical-performance-sets?   int16
          +--ro time-of-latest-change?                   yang:date-and-time
```

8.1.7. ONF TR532 - ltp-augment model parameters

```
module: ltp-augment-1-0
  augment /core-model:control-construct/core-model:logical-termination-point:
    +--rw ltp-augment-pac
       +--ro ltp-augment-capability
          +--ro equipment*   -> /core-model:control-construct/equipment/uuid
          +--ro connector?   -> /core-model:control-construct/equipment/connector/local-id
```

8.1.8. ONF TR532 - mac-interface model parameters

```
module: mac-interface-1-0
  augment /core-model:control-construct/core-model:logical-termination-point/core-model:layer-
protocol:
    +--rw mac-interface-pac
       +--ro mac-interface-capability
       |  +--ro hardware-mac-address?                   string
       |  +--ro mac-address-configuration-is-avail?     boolean
       |  +--ro maximum-frame-size-min?                 int16
       |  +--ro maximum-frame-size-max?                 int16
       |  +--ro supported-frame-format-list*            frame-format-type
       |  +--ro supported-flow-control-mode-list*       flow-control-mode-type
       |  +--ro link-loss-forwarding-is-avail?          boolean
       |  +--ro broadcast-frame-suppression-is-avail?   boolean
       |  +--ro loop-port-shut-down-is-avail?           boolean
       |  +--ro loop-detection-is-avail?                boolean
       |  +--ro admin-shut-down-is-avail?               boolean
       |  +--ro supported-loop-back-kind-list*          loop-back-type
       |  +--ro maintenance-timer-range?                string
       |  +--ro statistics-is-avail?                    boolean
       |  +--ro supported-alarm-list*                   string
       |  +--ro performance-monitoring-is-avail?        boolean
       +--rw mac-interface-configuration
       |  +--rw interface-name?                      string
       |  +--rw interface-is-on?                     boolean
       |  +--rw mac-address-configuration-is-on?     boolean
       |  +--rw configured-mac-address?              string
       |  +--rw maximum-frame-size?                  int16
       |  +--rw fragmentation-allowed?               fragmentation-type
       |  +--rw transmitted-frame-format?            frame-format-type
       |  +--rw flow-control-mode?                   flow-control-mode-type
       |  +--rw link-loss-forwarding-is-on?          boolean
       |  +--rw link-loss-forwarding-delay?          int8
       |  +--rw broadcast-frame-suppression-is-on?   boolean
       |  +--rw maximum-share-of-broadcast-frames?   int8
       |  +--rw loop-port-shut-down-is-on?           boolean
       |  +--rw loop-detection-is-on?                boolean
       |  +--rw loop-back-kind-on?                   loop-back-type
       |  +--rw maintenance-timer?                   int32
       |  +--rw statistics-is-on?                    boolean
       |  +--rw problem-kind-severity-list* [problem-kind-name]
       |  |  +--rw problem-kind-name         string
       |  |  +--rw problem-kind-severity?   severity-type
       |  +--rw performance-monitoring-is-on?        boolean
       +--ro mac-interface-status
       |  +--ro interface-status?                     interface-status-type
       |  +--ro mac-address-cur?                      string
       |  +--ro received-ethernet-frame-format-cur?   frame-format-type
       |  +--ro flow-control-mode-cur?                flow-control-mode-type
       |  +--ro loop-detection-result?                loop-detection-result-type
       |  +--ro loop-back-kind-up?                    loop-back-type
       |  +--ro statistics-is-up?                     boolean
       |  +--ro performance-monitoring-is-up?         boolean
       |  +--ro timestamp?                            yang:date-and-time
       |  +--ro last-10-sec-frame-input-rate?         int32
       |  +--ro last-10-sec-frame-output-rate?        int32
       |  +--ro total-frames-input?                   int64
       |  +--ro total-frames-output?                  int64
       |  +--ro forwarded-frames-input?               int64
       |  +--ro forwarded-frames-output?              int64
       |  +--ro unicast-frames-input?                 int64
       |  +--ro unicast-frames-output?                int64
       |  +--ro multicast-frames-input?               int32
       |  +--ro multicast-frames-output?              int32
       |  +--ro broadcast-frames-input?               int32
       |  +--ro broadcast-frames-output?              int32
       |  +--ro fragmented-frames-input?              int32
       |  +--ro errored-frames-input?                 int32
       |  +--ro errored-frames-output?                int32
       |  +--ro dropped-frames-input?                 int32
       |  +--ro dropped-frames-output?                int32
       +--ro mac-interface-current-problems
       |  +--ro current-problem-list* [sequence-number]
       |  |  +--ro problem-name?       string
       |  |  +--ro sequence-number     int16
       |  |  +--ro timestamp?          yang:date-and-time
       |  |  +--ro problem-severity?   severity-type
       |  +--ro number-of-current-problems?   int8
       |  +--ro time-of-latest-change?        yang:date-and-time
       +--ro mac-interface-current-performance
       |  +--ro current-performance-data-list* [granularity-period]
       |  |  +--ro performance-data
       |  |  |  +--ro mac-control-frames-ingress?    int32
       |  |  |  +--ro mac-pause-frames-ingress?      int32
       |  |  |  +--ro oversized-frames-ingress?      int32
       |  |  |  +--ro undersized-frames-ingress?     int32
       |  |  |  +--ro jabber-frames-ingres?          int32
       |  |  |  +--ro fragmented-frames-ingress?     int64
       |  |  |  +--ro tagged-frames-ingress?         int64
       |  |  |  +--ro mac-control-frames-egress?     int32
       |  |  |  +--ro mac-pause-frames-egress?       int32
       |  |  |  +--ro tagged-frames-egress?          int64
       |  |  |  +--ro frames-of-64-byte?             int64
       |  |  |  +--ro frames-of-65-to-127-byte?      int64
       |  |  |  +--ro frames-of-128-to-255-byte?     int64
       |  |  |  +--ro frames-of-256-to-511-byte?     int64
       |  |  |  +--ro frames-of-512-to-1023-byte?    int64
       |  |  |  +--ro frames-of-1024-to-1518-byte?   int64
       |  |  +--ro timestamp?               yang:date-and-time
       |  |  +--ro suspect-interval-flag?   boolean
       |  |  +--ro elapsed-time?            int64
       |  |  +--ro scanner-id?              string
       |  |  +--ro granularity-period       granularity-period-type
       |  +--ro number-of-current-performance-sets?   int8
       +--ro mac-interface-historical-performances
          +--ro historical-performance-data-list* [granularity-period period-end-time]
          |  +--ro performance-data
          |  |  +--ro mac-control-frames-ingress?    int32
          |  |  +--ro mac-pause-frames-ingress?      int32
          |  |  +--ro oversized-frames-ingress?      int32
          |  |  +--ro undersized-frames-ingress?     int32
          |  |  +--ro jabber-frames-ingres?          int32
          |  |  +--ro fragmented-frames-ingress?     int64
          |  |  +--ro tagged-frames-ingress?         int64
          |  |  +--ro mac-control-frames-egress?     int32
          |  |  +--ro mac-pause-frames-egress?       int32
          |  |  +--ro tagged-frames-egress?          int64
          |  |  +--ro frames-of-64-byte?             int64
          |  |  +--ro frames-of-65-to-127-byte?      int64
          |  |  +--ro frames-of-128-to-255-byte?     int64
          |  |  +--ro frames-of-256-to-511-byte?     int64
          |  |  +--ro frames-of-512-to-1023-byte?    int64
          |  |  +--ro frames-of-1024-to-1518-byte?   int64
          |  +--ro suspect-interval-flag?   boolean
          |  +--ro history-data-id?         string
          |  +--ro granularity-period       granularity-period-type
          |  +--ro period-end-time          yang:date-and-time
          +--ro number-of-historical-performance-sets?   int16
          +--ro time-of-latest-change?                   yang:date-and-time
```

8.1.9. ONF TR532 - pure-ethernet-structure model parameters

```
module: pure-ethernet-structure-2-0
  augment /core-model:control-construct/core-model:logical-termination-point/core-model:layer-
protocol:
    +--rw pure-ethernet-structure-pac
       +--ro pure-ethernet-structure-capability
       |  +--ro supported-alarm-list*              string
       |  +--ro performance-monitoring-is-avail?   boolean
       +--rw pure-ethernet-structure-configuration
       |  +--rw problem-kind-severity-list* [problem-kind-name]
       |  |  +--rw problem-kind-name        string
       |  |  +--rw problem-kind-severity?   severity-type
       |  +--rw g-826-threshold-cross-alarm-list* [g-826-value-kind granularity-period]
       |  |  +--rw g-826-value-kind            g-826-type
       |  |  +--rw alarm-raising-threshold?    int32
       |  |  +--rw alarm-clearing-threshold?   int32
       |  |  +--rw granularity-period          granularity-period-type
       |  +--rw clearing-threshold-cross-alarms-is-on?   boolean
       |  +--rw performance-monitoring-is-on?            boolean
       +--ro pure-ethernet-structure-status
       |  +--ro segment-status-list* [segment-status-type-id]
       |  |  +--ro segment-status-type-id    int16
       |  |  +--ro operational-status?       operational-state-type
       |  +--ro performance-monitoring-is-up?   boolean
       +--ro pure-ethernet-structure-current-problems
       |  +--ro current-problem-list* [sequence-number]
       |  |  +--ro problem-name?       string
       |  |  +--ro sequence-number     int16
       |  |  +--ro timestamp?          yang:date-and-time
       |  |  +--ro problem-severity?   severity-type
       |  +--ro number-of-current-problems?   int8
       |  +--ro time-of-latest-change?        yang:date-and-time
       +--ro pure-ethernet-structure-current-performance
       |  +--ro current-performance-data-list* [granularity-period]
       |  |  +--ro performance-data
       |  |  |  +--ro time-period?      int32
       |  |  |  +--ro es?               int32
       |  |  |  +--ro ses?              int32
       |  |  |  +--ro cses?             int32
       |  |  |  +--ro unavailability?   int32
       |  |  |  +--ro rx-level-min?     int8
       |  |  |  +--ro rx-level-max?     int8
       |  |  |  +--ro rx-level-avg?     int8
       |  |  +--ro timestamp?                yang:date-and-time
       |  |  +--ro suspect-interval-flag?   boolean
       |  |  +--ro elapsed-time?            int64
       |  |  +--ro scanner-id?              string
       |  |  +--ro granularity-period       granularity-period-type
       |  +--ro number-of-current-performance-sets?   int8
       +--ro pure-ethernet-structure-historical-performances
          +--ro historical-performance-data-list* [granularity-period period-end-time]
          |  +--ro performance-data
          |  |  +--ro time-period?      int32
          |  |  +--ro es?               int32
          |  |  +--ro ses?              int32
          |  |  +--ro cses?             int32
          |  |  +--ro unavailability?   int32
          |  |  +--ro rx-level-min?     int8
          |  |  +--ro rx-level-max?     int8
          |  |  +--ro rx-level-avg?     int8
          |  +--ro suspect-interval-flag?   boolean
          |  +--ro history-data-id?         string
          |  +--ro granularity-period       granularity-period-type
          |  +--ro period-end-time          yang:date-and-time
          +--ro number-of-historical-performance-sets?   int16
          +--ro time-of-latest-change?                   yang:date-and-time
```

8.1.10. ONF TR532 - tdm-container model parameters

```
module: tdm-container-2-0
  augment /core-model:control-construct/core-model:logical-termination-point/core-model:layer-
protocol:
    +--rw tdm-container-pac
       +--ro tdm-container-capability
       |  +--ro supported-tdm-container-kind-list* [tdm-container-name]
       |  |  +--ro tdm-container-name    string
       |  |  +--ro tdm-container-size?   int32
       |  +--ro admin-shut-down-is-avail?            boolean
       |  +--ro supported-loop-back-kind-list*       loop-back-type
       |  +--ro maintenance-timer-range?             string
       |  +--ro supported-alarm-list*                string
       |  +--ro performance-monitoring-is-avail?     boolean
       +--rw tdm-container-configuration
       |  +--rw interface-name?                 string
       |  +--rw interface-is-on?                boolean
       |  +--rw tdm-container-kind?             -> /core-model:control-construct/logical-
termination-point/layer-protocol/tdm-container:tdm-container-pac/tdm-container-
capability/supported-tdm-container-kind-list/tdm-container-name
       |  +--rw segment-number?                 int16
       |  +--rw loop-back-kind-on?              loop-back-type
       |  +--rw maintenance-timer?              int32
       |  +--rw problem-kind-severity-list* [problem-kind-name]
       |  |  +--rw problem-kind-name          string
       |  |  +--rw problem-kind-severity?     severity-type
       |  +--rw performance-monitoring-is-on?   boolean
       +--ro tdm-container-status
       |  +--ro interface-status?               interface-status-type
       |  +--ro loop-back-kind-up?              loop-back-type
       |  +--ro statistics-is-up?               boolean
       |  +--ro performance-monitoring-is-up?   boolean
       +--ro tdm-container-current-problems
       |  +--ro current-problem-list* [sequence-number]
       |  |  +--ro problem-name?       string
       |  |  +--ro sequence-number     int16
       |  |  +--ro timestamp?          yang:date-and-time
       |  |  +--ro problem-severity?   severity-type
       |  +--ro number-of-current-problems?   int8
       |  +--ro time-of-latest-change?        yang:date-and-time
       +--ro tdm-container-current-performance
       |  +--ro current-performance-data-list* [granularity-period]
       |  |  +--ro performance-data?        container-performance-type
       |  |  +--ro timestamp?               yang:date-and-time
       |  |  +--ro suspect-interval-flag?   boolean
       |  |  +--ro elapsed-time?            int64
       |  |  +--ro scanner-id?              string
       |  |  +--ro granularity-period       granularity-period-type
       |  +--ro number-of-current-performance-sets?   int8
       +--ro tdm-container-historical-performances
          +--ro historical-performance-data-list* [granularity-period period-end-time]
          |  +--ro performance-data?        container-performance-type
          |  +--ro suspect-interval-flag?   boolean
          |  +--ro history-data-id?         string
          |  +--ro granularity-period       granularity-period-type
          |  +--ro period-end-time          yang:date-and-time
          +--ro number-of-historical-performance-sets?   int16
          +--ro time-of-latest-change?                  yang:date-and-time
```

8.1.11. ONF TR-532 - vlan-fc model parameters

```
module: vlan-fc-1-0
  augment /core-model:control-construct/core-model:forwarding-domain/core-model:fc:
    +--rw vlan-fc-pac
       +--ro vlan-fc-capability
       |  +--ro supported-sub-layer-protocol-name-list*   sub-layer-protocol-name-type
       +--rw vlan-fc-configuration
          +--rw fc-name?                   string
          +--rw sub-layer-protocol-name?   sub-layer-protocol-name-type
          +--rw vlan-id?                   int64

  rpcs:
    +---x create-vlan-fc-port
    |  +---w input
    |  |  +---w affected-vlan-fc?            -> /core-model:control-construct/forwarding-
domain/fc/uuid
    |  |  +---w associated-vlan-interface?   -> /core-model:control-construct/logical-
termination-point/uuid
    |  +--ro output
    |     +--ro created-vlan-fc-port?   string
    +---x delete-vlan-fc-port
       +---w input
          +---w affected-vlan-fc?          -> /core-model:control-construct/forwarding-
domain/fc/uuid
          +---w obsolete-vlan-interface?   -> /core-model:control-construct/logical-
termination-point/uuid
```

8.1.12. ONF TR-532 - vlan-fd model parameters

```
module: vlan-fd-1-0
  augment /core-model:control-construct/core-model:forwarding-domain:
    +--rw vlan-fd-pac
       +--ro vlan-fd-capability
       |  +--ro supported-sub-layer-protocol-name-list*        sub-layer-protocol-name-type
       |  +--ro component-id?                                  int32
       |  +--ro extended-filtering-is-avail?                   boolean
       |  +--ro traffic-classes-is-avail?                      boolean
       |  +--ro static-entries-on-individual-ports-is-avail?   boolean
       |  +--ro independent-vlan-learning-is-avail?            boolean
       |  +--ro shared-vlan-learning-is-avail?                 boolean
       |  +--ro hybrid-vlan-learning-is-avail?                 boolean
       |  +--ro configurable-port-vlan-id-tagging-is-avail?    boolean
       |  +--ro multiple-local-bridges-is-avail?               boolean
       |  +--ro supported-version?                             int16
       |  +--ro maximum-number-of-vlan-ids?                    int16
       |  +--ro overriding-default-port-vlan-id-is-avail?      boolean
       |  +--ro protocol-frame-format?                         protocol-frame-format-type
       |  +--ro maximum-number-of-msti?                        int16
       |  +--ro port-and-protocol-based-vlan-is-avail?         boolean
       +--rw vlan-fd-configuration
       |  +--rw fd-name?                    string
       |  +--rw sub-layer-protocol-name?    sub-layer-protocol-name-type
       |  +--rw mac-address?                string
       |  +--rw traffic-classes-is-on?      boolean
       |  +--rw protocol-group-list* [db-index]
       |     +--rw db-index                  uint16
       |     +--rw protocol-group-id?        int32
       |     +--rw protocol-frame-format?    protocol-frame-format-type
       |     +--rw ethertype?                string
       |     +--rw protocol-id?              string
       |     +--rw llc-address-list*         string
       +--ro vlan-fd-status
          +--ro mac-address-cur?                              string
          +--ro number-of-ports-cur?                          int16
          +--ro number-of-static-vlan-registrations-cur?      int32
          +--ro number-of-dynamic-vlan-registrations-cur?     int32
          +--ro fd-status?                                    fd-status-type

  rpcs:
    +---x create-vlan-fc
    |  +---w input
    |  |  +---w affected-vlan-fd?   -> /core-model:control-construct/forwarding-domain/uuid
    |  |  +---w new-vlan-id?        uint64
    |  +--ro output
    |     +--ro created-vlan-fc?   -> /core-model:control-construct/forwarding-domain/fc/uuid
    +---x delete-vlan-fc
       +---w input
          +---w affected-vlan-fd?   -> /core-model:control-construct/forwarding-domain/uuid
          +---w obsolete-vlan-fc?   -> /core-model:control-construct/forwarding-domain/fc/uuid
```

8.1.13. ONF TR-532 - vlan-interface model parameters

```
module: vlan-interface-1-0
  augment /core-model:control-construct/core-model:logical-termination-point/core-model:layer-
protocol:
    +--rw vlan-interface-pac
       +--ro vlan-interface-capability
       |  +--ro supported-sub-layer-protocol-name-list*                 sub-layer-protocol-
name-type
       |  +--ro supported-interface-kind-list*                          interface-kind-type
       |  +--ro tagging-and-mvrp-is-avail?                              boolean
       |  +--ro configuring-ingress-tag-filtering-is-avail?             boolean
       |  +--ro ingress-vlan-id-filtering-is-avail?                     boolean
       |  +--ro available-pcp-bits-interpretation-kind-list*            pcp-bits-
interpretation-kind-type
       |  +--ro configuring-pcp-bits-decoding-is-avail?                 boolean
       |  +--ro configuring-pcp-bits-encoding-is-avail?                 boolean
       |  +--ro drop-eligible-indicator-is-avail?                       boolean
       |  +--ro number-of-available-priorities?                         int8
       |  +--ro received-priority-overwriting-is-avail?                 boolean
       |  +--ro vlan-id-translation-is-avail?                           boolean
       |  +--ro egress-vlan-id-translation-is-avail?                    boolean
       |  +--ro port-and-protocol-based-vlan-is-avail?                  boolean
       |  +--ro max-number-of-protocol-vlan-id-groupings?               int16
       |  +--ro service-access-priority-tagging-is-avail?               boolean
       |  +--ro configuring-service-access-priority-mapping-is-avail?   boolean
       |  +--ro number-of-available-traffic-classes?                    int8
       |  +--ro restricted-automated-vlan-registration-is-avail?        boolean
       |  +--ro admin-shut-down-is-avail?                               boolean
       |  +--ro statistics-is-avail?                                    boolean
       +--rw vlan-interface-configuration
       |  +--rw interface-name?                                 string
       |  +--rw sub-layer-protocol-name?                        sub-layer-protocol-name-type
       |  +--rw interface-kind?                                 interface-kind-type
       |  +--rw default-vlan-id?                                int64
       |  +--rw default-priority?                               int8
       |  +--rw ingress-tag-filtering?                          ingress-tag-filtering-type
       |  +--rw ingress-vlan-id-filtering-is-on?                boolean
       |  +--rw pcp-bits-interpretation-kind?                   pcp-bits-interpretation-kind-
type
       |  +--rw pcp-bit-to-priority-mapping-list* [to-be-decoded-pcp-bits-value]
       |  |  +--rw to-be-decoded-pcp-bits-value    int8
       |  |  +--rw associated-priority-value?      int8
       |  |  +--rw associated-drop-eligibility?    boolean
       |  +--rw pcp-bits-encoding-mapping-list* [to-be-encoded-priority-value to-be-encoded-
drop-eligibility]
       |  |  +--rw to-be-encoded-priority-value      int8
       |  |  +--rw to-be-encoded-drop-eligibility    boolean
       |  |  +--rw associated-pcp-bits-value?        int8
       |  +--rw drop-eligible-indicator-is-on?                  boolean
       |  +--rw drop-eligible-encoding-is-required?             boolean
       |  +--rw received-priority-overwriting-is-on?            boolean
       |  +--rw received-priority-overwriting-list* [to-be-overwritten-priority-value]
       |  |  +--rw to-be-overwritten-priority-value    int8
       |  |  +--rw new-priority-value?                 int8
       |  +--rw vlan-id-translation-is-on?                      boolean
       |  +--rw external-to-internal-vlan-id-mapping-list* [external-vlan-id]
       |  |  +--rw external-vlan-id    int16
       |  |  +--rw internal-vlan-id?   int16
       |  +--rw egress-vlan-id-translation-is-on?               boolean
       |  +--rw internal-to-egress-vlan-id-mapping-list* [internal-vlan-id]
       |  |  +--rw internal-vlan-id    int16
       |  |  +--rw egress-vlan-id?     int16
       |  +--rw forwarded-protocol-vlan-id-grouping-list* [forwarded-protocol-group-id]
       |  |  +--rw forwarded-protocol-group-id    int32
       |  |  +--rw forwarded-vlan-id-list*        int64
       |  +--rw service-access-priority-tagging-is-on?          boolean
       |  +--rw service-access-priority-mapping-list* [c-vlan-priority-value]
       |  |  +--rw c-vlan-priority-value    int8
       |  |  +--rw s-vlan-pcp-bits-value?   int8
       |  +--rw priority-to-traffic-class-mapping-list* [priority-value]
       |  |  +--rw priority-value         int8
       |  |  +--rw traffic-class-value?   int8
       |  +--rw restricted-automated-vlan-registration-is-on?   boolean
       |  +--rw admin-point-to-point?                           admin-point-to-point-type
       |  +--rw statistics-is-on?                               boolean
       +--ro vlan-interface-status
          +--ro interface-status?      interface-status-type
          +--ro statistics-is-up?      boolean
          +--ro timestamp?             yang:date-and-time
          +--ro total-bytes-input?     int64
          +--ro total-frames-input?    int64
          +--ro total-bytes-output?    int64
          +--ro total-frames-output?   int64
```

8.1.14. ONF TR-532 - wire-interface model parameters

```
module: wire-interface-2-0
  augment /core-model:control-construct/core-model:logical-termination-point/core-model:layer-
protocol:
    +--rw wire-interface-pac
       +--ro wire-interface-capability
       |  +--ro supported-pmd-kind-list* [pmd-name]
       |  |  +--ro pmd-name    string
       |  |  +--ro speed?      string
       |  |  +--ro duplex?     duplex-type
       |  +--ro auto-pmd-negotiation-is-avail?                           boolean
       |  +--ro auto-negotiation-pmd-selection-is-avail?                 boolean
       |  +--ro supported-signal-ordering-kind-list*                     signal-ordering-
kind-type
       |  +--ro auto-signal-ordering-is-avail?                           boolean
       |  +--ro configuration-of-rx-sync-preference-is-avail?            boolean
       |  +--ro mii-kind?                                                mii-kind-type
       |  +--ro mdi-kind?                                                mdi-kind-type
       |  +--ro required-medium-kind?                                    medium-kind-type
       |  +--ro wavelength-min-list*                                     int32
       |  +--ro wavelength-max-list*                                     int32
       |  +--ro wavelength-grid-min?                                     int32
       |  +--ro short-reach-mode-is-avail?                               boolean
       |  +--ro eee-is-avail?                                            boolean
       |  +--ro unidirectional-operation-is-avail?                       boolean
       |  +--ro rxlevel-low-threshold?                                   int8
       |  +--ro rxlevel-high-threshold?                                  int8
       |  +--ro temperature-low-threshold?                               int8
       |  +--ro temperature-high-threshold?                              int8
       |  +--ro configuration-of-number-of-bip-errors-per-ses-is-avail?  boolean
       |  +--ro admin-shut-down-is-avail?                                boolean
       |  +--ro isolation-is-avail?                                      boolean
       |  +--ro supported-loop-back-kind-list*                           loop-back-type
       |  +--ro maintenance-timer-range?                                 string
       |  +--ro supported-alarm-list*                                    string
       |  +--ro performance-monitoring-is-avail?                         boolean
       +--rw wire-interface-configuration
       |  +--rw interface-name?                   string
       |  +--rw interface-is-on?                  boolean
       |  +--rw remote-wire-interface-name?       string
       |  +--rw transceiver-is-on-list*           boolean
       |  +--rw auto-pmd-negotiation-is-on?       boolean
       |  +--rw fixed-pmd-kind?                   -> /core-model:control-construct/logical-
termination-point/layer-protocol/wire-interface:wire-interface-pac/wire-interface-
capability/supported-pmd-kind-list/pmd-name
       |  +--rw auto-negotiation-pmd-list*        -> /core-model:control-construct/logical-
termination-point/layer-protocol/wire-interface:wire-interface-pac/wire-interface-
capability/supported-pmd-kind-list/pmd-name
       |  +--rw auto-signal-ordering-is-on?       boolean
       |  +--rw fixed-signal-ordering-kind?       signal-ordering-kind-type
       |  +--rw wavelength-list*                  int32
       |  +--rw rx-sync-preference?               rx-sync-preference-type
       |  +--rw short-reach-mode-is-on?           boolean
       |  +--rw eee-is-on?                        boolean
       |  +--rw unidirectional-operation-is-on?   boolean
       |  +--rw number-of-bip-errors-per-ses?     int16
       |  +--rw restart-pmd-negotiation-is-on?    boolean
       |  +--rw isolation-is-on?                  boolean
       |  +--rw loop-back-kind-on?                loop-back-type
       |  +--rw maintenance-timer?                int32
       |  +--rw problem-kind-severity-list* [problem-kind-name]
       |  |  +--rw problem-kind-name        string
       |  |  +--rw problem-kind-severity?   severity-type
       |  +--rw performance-monitoring-is-on?     boolean
       +--ro wire-interface-status
       |  +--ro interface-status?                interface-status-type
       |  +--ro receive-signal-is-detected-list*   boolean
       |  +--ro pmd-negotiation-state?           pmd-negotiation-state-type
       |  +--ro pmd-is-up?                       boolean
       |  +--ro pmd-kind-cur?                    -> /core-model:control-construct/logical-
termination-point/layer-protocol/wire-interface:wire-interface-pac/wire-interface-
capability/supported-pmd-kind-list/pmd-name
       |  +--ro signal-ordering-kind-cur?          signal-ordering-kind-type
       |  +--ro rx-sync-role?                    rx-sync-role-type
       |  +--ro eee-is-up?                       boolean
       |  +--ro link-is-up?                      boolean
       |  +--ro link-is-idle?                    boolean
       |  +--ro loop-back-kind-up?               loop-back-type
       |  +--ro tx-level-cur?                    int8
       |  +--ro rx-level-cur-list*               int8
       |  +--ro performance-monitoring-is-up?      boolean
       +--ro wire-interface-current-problems
       |  +--ro current-problem-list* [sequence-number]
       |  |  +--ro problem-name?       string
       |  |  +--ro sequence-number     int16
       |  |  +--ro timestamp?          yang:date-and-time
       |  |  +--ro problem-severity?   severity-type
       |  +--ro number-of-current-problems?   int8
       |  +--ro time-of-latest-change?        yang:date-and-time
       +--ro wire-interface-current-performance
       |  +--ro current-performance-data-list* [granularity-period]
       |  |  +--ro performance-data
       |  |  |  +--ro es?                            int32
       |  |  |  +--ro ses?                           int32
       |  |  |  +--ro symbol-error-during-carrier?     int32
       |  |  |  +--ro low-power-idle-transmitter-ms?   int32
       |  |  |  +--ro low-power-idle-receiver-ms?      int32
       |  |  +--ro timestamp?              yang:date-and-time
       |  |  +--ro suspect-interval-flag?   boolean
       |  |  +--ro elapsed-time?           int64
       |  |  +--ro scanner-id?             string
       |  |  +--ro granularity-period      granularity-period-type
       |  +--ro number-of-current-performance-sets?   int8
       +--ro wire-interface-historical-performances
          +--ro historical-performance-data-list* [granularity-period period-end-time]
          |  +--ro performance-data
          |  |  +--ro es?                            int32
          |  |  +--ro ses?                           int32
          |  |  +--ro symbol-error-during-carrier?     int32
          |  |  +--ro low-power-idle-transmitter-ms?   int32
          |  |  +--ro low-power-idle-receiver-ms?      int32
          |  +--ro suspect-interval-flag?   boolean
          |  +--ro history-data-id?         string
          |  +--ro granularity-period      granularity-period-type
          |  +--ro period-end-time         yang:date-and-time
          +--ro number-of-historical-performance-sets?   int16
          +--ro time-of-latest-change?                   yang:date-and-time
```

8.1.15. ONF TR-532 - wred-profile model parameters

```
module: wred-profile-1-0
  augment /core-model:control-construct/core-model:profile-collection/core-model:profile:
    +--rw wred-profile-pac
       +--ro wred-profile-capability
       |  +--ro available-buffer-size?                     int32
       |  +--ro drop-probability-at-threshold-low-is-avail?   boolean
       |  +--ro gentle-wred-is-avail?                      boolean
       |  +--ro sensitivity-setting-is-avail?              boolean
       |  +--ro coloring-is-avail?                         boolean
       +--rw wred-profile-configuration
          +--rw profile-name?                         string
          +--rw threshold-low?                        int32
```

```
+--rw drop-probability-at-threshold-low?    int8
+--rw threshold-high?                       int32
+--rw drop-probability-at-threshold-high?   int8
+--rw gentle-wred-is-on?                    boolean
+--rw threshold-gentle?                     int32
+--rw sensitivity?                          int8
+--rw coloring-is-on?                       boolean
```

8.2. OpenConfig Examples

8.2.1. OpenConfig query to get the inventory details for all the components

The below XML query is to get all components and subcomponents of an Infinera DRX-30 stacked node.

Request:

```
<rpc
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="">
  <get>
    <filter>
      <components
        xmlns=http://openconfig.net/yang/platform>
        <component/>
      </components>
    </filter>
  </get></rpc>]]>]]>
```

Reply:

```
  <rpc-reply
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="">
    <data>
      <components
        xmlns=http://openconfig.net/yang/platform>
        <component>
          <name>DRX-30</name>
          <config>
            <name>DRX-30</name>
          </config>
          <state>
            <type>CHASSIS</type>
            <name>DRX-30</name>
            <id>0</id>
            <description>x86_64-accton_as7315_27xb-r0</description>
          </state>
        </component>
        ...
        <component>
          <name>unit1</name>
          <config>
            <name>unit1</name>
          </config>
          <state>
            <type>CONTROLLER_CARD</type>
            <name>unit1</name>
            <id>unit1</id>
            <location>unit1</location>
            <description>cuUnit</description>
            <mfg-date>2019-6-11, 2:23:52.0, +0:00</mfg-date>
            <software-version>2.4.7</software-version>
            <serial-no>731527XB1924006</serial-no>
            <part-no>VQS-X30CHASS-0A</part-no>
            <oper-status>ACTIVE</oper-status>
            <parent>DRX-30</parent>
        </state>
        ...
    </component>

    <component>
        <name>unit13</name>
        <config>
            <name>unit13</name>
        </config>
        <state>
            <type>LINECARD</type>
            <name>unit13</name>
            <id>unit13</id>
            <location>unit13</location>
            <description>luUnit</description>
            <mfg-date>2019-6-11, 13:52:13.0, +0:00</mfg-date>
            <software-version>2.4.7</software-version>
            <serial-no>731527XB1924001</serial-no>
            <part-no>VQS-X30CHASS-0A</part-no>
            <oper-status>ACTIVE</oper-status>
            <parent>DRX-30</parent>
        </state>
        <subcomponents>
            <subcomponent>
                <name>unit13:0</name>
                <config>
                    <name>unit13:0</name>
                </config>
                <state>
                    <name>unit13:0</name>
                </state>
            </subcomponent>
            ...
        </subcomponents>
        <linecard
            xmlns=http://openconfig.net/yang/platform/linecard>
            <config>
                <power-admin-state/>
            </config>
            <state>
                <power-admin-state>POWER_ENABLED</power-admin-state>
                <slot-id>unit13</slot-id>
            </state>
        </linecard>
    </component>
    ...
    <component>
        <name>13/0/0</name>
        <config>
            <name>13/0/0</name>
        </config>
        <state>
            <type>PORT</type>
            <name>13/0/0</name>
            <id>13/0/0</id>
            <location>13/0/0</location>
            <oper-status>ACTIVE</oper-status>
            <parent>unit13:0</parent>
            <subcomponents>
              <subcomponent>
                <name>13/0/0-Transceiver</name>
                <config>
                  <name>13/0/0-Transceiver</name>
                </config>
                <state>
                  <name>13/0/0-Transceiver</name>
                </state>
              </subcomponent>
            </subcomponents>
          </state>
          <port
            xmlns=http://openconfig.net/yang/platform/port>
            <breakout-mode>
              <config>
                <channel-speed>100GbitPerSec</channel-speed>
                <num-channels>1</num-channels>
              </config>
              <state>
                <channel-speed>100GbitPerSec</channel-speed>
                <num-channels>1</num-channels>
              </state>
            </breakout-mode>
          </port>
        </component>
        ...
        <component>
          <name>13/0/0-Transceiver</name>
          <config>
            <name>13/0/0-Transceiver</name>
          </config>
          <state>
            <type>TRANSCEIVER</type>
            <name>13/0/0-Transceiver</name>
            <id>13/0/0</id>
            <location>13/0/0</location>
            <removable>true</removable>
            <empty>true</empty>
            <parent>13/0/0</parent>
          </state>
          <transceiver
            xmlns=http://openconfig.net/yang/platform/transceiver>
            <config>
              <enabled>TRUE</enabled>
            </config>
            <state>
              <enabled>TRUE</enabled>
              <present>NOT_PRESENT</present>
            </state>
          </transceiver>
        </component>
        ...
      </components>
    </data></rpc-reply>]]>]]>
```

8.2.2. OpenConfig query to get details for all the interfaces

The below XML query is to get all interfaces in a DRX-30 stacked node.

Request:

```
<rpc
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="">
    <get>
        <filter type="subtree">
            <oc-if:interfaces
                xmlns:oc-if=http://openconfig.net/yang/interfaces>
                <oc-if:interface/>
            </oc-if:interfaces>
        </filter>
    </get></rpc>]]>]]>
```

Reply:

```
    <rpc-reply
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="">
        <data>
            <interfaces
                xmlns=http://openconfig.net/yang/interfaces>
                <interface>
                    <name>13/0/0</name>
                    <config>
                        <name>13/0/0</name>
                        <type
                            xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-
type">ianaift:ethernetCsmacd
                        </type>
                        <mtu>1530</mtu>
                        <loopback-mode>false</loopback-mode>
                        <description/>
                        <enabled>false</enabled>
                    </config>
                    <state>
                        <name>13/0/0</name>
                        <type
                            xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-
type">ianaift:ethernetCsmacd
                        </type>
                        <mtu>1530</mtu>
                        <loopback-mode/>
                        <description/>
                        <enabled/>
                        <ifindex/>
                        <admin-status/>
                        <oper-status/>
                        <logical/>
                        <counters>
                            <in-octets>0</in-octets>
                            <in-pkts>0</in-pkts>
                            <in-unicast-pkts>0</in-unicast-pkts>
                            <in-broadcast-pkts>0</in-broadcast-pkts>
                            <in-multicast-pkts>0</in-multicast-pkts>
                            <in-discards>0</in-discards>
                            <in-errors>0</in-errors>
                            <in-unknown-protos>0</in-unknown-protos>
                            <in-fcs-errors>0</in-fcs-errors>
                            <out-octets>0</out-octets>
                            <out-pkts>0</out-pkts>
                            <out-unicast-pkts>0</out-unicast-pkts>
                            <out-broadcast-pkts>0</out-broadcast-pkts>
                            <out-multicast-pkts>0</out-multicast-pkts>
                            <out-discards>0</out-discards>
                            <out-errors/>
                        </counters>
                    </state>
```

```
<subinterfaces>
    <subinterface>
        <index>0</index>
        <config>
            <index>0</index>
        </config>
        <ipv4
            xmlns=http://openconfig.net/yang/interfaces/ip>
            <addresses>
                <address>
                    <ip>44.44.44.1</ip>
                    <config>
                        <ip>44.44.44.1</ip>
                        <prefix-length>24</prefix-length>
                    </config>
                    <state>
                        <ip>44.44.44.1</ip>
                        <prefix-length>24</prefix-length>
                        <origin>STATIC</origin>
                    </state>
                </address>
            </addresses>
            <config>
                <mtu>useLayer2Mtu</mtu>
            </config>
            <state>
                <enabled>false</enabled>
                <mtu>useLayer2Mtu</mtu>
                <dhcp-client/>
                <counters>
                    <in-pkts>0</in-pkts>
                    <in-octets>0</in-octets>
                    <in-error-pkts>0</in-error-pkts>
                    <in-unknown-proto-pkts>0</in-unknown-proto-pkts>
                    <in-forwarded-pkts/>
                    <in-forwarded-octets/>
                    <in-discarded-pkts/>
                    <out-pkts>0</out-pkts>
                    <out-octets>0</out-octets>
                    <out-error-pkts>0</out-error-pkts>
                    <out-forwarded-pkts/>
                    <out-forwarded-octets/>
                    <out-discarded-pkts>0</out-discarded-pkts>
                </counters>
            </state>
        </ipv4>
    </subinterface>
    <subinterface>
        <index>13/0/0.1</index>
        <config>
            <index>13/0/0.1</index>
        </config>
        <vlan
            xmlns=http://openconfig.net/yang/vlan>
            <config>
                <vlan-id>13/0/0.1</vlan-id>
            </config>
            <state>
                <vlan-id>13/0/0.1</vlan-id>
            </state>
        </vlan>
        <ipv4
            xmlns=http://openconfig.net/yang/interfaces/ip>
            <addresses>
```

```
                            <address>
                                <ip>45.45.45.1</ip>
                                <config>
                                    <ip>45.45.45.1</ip>
                                    <prefix-length>24</prefix-length>
                                </config>
                                <state>
                                    <ip>45.45.45.1</ip>
                                    <prefix-length>24</prefix-length>
                                    <origin>STATIC</origin>
                                </state>
                            </address>
                        </addresses>
                        <config>
                            <mtu>useLayer2Mtu</mtu>
                        </config>
                        <state>
                            <enabled>false</enabled>
                            <mtu>useLayer2Mtu</mtu>
                            <dhcp-client/>
                            <counters>
                                <in-pkts/>
                                <in-octets/>
                                <in-error-pkts/>
                                <in-unknown-proto-pkts/>
                                <in-forwarded-pkts/>
                                <in-forwarded-octets/>
                                <in-discarded-pkts/>
                                <out-pkts/>
                                <out-octets/>
                                <out-error-pkts/>
                                <out-forwarded-pkts/>
                                <out-forwarded-octets/>
                                <out-discarded-pkts/>
                            </counters>
                        </state>
                    </ipv4>
                </subinterface>
            </subinterfaces>
        </interface>
        ...
    </interfaces>
</data></rpc-reply>]]>]]>
```

References

[1] Vilalta R, de la Cruz JL, López-de-Lerma AM, López V, Martínez R, Casellas R, Muñoz R. uABNO: A Cloud-Native Architecture for Optical SDN Controllers. In2020 Optical Fiber Communications Conference and Exhibition (OFC) 2020 Mar 8 (pp. 1-3). IEEE.

[2] Qin Wu, Igor Bryskin, Henk Birkholz, Xufeng Liu, Benoit Claise, "A YANG Data model for ECA Policy Management", IEFT draft NETMOD Working Group, February 19, 2021. Work in progress. Available from: https://datatracker.ietf.org/doc/html/draft-ietf-netmod-eca-policy

[3] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. 2008. OpenFlow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. 38, 2 (April 2008), 69–74. DOI: https://doi.org/10.1145/1355734.1355746

[4] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. 2014. P4: programming protocol-independent packet processors. SIGCOMM Comput. Commun. Rev. 44, 3 (July 2014), 87–95. DOI: https://doi.org/10.1145/2656877.2656890

[5] Open Networking Foundation (ONF): https://opennetworking.org/

[6] ONF Stratum OS: https://opennetworking.org/stratum/

[7] ONF Open Network Operating System (ONOS): https://opennetworking.org/onos/

[8] H2020 EU TeraFlow project, Milestone 3.1

[9] H2020 EU TeraFlow project, Milestone 3.2

[10] ONF TR-532 models: https://github.com/openBackhaul/Overview

[11] Farrel, A., Gray, E., Drake, J., Rokui, R., Homma, S., Makhijani, K., Contreras, L. M., and J. Tantsura, "Framework for IETF Network Slices", IETF draft TEAS Working Group, August 2021. Work in progress. Available from: https://datatracker.ietf.org/doc/draft-ietf-teas-ietf-network-slices

[12] Wu, Q., Liu, W., and A. Farrel, "Service Models Explained", RFC 8309, DOI 10.17487/RFC8309, January 2018, Available from https://www.rfc-editor.org/info/rfc8309

[13] X. Liu, et al., "IETF Network Slice YANG Data Model", IETF draft TEAS Working Group, July 2021. Work in progress. Available from https://datatracker.ietf.org/doc/draft-liu-teas-transport-network-slice-yang/.

[14] A. Alcalá, S. Barguil, V. López, L. M. Contreras, C. Manso, P. Alemany, R. Casellas, R. Martínez, D. Gonzalez-Perez, X. Liu, J.M. Pulido, J.P. Fernandez-Palacios, R. Muñoz, R. Vilalta, Multi-layer Transport Network Slicing with Hard and Soft Isolation , in Proceedings of The Optical Networking and Communication Conference & Exhibition (OFC), 6-11 June 2021, virtual event.

[15] R. Enns, M. Bjorklund, J. Schoenwaelder, A. Bierman, "Network Configuration Protocol (NETCONF)," IETF RFC6241, June 2011, Available from https://datatracker.ietf.org/doc/html/rfc6241

[16] OpenConfig, November 2021, https://www.openconfig.net/

[17] Transport API OpenAPI Specification. Available: https://github.com/OpenNetworkingFoundation/TAPI/tree/develop/OAS