INFORMATIONAL REFERENCE DESIGN

Open Disaggregated Transport Network (ODTN)

ONF Info-1002

Version 0.4 | March 2019

About ONF Reference Designs

Reference Designs (RDs) represent a particular assembly of components that are required to build a deployable platform. They are "blueprints" developed by ONF's Operator members to address specific use cases for the emerging edge cloud.

RDs are the vehicles to describe how a collection of projects can be assembled into a platform to address specific needs of operators. By defining RDs, ONF's operator members are showing the industry the path forward to solutions they plan to procure and deploy.

Each RD is backed by specific Operator partner(s) who plan to deploy these designs into their production networks and will include participation from invited supply chain partners sharing the vision and demonstrating active investment in building open source solutions. The RD thus enables a set of committed partners to work on the specification and a related open source platform.

Assembling the set of selected components defined by the RDs into a platform enables a proof-of-concept to allow the test and trial of the design. These platforms are called Exemplar Platforms and each of them will be based on a Reference Design and will serve as reference implementations. These platforms are designed to make it easy to download, modify, trial and deploy an operational instantiation and thereby speed up adoption and deployment.

About the Open Networking Foundation

The Open Networking Foundation (ONF) is an operator led consortium spearheading disruptive network transformation. Now the recognized leader for open source solutions for operators, the ONF first launched in 2011 as the standard bearer for Software Defined Networking (SDN). Led by its operator partners AT&T, China Unicom, Comcast, Deutsche Telekom, Google, NTT Group and Turk Telekom, the ONF is driving vast transformation across the operator space. For further information visit http://www.opennetworking.org

Disclaimer

Open Disaggregated Transport Network (ODTN)

Informational Reference Design v0.4

ONF Info-1002

Document Revision Date: March 27, 2019

Document Revision Number: v0.4

Document Release Status: Informational

This Reference Design has been published as "Informational" by the Open Networking Foundation. Informational Reference Designs represent work is not yet ready for operator field trials and deployments, but represents an important line of investigation that the ONF leadership would still like to release to the public. It may be the case that an open source Exemplar Platform is not yet available, or that the work has not yet sufficiently matured to be consumable in production environments. Public release is intended to enable and spur community participation, and all feedback and contributions are encouraged to help further this work.

This Informational Reference Design is not considered an ONF "Final Specification".

This specification was authored by an operator-led Reference Design Team (RDT) composed of experts from:

Operator Group:

NTT Communications and Telefonica

ONF Supply-Chain Partners:

Ciena and Edgecore

Contributors

NTT: Dai Kashiwa (RDT Lead), Hiroki Okui, Wenyu Shen

Telefonica: Arturo Mayoral, Victor Lopez, Oscar Gonzalez De dios

Edgecore: Dileep Kuchhangi

Ciena: Lydon Ong, Nigel Davis

ONF: Andrea Campanella

Document Revision History

Date	Revision	Description
08/23/18	Draft 0.1	Initial Draft
09/18/18	Draft 0.2	Publication Ready first version
09/24/18	Draft 0.3	approved by TLT
12/04/18	Release 0.3	release to members
03/18/18	Release 0.4	release as informational RD

TABLE OF CONTENTS

1 INTRODUCTION

This Open Networking Foundation (ONF) Reference Design (RD) describes Open Disaggregated Transport Network (ODTN). ODTN is intended to open the optical network for choice and innovation by disaggregating the components of the network and by providing open software that can control a multi-vendor assembly of different components. The ODTN reference design provides a high level template and architecture that aims to deliver a number of benefits for the operator metro and core optical networks such as cost reduction, absence of vendor lock-in, flexibility and less time to deployment of new features and solutions. In addition to the ODTN reference design, ONF intends to develop an exemplar implementation of ODTN.

1.2 OPERATOR INTEREST

The following table captures the operators that are actively behind ODTN design, implementation and deployment.

Operator	**Interest**
NTT	We expect that ODTN will dramatically shorten the service development term and reduce costs
Telefonica	This SDN application is key to accomplish our goals regarding network programmability, focused on extending and personalizing user services, and pursuing full operational automation. An open implementation is paramount to demonstrate feasibility and stimulate interoperability.
Telecom Italia	decoupling and disaggregation of hardware and software layers in the transport layer is a very attractive strategy

	for achieving efficiency and automation for network operators.
China Unicom	By disaggregating optical transport networks, we expect to yield benefits such as the ability to build more cost effective networks while giving us greater ability to innovate rapidly

2 BACKGROUND

Today's optical transport market is served by vertically integrated network solutions. Operators select a vendor for their network, and then all transponders, ROADMs, line systems and optical design tools are provided by this same vendor. As a result, operator choice becomes limited once the initial vendor selection is made. This inhibits flexibility and innovation, and further expands vendor lock-in as the network is maintained and expanded over time.

Historically, DWDM optical communications was known to pose technological challenges due to the analog nature of the technology. This technical complexity has long been the basis of reasoning for requiring vertically integrated solutions, but ODTN is taking an approach designed to bridge this gap. To ease optical distance and compatibility issues, ODTN will assume that every optical link uses a matched pair of transponders from a single vendor. But unlike single vendor solutions, the network can use a different brand of transponder for each link, and these transponders can run over an open line system from yet another supplier. ODTN, through it's reference platform implementation, automatically and transparently discovers the disaggregated components and will control the entire transport network as a unified whole, thus enabling multi-vendor choice.

3 ODTN MOTIVATION AND BENEFITS

Through the disaggregation and opening of the optical systems and by using open APIs ODTN brings several benefits to the optical space.

A rapid cycle of innovations can happen in terminal equipment thus enabling vendors to bring new features and increment speed, reach, QoT and optimize use of spectrum e.g. through flex-grid. At the same time operators can reap benefits through simple bookending. New services can also be rapidly created, prototyped, tested and introduced in production networks through the easiness of software development and deployment with the use of open APIs.

Another benefit of an open and disaggregated system is the capability of building a CI/CD pipeline between operators and vendors through an open source software stack.

At the same time Open APIs mean that the project and it's deployment will be vendor neutral and modular.

ODTN will also enable the creation of a robust market of white-box optical 'peripherals' from many suppliers that can be intermixed to build complete solutions. Suppliers can narrowly focus on a specialty component (e.g. transponder) without having to build a complete solution themselves. Such approach drives both innovation and lower costs. Operators can also select best-of-breed components and avoid vendor lock-in, enabling innovation and flexibility as their network needs grow.

ODTN's modularity and flexibility also enables a better Life-Cycle Cost Approach (LCCA) since components can be interchanged at the end of their lifespan without the need to completely modify the network architecture given that a single component reached the end of it's lifespan.

4 ODTN USE CASES

This Section highlights some of the Service Providers use cases for ODTN explaining network architectures, provisioning and connectivity lifecycle.

4.1 NTTCOM EXEMPLAR USE CASE: DC / CLOUD INTERCONNECT

ODTN is expected to serve as the SDN controller which will enable us to provision Data center / Cloud interconnect transport networks. Service provider's enterprise customers are recently using multi-cloud environments including Amazon Web Services (AWS), Microsoft Azure and Google Cloud Platform (GCP), and also their own Data Centers (DC) to satisfy their application requirements. SPs requirements on their own network service have become more and more diversified. SPs are expected to provide various network services which include best effort low-cost or dedicated and high-speed connectivity, enforcing also other properties such as security and low latency.

Adopting a dis-aggregated optical network architecture enables Service Providers to select multi-vendor components, such as transponders, with different bandwidth, service level and additional value-added features. A disaggregated and flexible architecture enables the SP to provide the network services that each customer really wants using equipment that suits their requirements.

Enterprise customers will request a new network service through an End-to-end (DC/Cloud-to-DC/Cloud) service order. The service order will be stored in the operators' Operations Support System (OSS) or Business Support System (BSS). Transponders and OLS devices which are already installed in the DC/Cloud Interconnect network and its service interface points will also be stored in OSS/BSS as a devices and services inventory list. The devices and capabilities are expected to be discovered using the ODTN platform discovery feature, gathered from the End to end orchestrator (E2E) and offered to the OSS/BSS.

With Open API integration, the E2E orchestrator will be requested to instantiate a service by the OSS/BSS when the service order comes. The E2E orchestrator will reflect part of the request in turn, the part related to connectivity of the optical domain, automatically to the ODTN platform. In the provisioning function the E2E orchestrator will send a request to create optical connectivity service to the ODTN platform, which should complete this dis-aggregated transport provisioning task, configuring the transponders and OLS devices successfully in order to provide the demanded path, and return its result to the E2E orchestrator.

The E2E orchestrator will also be responsible for other tasks like monitoring, network state analysis and provisioning of any non ODTN managed domains.

A full network architecture is shown in Figure 1 in order to provide a frame of reference and the context in which ODTN will position itself.

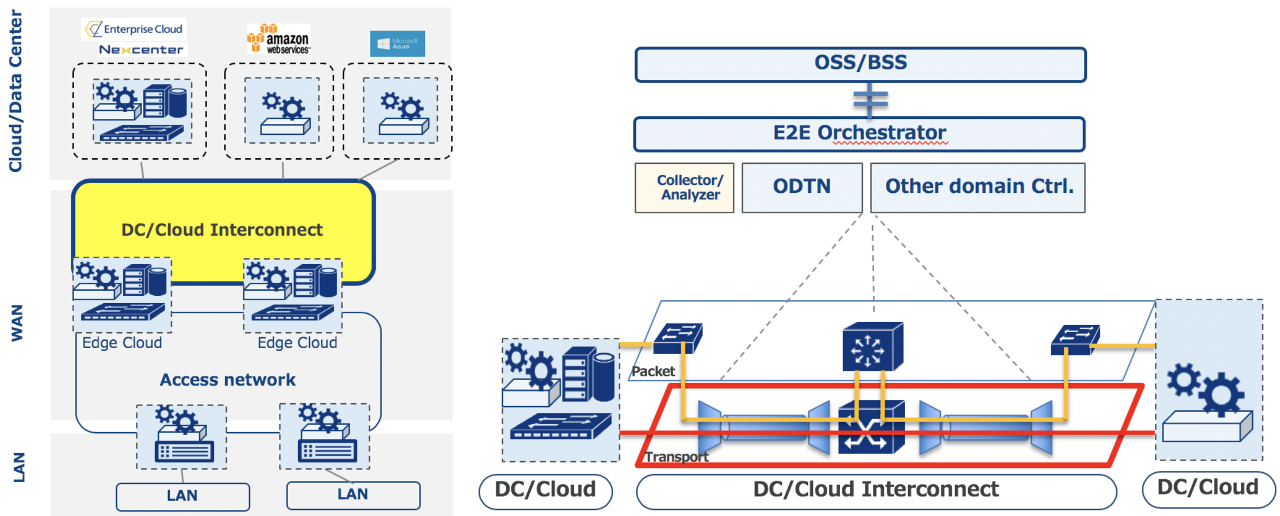


Figure 1

4.2 TELEFONICA USE CASE: PARTIALLY DISAGGREGATED TRANSPORT NEWORKS

Optical networks are the basis of the Telecommunications Operators (TELCOs) transport infrastructure in the different aggregation levels (metropolitan, regional and long-haul).The TELCOs deploy in the different aggregation levels the solution from a single vendor. The reasons are the complexity of optical networking, the low level interoperability and the improvement of maintenance activities.

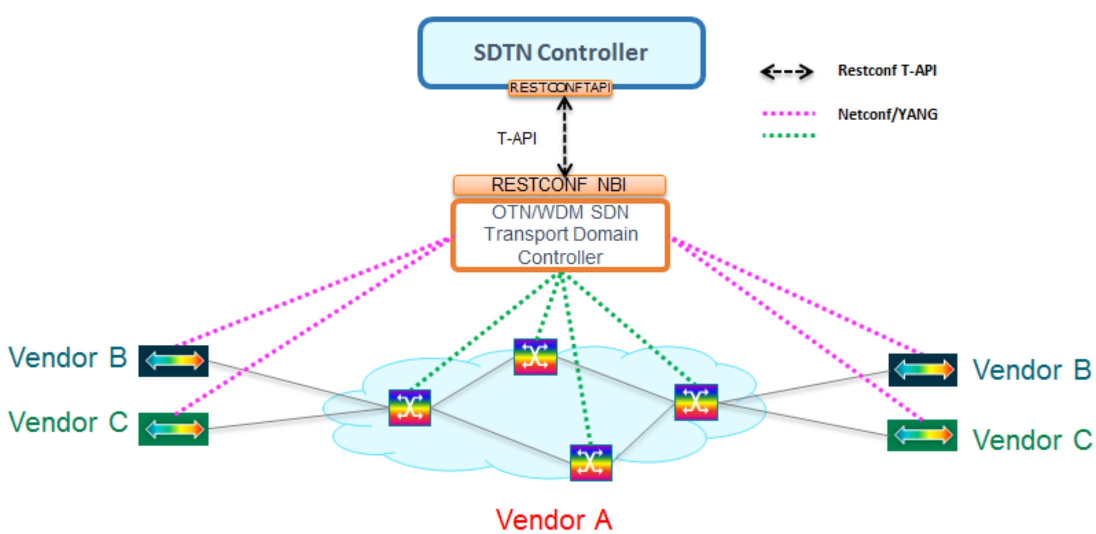
Optical Disaggregation aims to decouple the components of a line system (transponders, ROADMs, amplifiers, etc.), so the components can interoperate with solutions from other vendors. This initiative eliminates the lock-in situations and fosters competition and innovation in the transmission segment.

Hardware and software disaggregation are a recognized strategy for achieving efficiency and cost reduction within datacenter warehouse.

In order for optical disaggregation technology to be a reality in our operations, in real scenarios, the following activities must be developed:

- Standardization of the management interface and APIs of the terminal devices or Open Terminals (OTs) and the Open Line System (OLS).
- Definition of a
- For any OLS definition, it is inevitable to define a "termination point" (TP) in each participating platform to handover an optical signal between the structures/layers.
- Capability for a a line system to be able to transport any kind of signal within a given power level and channel characteristics.

Software Defined Networking (SDN) is a key technology enabler to make optical disaggregation feasible and is a key component of the partially disaggregated program within Telefonica. The interfaces towards the transponders should be able to configure and adjust in real time all optical parameters (modulation format, output power level, central frequency, among others). On the other hand, it should be possible to set-up network media channels, which consist of a "chunk" of frequency between two devices in the network able to carry optical signals.



Telefónica's SDN target architecture includes an optical WDM/OTN Domain controller which is responsible of the management and control of the optical transport domain. In disaggregated environment Telefonica expects that this SDN Domain controller will be able to manage the disaggregated OTs and OLS components and export vendor agnostic network management capabilities through its Northbound Interface towards OSS/BSS clients or towards a hierarchical Software Transport Network Controller (SDTN) for end-to-end integration with other technologies and layers. ODTN will play the role of the WDM/OTN Domain controller.

5 ODTN REFERENCE DESIGN TARGET

ODTN plans to incrementally address complex network scenarios, starting with relatively simple point-to-point open line systems and ending with a meshed network consisting of disaggregated optical equipment. The most complex scenario would be a fully disaggregated architecture, where the flex-grid technology with ROADM devices is deployed.

ODTN consumes and relies heavily on standard models (particularly OpenConfig and TAPI), but does not develop the standards themselves.

The project scope is to build an Open source network operating system for control and configuration of the Disaggregated DWDM system through Open and common data models, APIs and protocols. Devices and components will be including, but not limited to, transponders and Open Line Systems (OLS), which includes amplifiers, multiplexers, all-optical switches and ROADMs.

The ODTN reference design outlines a two phase approach to achieve its use cases. The first phase describes and designs a point to point solution, while the second phase builds on top of that to achieve a fully meshed ROADM solution.

5.1 POINT TO POINT SOLUTION (PHASE 1)

The ODTN project, during phase 1, does not require a completely new network infrastructure. It aims at leveraging existing deployments while augmenting them with programmability and new features through adding an SDN controller and open APIs enabled devices (e.g Transponders).

The first phase of the ODTN project aims at establishing connectivity between two pairs of transponders from the same vendor, thus avoiding incompatibility issues such as the ones between different and proprietary FEC algorithms. Connectivity establishment is achieved not only by controlling with open APIs and models the transponders, but also the OLS in the middle.

The OLS is used to execute "Path computation" and "Power Control" of the devices underneath the OLS domain, e.g. Muxponders, WSS and Amplifiers.

The Northbound service definition is done through ONF Transport API (T-API) through it's connectivity model.

ONF T-API is also used to program the OLS through the connectivity model, if needed augmented with topology and power control information.

Transponders configuration is done according to OpenConfig standard models.

Communications protocol used may vary but the design team suggests going for open and widely adopted ones such as NETCONF or gRPC.

To achieve phase 1 different devices need to support different operations.

The transponder devices need to expose API to:

- allow discovery of device and port information
- enable a cross-connect between any line side and any client side port with a given wavelength and optical channel

The OLS needs to expose API to:

- allow discovery of device and port information
- provision a path between two transponder facing ports (INNI). If the OLS exposes itself as a single component, a "big-switch", it needs the capability to compute internally a path between the outwards-facing ports with the given optical parameters. If the OLS exposes a full mesh topology the ODTN reference3 platform has to be capable of doing path computation given the optical constraints.

5.2 FULLY DISAGGREGATED ROADM WITH OPEN APIS (PHASE 2)

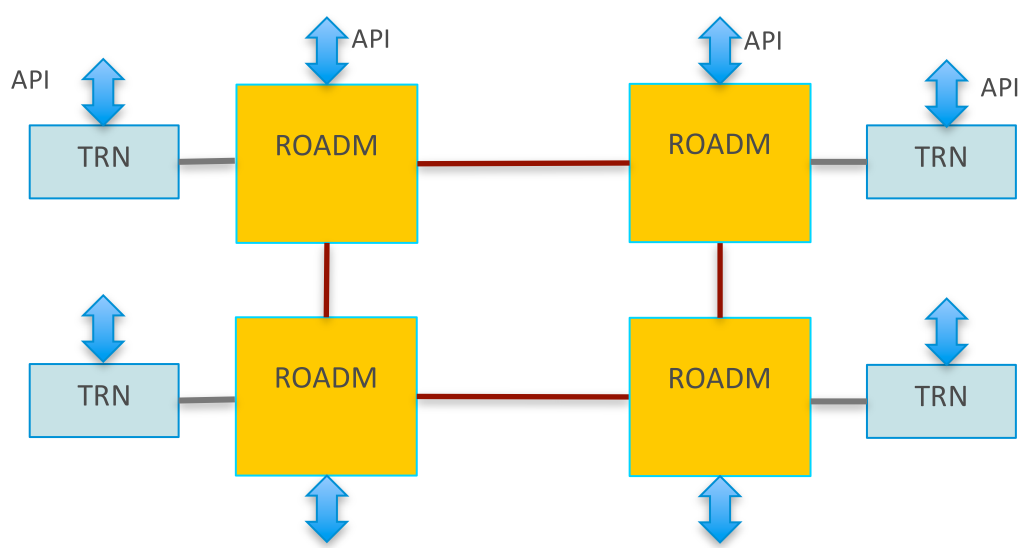
The ODTN project, during Phase 1 aims to have all the optical devices along the path between the two endpoints be disaggregated and open, controlled through open APIs and standard open source models. Phase 2 will remove the intermediate OLS controller and enable the ODTN platform to control fully disaggregated ROADM devices.

The APIs used to to define the interaction, the capability and to control the ROADM devices in Phase 2 are yet to be defined for the scope of ODTN. Two possibilities currently are prominent:

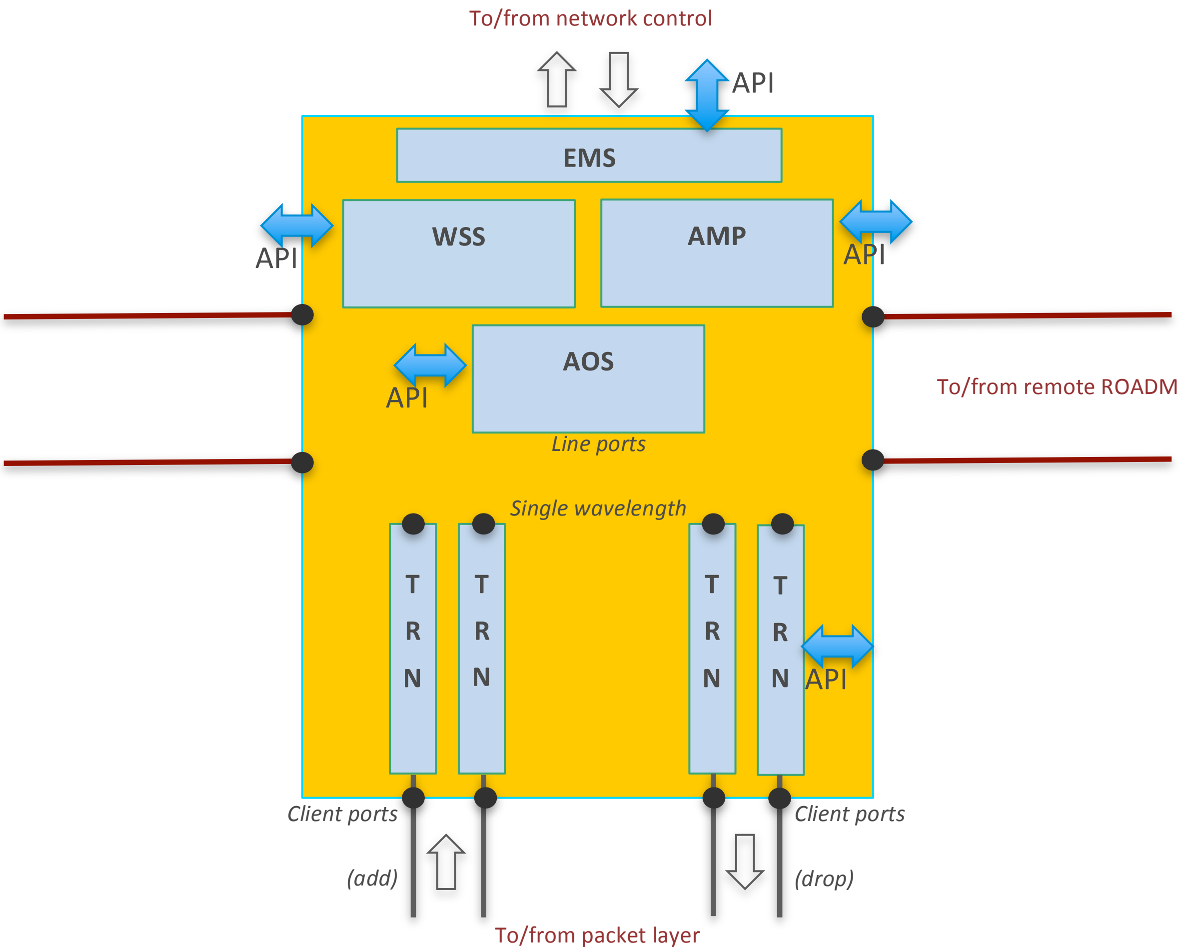
- OpenROADM Standard and Yang models.
- TAPI 2.1

The decision on which API and models to be used during this phase will be taken at a later time after more technical discussion and studies providing detailed information based on which a proper and long lasting decision can be made.

The disaggregation during this phase enables to control a ROADM as whole, as shown in the next figure:



Another option is to control each and every component of the ROADM architecture separately depending on the deployment need and device capabilities, as shown in the next figure:



5.3 SERVICE REQUEST APIS

5.3.1 Northbound

ODTN platform will use TAPI 2.1 to receive network wide edicts and point to point connectivity requests.

The Tapi 2.1 release is here:

https://github.com/OpenNetworkingFoundation/TAPI/releases/tag/v2.1.0

The TAPI connectivity model is

https://github.com/OpenNetworkingFoundation/TAPI/blob/master/YANG/tapi-connectivity%402018-10-16.yang

Also the TAPI common model will be used

https://github.com/OpenNetworkingFoundation/TAPI/blob/master/YANG/tapi-common%402018-10-16.yang

Tapi Photonic Media

https://github.com/OpenNetworkingFoundation/TAPI/blob/master/YANG/tapi-photonic-media%402018-10-16.yang

The following models are needed to save internally the state of the ODTN platform topology and properly expose it through restconf on the NB.

Tapi Topology Model

https://github.com/OpenNetworkingFoundation/TAPI/blob/master/YANG/tapi-topology%402018-10-16.yang

A first list for the endpoints is:

- tapi-common
    - get-service-interface-point-details
    - get-service-interface-point-list *
- tapi-connectivity-service
    - create-connectivity-service
    - delete-connectivity-service
    - get-connectivity-service-details
    - get-connectivity-service-list
    -

The T-API defined connectivity request issued to the ODTN platform will include the parameters to configure the optical channel, as defined in the photonic media channel model. This parameter will be used by the ODTN platform to compute a valid end to end connectivity at both the photonic layer and the User Connectivity Layer (DSR).

This list is subject to extension and augmentation in future versions of the document.

ODTN path provisioning will be triggered by two northbound TAPI requests.

- Layer 0 photonic request. This request handles the provisioning of a Layer 0 photonic path between the line side ports of the transponders, across the OLS or the ROADM devices. This request will contain the photonic parameters needed both by the Transponder to configure the line-side port and from the OLS or ROADM devices to create an end to end path across the underlying equipment.

- Layer 2, data link connectivity or DSR request. This request handles the provisioning of the cross connection between the client side port of the Transponder and the previously provisioned line side port.

Requests should come in the proper ordering of L0 photonic request and then L2 request in order for the path to be properly provisioned by the ODTN platform.

5.3.2 Southbound

This section defines the APIs and models ODTN expects to find from devices on the Southbound. The platform can nonetheless perform actions to devices which are not compliant to this API and supports model deviations thanks to the modular architecture described in section 6.2.

5.3.2.1 Transponder APIs

ODTN platform will use OpenConfig models to issue service activation in the Transponders, e.g. to connect client side port to line side port.

Openconfig Models and Openconfig calls

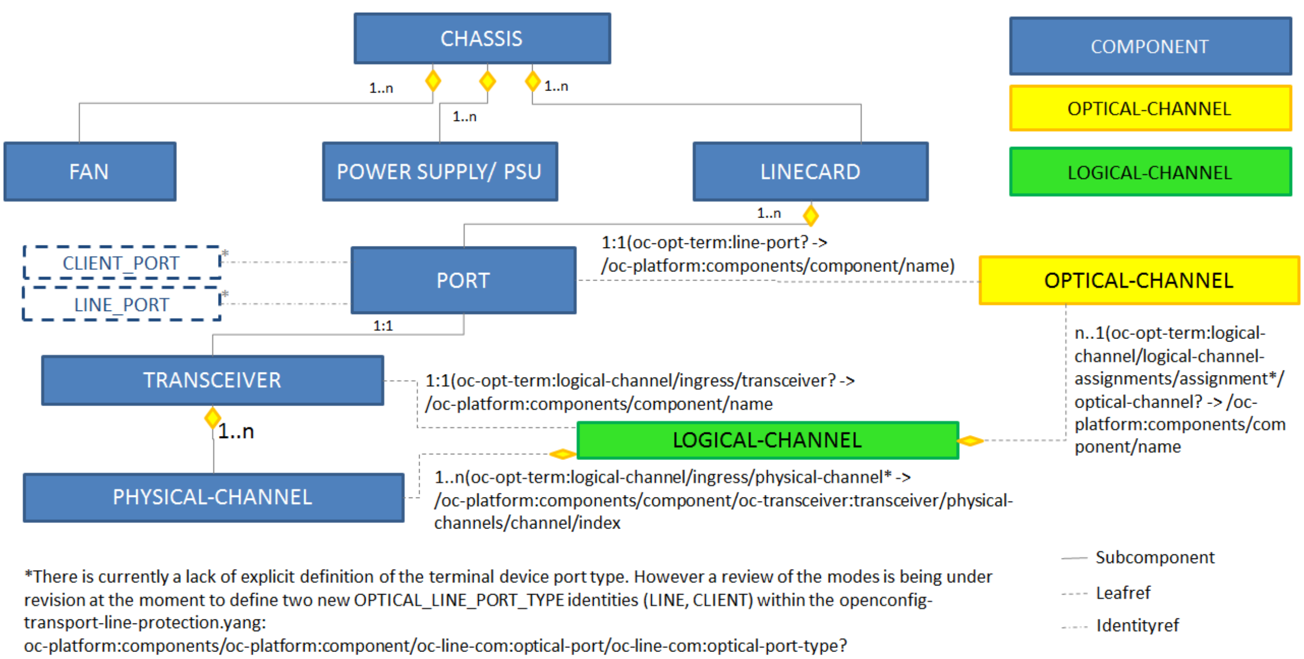
OpenConfig Models	Openconfig Version	Latest Revision
openconfig-platform-transceiver.yang	0.5.0	15/05/2018
openconfig-platform.yang	0.11.0	16/03/2018
openconfig-platform-linecard.yang	0.1.0	03/08/2017
openconfig-platform-port.yang	0.3.0	20/01/2018
openconfig-platform-types.yang	0.9.0	05/05/2018
openconfig-terminal-device.yang	1.0.0	08/07/2017
openconfig-transport-types.yang	0.6.0	16/05/2018
openconfig-types.yang	0.5.0	05/05/2018
openconfig-interfaces.yang	2.3.1	24/04/2018
openconfig-transport-line-common.yang	0.3.1	08/09/2017

openconfig-transport-line-protection.yang	0.3.1	08/09/2017

Terminal Inventory device discovery

The first step towards configuration of the terminal devices is the discovery of its physical and logical configuration. In this line, ODTN propose a reference design layout for both the physical and logical device structure based on openconfig models.

Following, it is presented the reference design of terminal devices physical layout based on current openconfig model definitions. This block diagram includes the relationship between the openconfig models entities designed to represent the internal physical structure of a terminal device (e.g., the composition of a chassis based on linecards and ports components), and also the relationship with the openconfig representation optical channel and logical entities (which represents the mapping between incoming client signals and egress optical signals towards the Optical Line System).



For the retrieval of the device inventory data (configuration and state), the openconfig path is:

Physical inventory (openconfig-platform.yang):

/oc-platform:components/

Logical inventory (openconfig-terminal-device.yang):

/oc-opt-term:terminal-device/

The list of possible tributary signal representations in openconfig as Logical-Channels is:

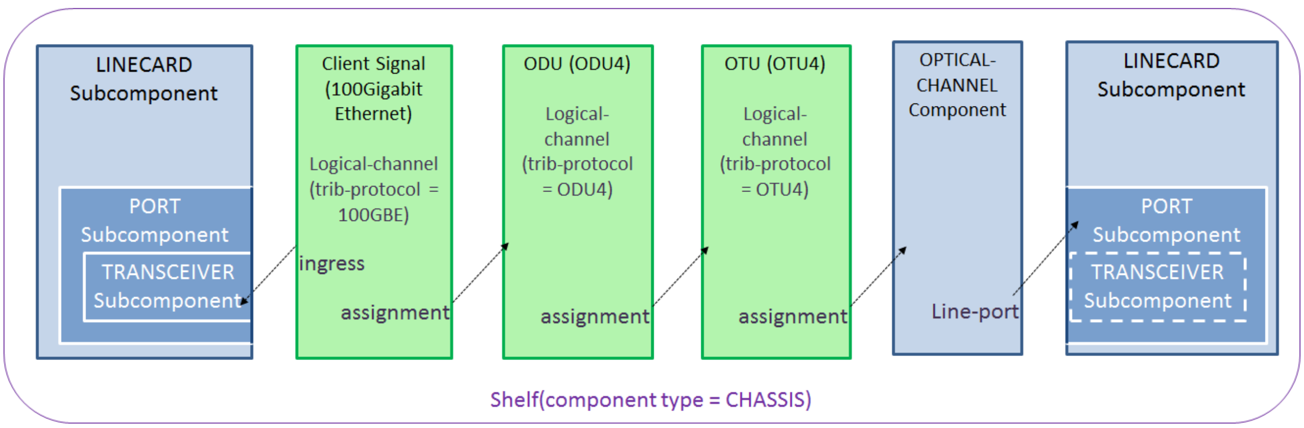
```
PROT_1GE
PROT_OC48
PROT_STM16
PROT_10GE_LAN
PROT_10GE_WAN
PROT_OC192
PROT_STM64
PROT_OTU2
PROT_OTU2E

PROT_OTU1E
PROT_ODU2
PROT_ODU2E
PROT_40GE
PROT_OC768
PROT_STM256
PROT_OTU3
PROT_ODU3
PROT_100GE
PROT_100G_MLG
PROT_OTU4
PROT_OTUCN
```

```
PROT_ODUCN
PROT_ODU4
```

Terminal device configuration

In the following diagram, the reference design of the logical layout for service configuration (e.g.,100GE) is shown. The diagram shows how a client service is mapped into an optical channel across the different layers (ODU, OTU) and how these mappings are represented in OpenConfig as relationships between different layer's logical channels.



Optical channel (OCh/OTSi) configuration.

The optical channel, representing the OCh/OTSi signal egressing the terminal device from a line port, allows to modify the following parameters:

- Central frequency of the optical channel (OCh).
- Configuration of the target output power of the optical channel.
- Configuration of the optical channel operational mode i.e, the channel characterization (modulation format, FEC, baud-rate).
- Attach the optical channel to an available line port.

The openconfig path for this operation is:

/oc-platform:components/oc-platform:component/oc-opt-term:optical-channel/

The yang tree of this configuration is:

module: openconfig-platform
+--rw components
+--rw component [name]*

*+--rw openconfig-terminal-device:optical-channel!**

*+--rw openconfig-terminal-device:config!**

| +--rw openconfig-terminal-device:frequency? oc-opt-types:frequency-type

| +--rw openconfig-terminal-device:target-output-power? decimal64

| +--rw openconfig-terminal-device:operational-mode? uint16

| +--rw openconfig-terminal-device:line-port? -> /oc-

platform:components/component/name

Operational-mode in OpenConfig is designated for vendors to define optical-channel

modulation format, symbol rate, FEC mode and other transmission parameters. In current openconfig definition operational-mode is only represented as an integer ID value, an extensive modelling of optical-transmission formats is under discussion. Thus, ODTN from now recommends that any terminal-device openconfig implementation shall be accompanied by a list of the supported operational models characterized by (and not limited to) the following list of parameters:

- Baud rate
- Modulation Format
- FEC
- Grid Type
- Frequency range (max/min central frequency configurable)
- TX power range (max/min channel output power configurable)
- Effective media channel width (including guard band)

Additionally it would be also desired to have a description of the receiver side tolerance parameters:

- Chromatic dispersion tolerance (ps/nm)
- Differential Group Delay (DGD) tolerance (ps)
- Input Power Sensitivity (min/max dBm)
- Minimum OSNR (dB) calculated with 0.5 nm RBW
- Pre-FEC BER
- Post-FEC BER

a. Client services' (ETH/OTN) - Line Side (OCh/OTSi) cross connection

As it is illustrated in the diagram at the head of this section, the configuration of the client services and its mapping to the line side optical channel is made on a per-layer basis following the ITU-T G.709 specification of optical transport network interfaces.

The range of possible configurations is wide and depends on the capabilities of the terminal device and the use case intended to be configured. In this document the intention is to provide a global guideline for a general case from which any possible specific case could be derived. In this line, the following common guidelines are proposed:

- Line side HW configuration (i.e., Optical Channel configuration and association to physical line side port component) needs to be made in advance before any service Logical Channel (OTUk, OTUCn) assignment references it. Pre-provisioning of the Optical Channel in the transponder should be considered by the ODTN platform if the HW comprises of pluggable modules that are installed at later times.

- Tributary logical channels (from incoming Client side Logical Channel to Line side Optical Channel) need to be created in sequential order from Line Side to Client Side (i.e., an OTUk logical channel needs to be created before the ODUk logical, if so, its created).The openconfig path for this operation is:

/oc-opt-term:terminal-device/oc-opt-term:logical-channels/oc-opt-term:channel/

The yang tree of this configuration is:

```
module: openconfig-terminal-device
   +--rw terminal-device
      +--rw logical-channels
      |  +--rw channel* [index]
      |     +--rw index                      -> ../config/index
      |     +--rw config
      |     |  +--rw index?                 uint32
      |     |  +--rw description?           string
      |     |  +--rw admin-state?           oc-opt-types:admin-state-type
      |     |  +--rw rate-class?            identityref
      |     |  +--rw trib-protocol?         identityref
      |     |  +--rw logical-channel-type?  identityref
      |     |  +--rw loopback-mode?         oc-opt-types:loopback-mode-type
      |     |  +--rw test-signal?           boolean
```

- A Client side Logical Channel, representing the client port incoming signal, need to be created before the corresponding port/transceiver component can be assigned to it by configuring the client side Logical Channel *ingress* parameter. The openconfig path for this operation is:

/oc-opt-term:terminal-device/oc-opt-term:logical-channels/oc-opt-term:channel/

The yang tree of this configuration is:

```
module: openconfig-terminal-device
   +--rw terminal-device
      +--rw logical-channels
      |  +--rw channel* [index]

      |        +--rw ingress
      |     |  +--rw config
      |     |  |  +--rw transceiver?       ->
```

```
/oc-platform:components/component/name

      |     | |  +--rw physical-channel*   -> /oc-platform:components/component/oc-transceiver:transceiver/physical-channels/channel/index
```

- Logical channel can be referenced only if it has been already assigned with the HW component (Optical-Component). I.e., in the example proposed, before the OTU4 Logical Channel can be referenced by the ODU4 Logical Channel, the assignment from OTU4 Logical Channel to the Optical Channel needs to be configured.The openconfig path for this operation is:

```
/oc-opt-term:terminal-device/oc-opt-term:logical-channels/oc-opt-term:channel/
```

The yang tree of this configuration is:

```
module: openconfig-terminal-device
   +--rw terminal-device
      +--rw config
      +--ro state
      +--rw logical-channels
      |  +--rw channel* [index]

      |     +--rw logical-channel-assignments
            |        +--rw assignment* [index]
            |           +--rw index      -> ../config/index
            |           +--rw config
            |           |  +--rw index?             uint32
            |           |  +--rw description?       string
            |           |  +--rw assignment-type?   enumeration
            |           |  +--rw logical-channel?   -> /terminal-device/logical-channels/channel/index
            |           |  +--rw optical-channel?   -> /oc-platform:components/component/name
            |           |  +--rw allocation?        decimal64
```

- Logical channel can be deleted only if it is not used as assignment.

5.3.2.2 Optical Line System APIs

ODTN platform will use the connectivity model of TAPI 2.1 to issue point to point connectivity requests to the Optical Line System.

The Tapi 2.1 release is here:

https://github.com/OpenNetworkingFoundation/TAPI/releases/tag/v2.1.0

The TAPI connectivity model is

https://github.com/OpenNetworkingFoundation/TAPI/blob/master/YANG/tapi-connectivity%402018-10-16.yang

Also the TAPI common model will be used

https://github.com/OpenNetworkingFoundation/TAPI/blob/master/YANG/tapi-common%402018-10-16.yang

Tapi Photonic Media

https://github.com/OpenNetworkingFoundation/TAPI/blob/master/YANG/tapi-photonic-media%402018-10-16.yang

The following models are needed to save internally the state of the ODTN platform topology and properly expose it through restconf on the NB.

Tapi Topology Model

https://github.com/OpenNetworkingFoundation/TAPI/blob/master/YANG/tapi-topology%402018-10-16.yang

TAPI Models and TAPI Calls for OLS

A first list for the endpoints is:

- tapi-common
    - get-service-interface-point-details
    - get-service-interface-point-list *
- tapi-connectivity-service
    - create-connectivity-service
    - delete-connectivity-service

- get-connectivity-service-details
- get-connectivity-service-list

The T-API defined connectivity request issued from the ODTN platform to the OLS system will include the parameters to configure the optical channel, as defined in the photonic media model. This parameter is to be used by the OLS controller to deploy an end to end connectivity at the photonic layer.

5.3.2.3 ROADM APIs

ODTN platform may use OpenRoadm models to control the ROADMs for phase 2. The version of the model and more details will be decided in a subsequent version of the document when a survey of existing implementations will be done.

6 ODTN DESIGN PRINCIPLES

The developed platform for ODTN needs to enable easy to develop apps, have a modular and composable architecture that enables reuse of existing pieces and easy addition of device drivers and multiple southbound interfaces. Model defined services and device configuration are also key for ODTN.

6.1 EASY TO DEVELOP APPS

Service providers operate multiple domains. Each domain has unique specifications, for example, metro domain has ring topologies, core domain has ladder topologies. They need to develop controller applications for each network domain so controller should provide a framework that enables easy and agile application development.

6.2 MODULAR PLATFORM

ODTN platform is required to be modular in order to achieve optimization of components, be able to benefit from other projects' platform enhancements and be easily extensible with new components, applications and device support. A clear example comes from the need to easily add device drivers.

Especially in the disaggregation domain, service providers manage different type of devices in their SDN controllers. Since the components integrated in all-in-one devices are disaggregated devices, the number of devices which SDN controllers have to manage increases further. Additionally, service providers use several models for each component. This practice increases the number of device models to manage.

To take advantage of the disaggregation, it should be easy to introduce new devices which support the chosen open APIs with minor vendor-specific deviations.

OpenConfig and Open ROADM are informal working groups adopting declarative model-based configuration and operation.

6.3 SCALABILITY AND FAULT TOLERANCE

The ODTN platform needs to be scalable to support different network loads and deployment scenarios. The platform needs to be scalable in terms of:

- number of devices
- number of operations requests from OSS/BSS
- number of southbound and device facing operations
- size of configuration trees

The ODTN platform also needs to be capable of tolerating network failures and platform failures, while keeping the requested services available.

6.4 CONFIGURATION

Service providers have to add a lot of device drivers to SDN controller to manage variety of devices from multiple vendors. Therefore, it is important for SDN controller to provide the common functions as built-in and save time to implement such common functions in each domain and each device. For example, transactional management, rollback and rehearsal functions as described in this document, should be provided by the ODTN platform.

In particular ODTN's platform should support:

6.4.1 Multi-device transactional update

It is necessary to set up configurations into multiple devices in order to provision a transport network service. If a failure occurs during a provisioning operation, all of the configurations that were successfully set must be restored to their original state.

6.4.2 Detect and resolve unmatched configurations between controller and devices

This is necessary to confirm that the configuration the controller has matches the configuration the device has, manually or periodically by operators, or at the timing of changing the configuration of devices. if inconsistency is present take action to solve it.

6.4.3 Configuration rollback

During the process of configuring a given set of resources failure might happen in a number of the elements thus every operation, even the successful one need to be rolled back. Rollback needs to be implemented both for a new and added resources and for modified ones.

6.4.4 Plug & play

Support needs to be present for configuration to be pre-loaded in the ODTN platform and applied to a device when it connects to the system.

6.4.5 Multi-version models support

Different versions of the same device and service models need to be supported in order to cope with different deployments and infrastructure.

6.4.6 Synchronize from device

Initial configuration from the device should be loaded into the configuration store in the ODTN platform.

6.4.7 Dry Run and Configuration rehearsal

During delete and modify operations of existing services the system would need to know in advance what would be changed. Knowing in advance would enable the system to make sure no other service is affected by the modification or deletion of the first configuration.

6.5 MONITORING AND ALARMS

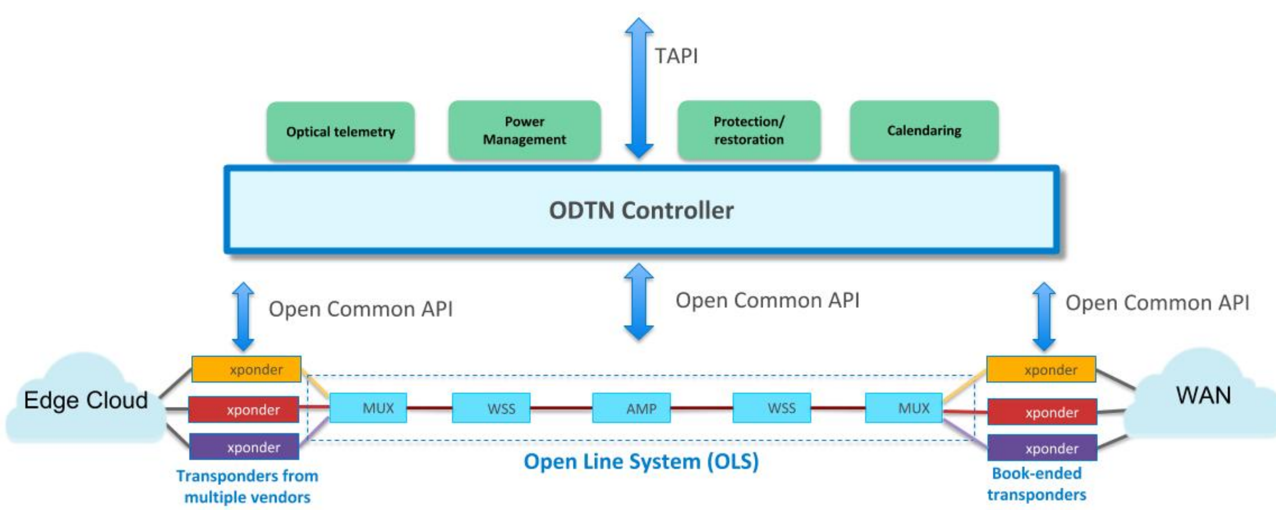
The ODTN platform must also achieve monitoring and network state analysis through alarms and notifications to identify, react to and resolve network issues.

7 ODTN PLACEMENT IN EXISTING NETWORKS

The ODTN project has to implemented in a way so that it can live and co-existing in deployed optical network where a BSS/OSS controls different domains through different components. One of such optical domains can be managed through the ODTN platform while others can be controlled through an existing EMS/NMS.

ODTN's compliance and interoperability with existing EMS/NMS systems will enable operators to incrementally install disaggregated components.

An example can be seen in the following figure.



8 ODTN RELATIONSHIP TO OTHER PROJECTS

ODTN is the only Open source solution in the optical transport space, but is leveraging other ongoing work which has focused on standardizing various interfaces and components.

ODTN will leverage and expose TAPI as its northbound interface, leveraging the work coming out of the ONF's Open Transport Configuration and Control (OTCC) project. Likewise, OpenConfig is the base southbound model and API for communicating to optical equipment.

TIP's Open Optical & Packet Transport project is producing open DWDM architectures, models and APIs, covering transponders, open line systems, and routers. In time, the ODTN project will benefit from the availability of open optical hardware coming from the TIP work. And visa versa, the TIP project will leverage the open source work coming out of ODTN on TIP white box hardware building blocks (such as Voyager or Cassini).

The Open ROADM MSA defines interoperability specifications and data models for optical devices, networks and services. ODTN benefits from this effort and, over time, it helps the industry achieve transponder compatibility. This will eliminate the need to deploy transponders in matched pairs, further disaggregating the solution and enabling even greater deployment flexibility.

ODTN will work with other standard bodies and open source projects to upstream addition and changes to standard models, code changes and works that is done.

End of

ODTN

Informational Reference Design

Write to rdspec@opennetworking.org with comments or questions.

Open Networking Foundation