# *Introducing network programmability into OTN/WDM networks*

# *Transport PCE*

## *a full open source approach based on OpenDaylight & Open ROADM*

**22nd of May 2019**

**Olivier Renais, Orange Labs**
olivier.renais@orange.com
**Shweta Vachhani, Dhruv Bhardwaj, AT&T**



orange™

# AGENDA

- Introduction
- Global architecture
- APIs
- Main modules description
- Current status
- Planned developments

# Transport PCE initial Goals

- **Transport PCE project was created in May 2016 with the aim of providing a Controller for Optical infrastructure based on open standards**

  **TransportPCE initial goals were multiple:**

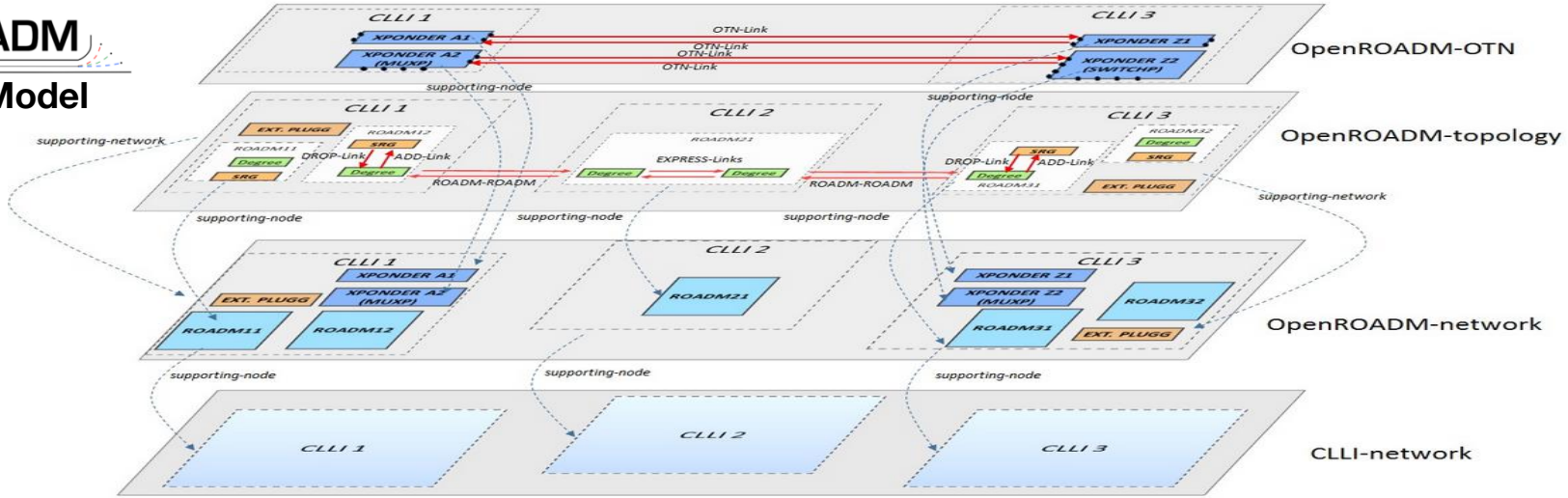
  - **Provide the community with:**
    - **an open implementation of main optical node management functions**
    - **feedbacks and a proof-of-concept of Open ROADM specifications**
  - **Standardize with yang models the API between the various components in an optical controller :**
    - **Northbound API based on Open ROADM service model**
    - **+ East/west APIs based on transportPCE models to help building a modular architecture and ease the integration of additional modules**
  - **Propose code and tests for a reference implementation that can be reused in third-parties derived products**

# The Open ROADM choice

- **Open ROADM Multi-Service Agreement (MSA) provides comprehensive and coherent Yang models, not only for device control, but also for network topology & service management.**
- **It also defines specifications for the optical layer.**
- **The disaggregation of ROADM & Xponders building blocks, provides a high level of interoperability**

Open ROADM

**Network Model**

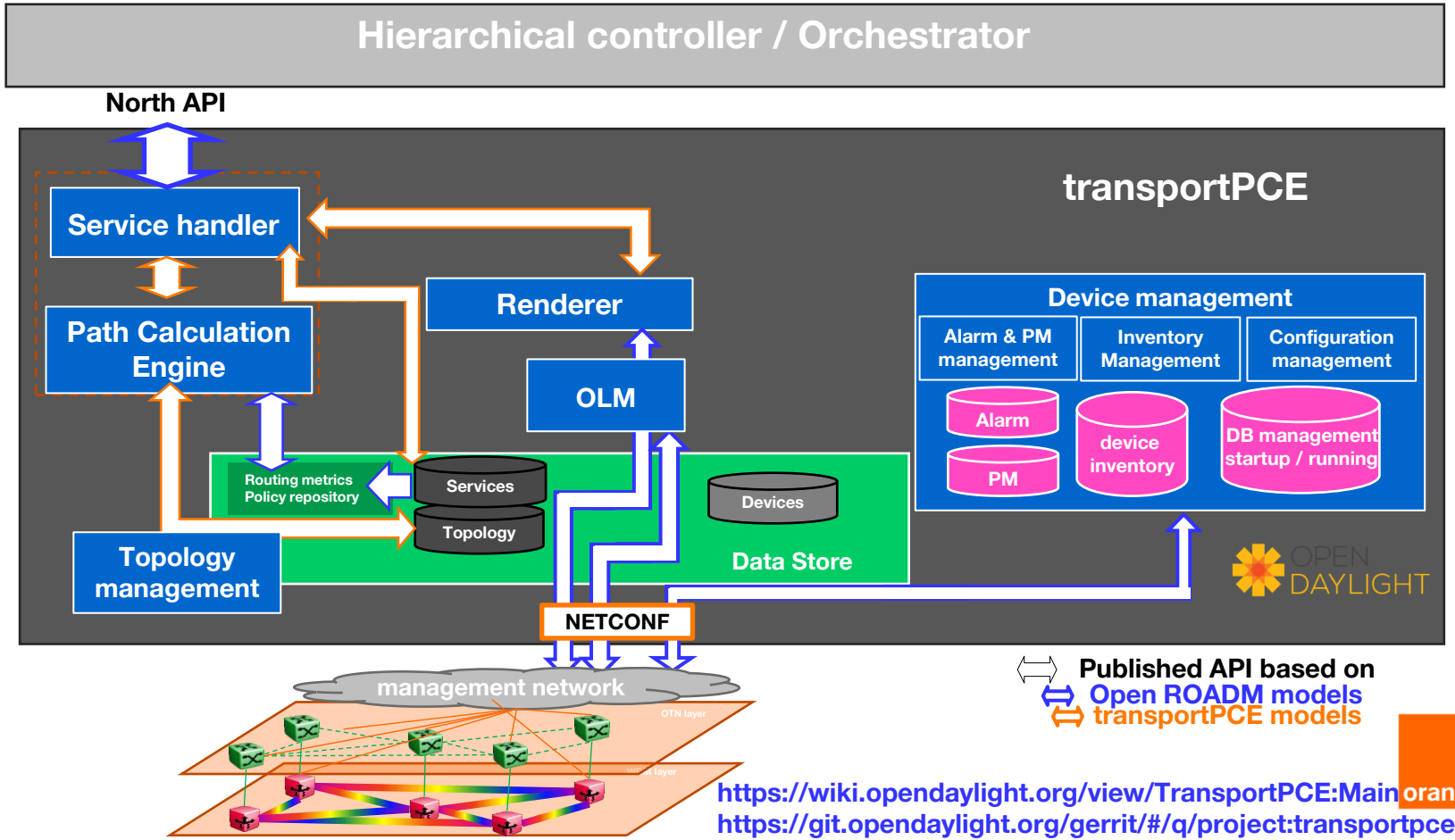



- **Interoperability is a key enabler for automation : Open ROADM turn out to be the most natural choice for tpce implementation**



orange™

# TransportPCE architecture

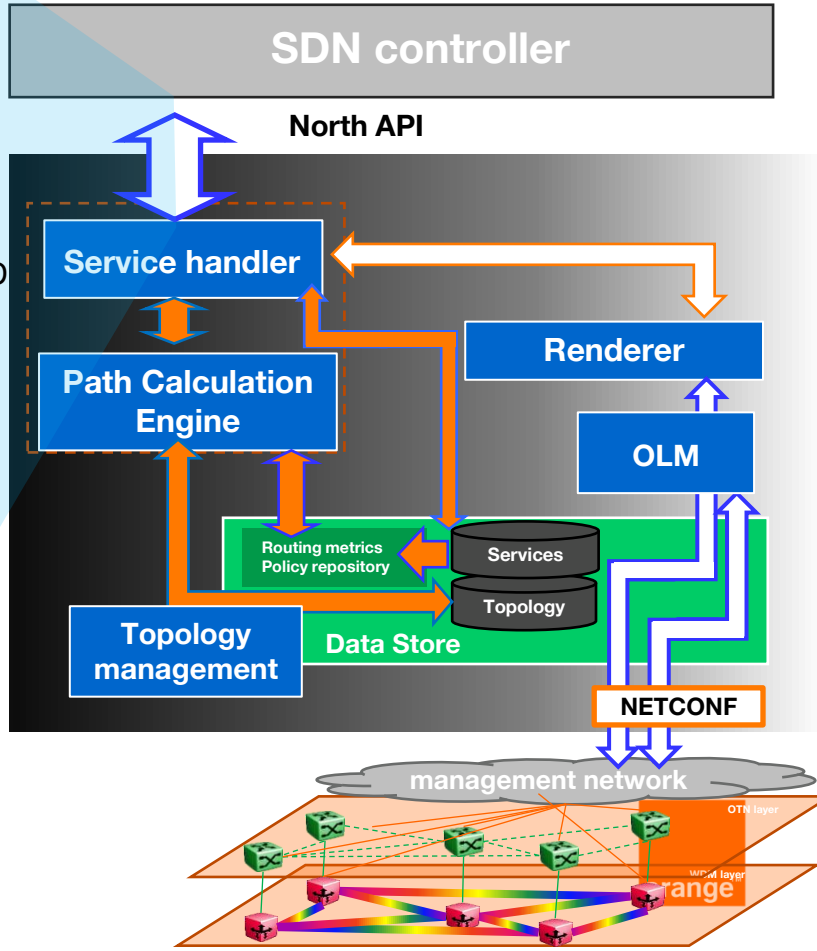


# APIs

- **Transport PCE North Bound Interface is based on the API defined in Open ROADM service model**
  - **Service interface is based on RESTCONF (RFC8040) as stated in the OR Service White paper**
    - RESTCONF protocol allows interrogating elements on their state using procedures based on http primitives
    - It uses POST, GET, PUT, DELETE… HTTP methods to provide CRUD (create, read, update, delete) operations on a conceptual Data Store containing YANG-defined data

| | | |
|---|---|---|
| DELETE | /openroadm/openroadm-services/{service-name} | Delete Service |
| GET | /openroadm/temp-services | Retrieve Temp Service |
| POST | /openroadm/temp-services | Create Temp Service |

  - **Open ROADM service model defines Remote Procedure Calls (RPCs).**
    - service create/delete, service-restoration, service-reversion
    - service-roll, service-reconfigure, service-reroute,
    - service-feasibility-check (-bulk)
  - **…. and notifications**
- **transport PCE also defines its own APIs**
  - Service-path (1.6) model



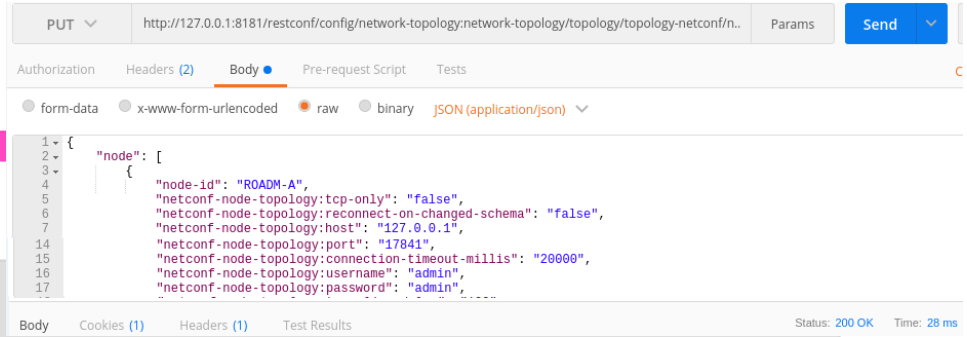
# transportPCE main modules

- **Service Handler :** handles request from Northbound API (service create, delete…), interrogates PCE for path calculation and calls Renderer and OLM for path implementation
- **Path Calculation Engine :** calculates path from A to Z according to existing topology
- **Topology management :** build and manage topology
  - Node discovery based on NETCONF Open ROADM equipment connection/disconnections
  - Link discovery based on lldp
- **Renderer :** configures connections and interfaces in the different nodes
- **OLM :** sets power levels in the different nodes
- **Device Management :**
  - *Alarms and Performance Monitoring management : store PMs and alarms in a specific Database (to be developed)*
  - Device inventory : stores nodes main information (Circuit packs, ports…) in a specific Database
  - *Configuration Management : manages configurations for the different nodes (to be developed)*



orange™

# Topology management







- 1 → Netconf details are provided to the network controller to reach a network element
  - 2→ as soon a NE is registered in the Netconf topology tPCE has all information to reach the node
    - 3 → A Netconf session is established and the node announces its capabilities
      - 4 → Node is added to the Open ROADM (I2RS compliant) topology

- As soon as Netconf nodes are in the Netconf topology of the controller, node discovery and Open ROADM topology building is based on equipment connection/disconnections
- Link discovery is based on lldp
  - Each nodes knows its lldp neighbors

orange™

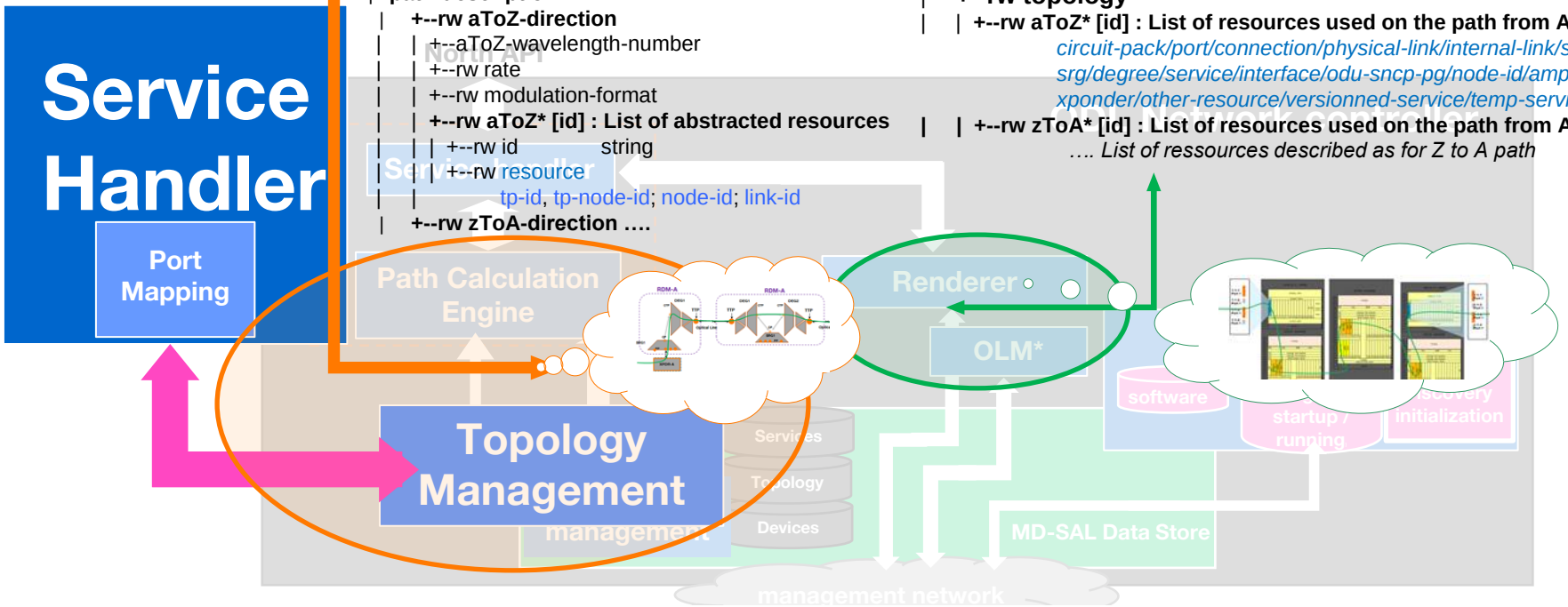
# Service Handler

**Service-path (transportPCE)**

```
| path-description
|   +--rw aToZ-direction
|   |  +--aToZ-wavelength-number
|   |  +--rw rate
|   |  +--rw modulation-format
|   |  +--rw aToZ* [id] : List of abstracted resources
|   |  |  +--rw id                string
|   |  |  +--rw resource
|   |  |        tp-id, tp-node-id; node-id; link-id
|   +--rw zToA-direction ….
```

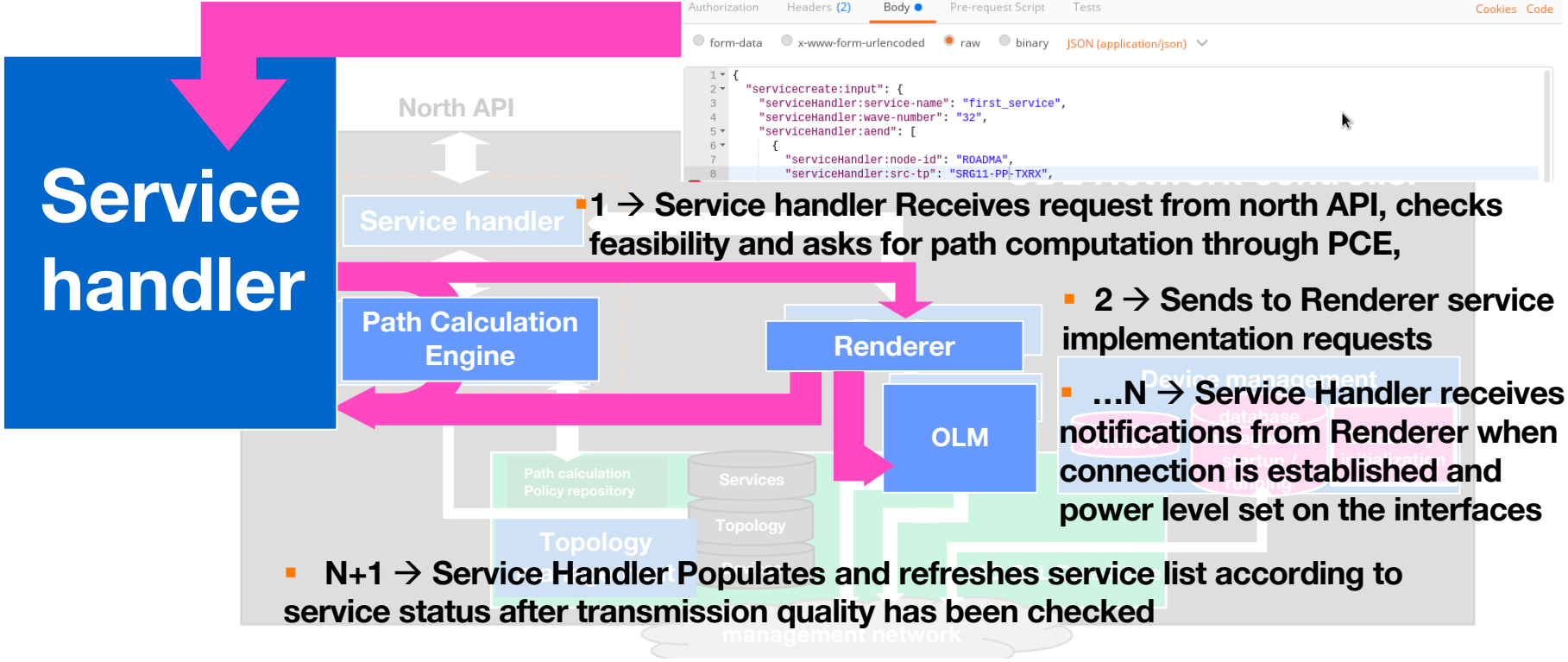
**Service (Open ROADM)**

```
|   +--rw topology
|   |  +--rw aToZ* [id] : List of resources used on the path from A to Z
          circuit-pack/port/connection/physical-link/internal-link/shelf/
          srg/degree/service/interface/odu-sncp-pg/node-id/amplifier/
          xponder/other-resource/versionned-service/temp-service
|   |  +--rw zToA* [id] : List of resources used on the path from A to Z
          …. List of ressources described as for Z to A path
```




- **Port mapping module of the Service Handler builds the mapping between physical elements as described in the Open ROADM service-model and abstracted elements used in the topology and in the path description of transport PCE service-path model**

# Service Handler



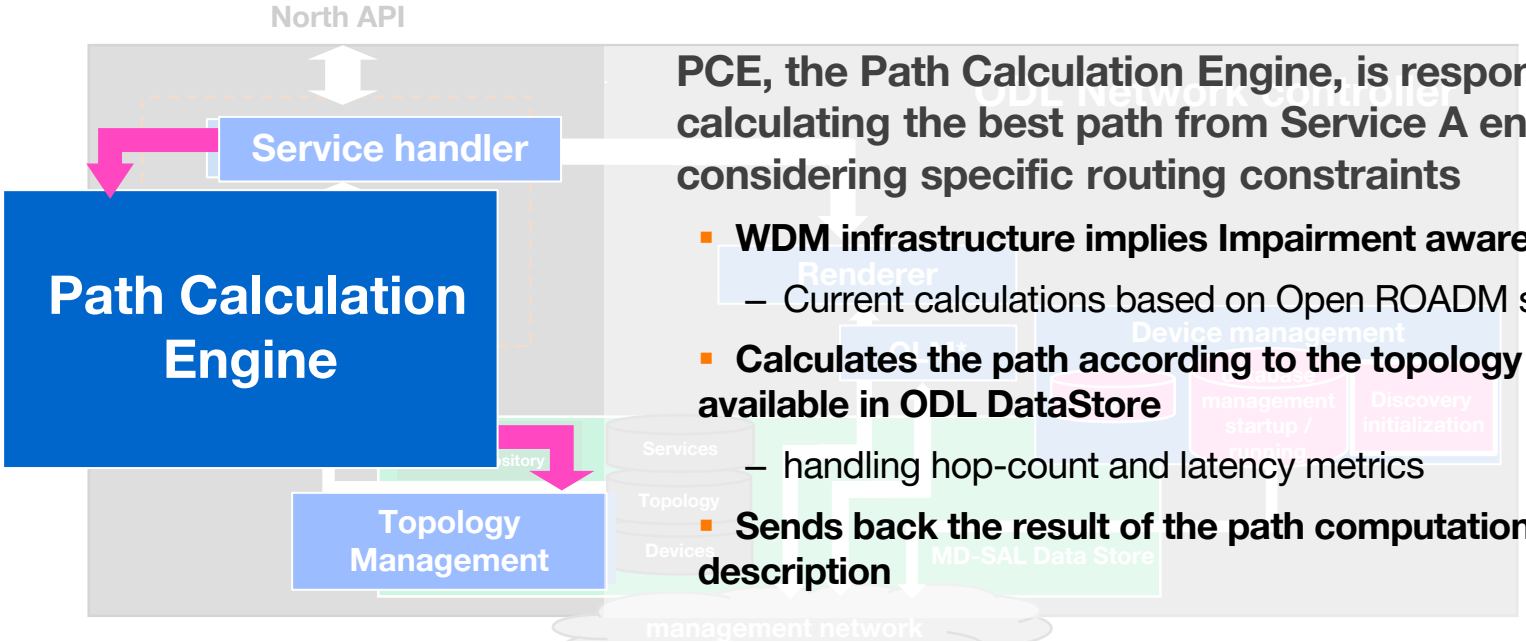

- 1 → Service handler Receives request from north API, checks feasibility and asks for path computation through PCE,
- 2 → Sends to Renderer service implementation requests
- …N → Service Handler receives notifications from Renderer when connection is established and power level set on the interfaces
- N+1 → Service Handler Populates and refreshes service list according to service status after transmission quality has been checked

## SH Manages service handling request coming from the North RESTCONF API

- Service creation, deletion, restoration, reconfiguration and rerouting…

# PCE

- The PCE receives from service handler request for path computation, or service feasibility check




PCE, the Path Calculation Engine, is responsible for calculating the best path from Service A end to Z end, considering specific routing constraints

- **WDM infrastructure implies Impairment aware path calculation**
  - Current calculations based on Open ROADM specifications
- **Calculates the path according to the topology information available in ODL DataStore**
  - handling hop-count and latency metrics
- **Sends back the result of the path computation including path description**

The following constraints are handled in the latest version of tpce

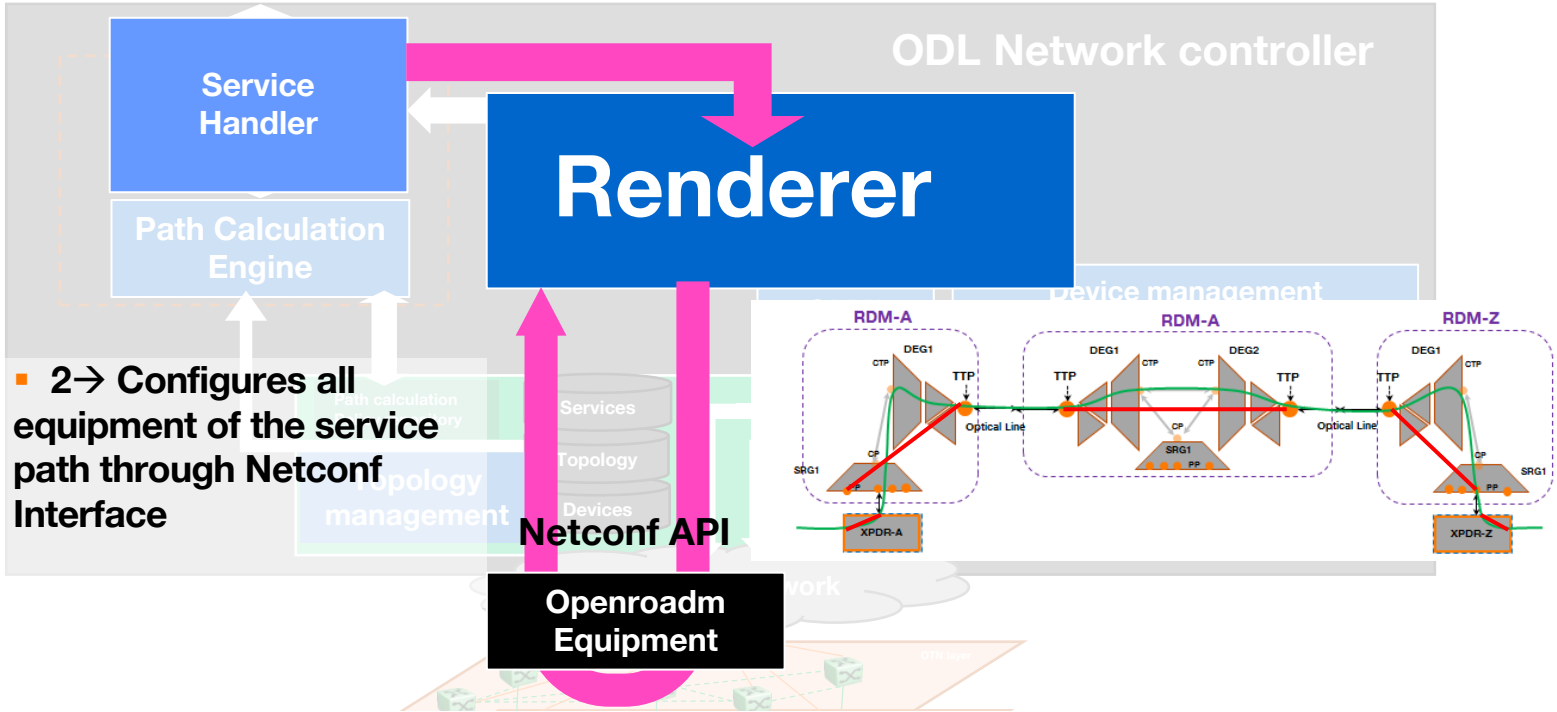
- **General constraints: exclude Node/SRLG, maximum latency**
- **Diversity constraints: with respect to the path(s) of a specific service or a list of service**

orange™

# Renderer

North API

- 1 → Receives Service implementation requests from service handler




- 2→ Configures all equipment of the service path through Netconf Interface

The Renderer configures the service path through the different network elements

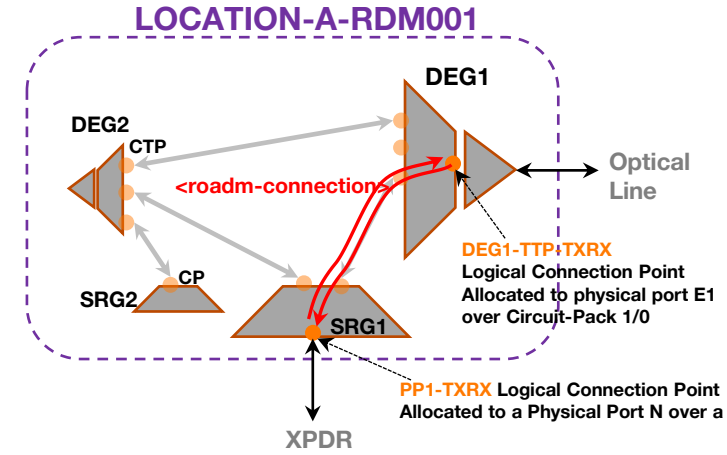
- Handles interfaces & connection creation and deletion



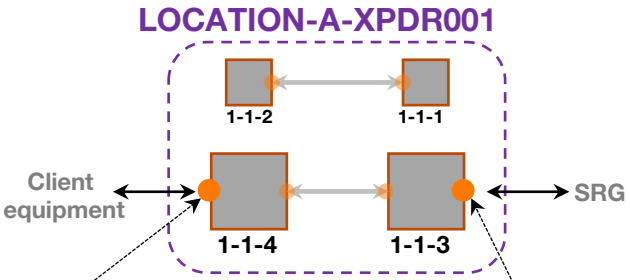
orange™

# Rendering tasks performed during service creation :

## Example of a 100Gbps wavelength (OCH channel) creation




- On DEG1-TTP-TXRX
  - Create OTS interface if does not exist
  - Create OMS interface if does not exist
  - Create Och interface
- On PP1-TXRX
  - Create Och interface
- On LOCATION-A-RDM001
  - Create <roadm-connection> between Och interfaces




- On XPDR-NETWORK1
  - Create Och interface
  - Create OTU interface
  - Create ODU interface
- On XPDR-CLIENT1
  - Create 100G Ethernet interface

XPDR-CLIENT1 Logical Connection Point allocated to a physical port N over a specific Circuit-Pack 1-1-4 (client pluggable module)

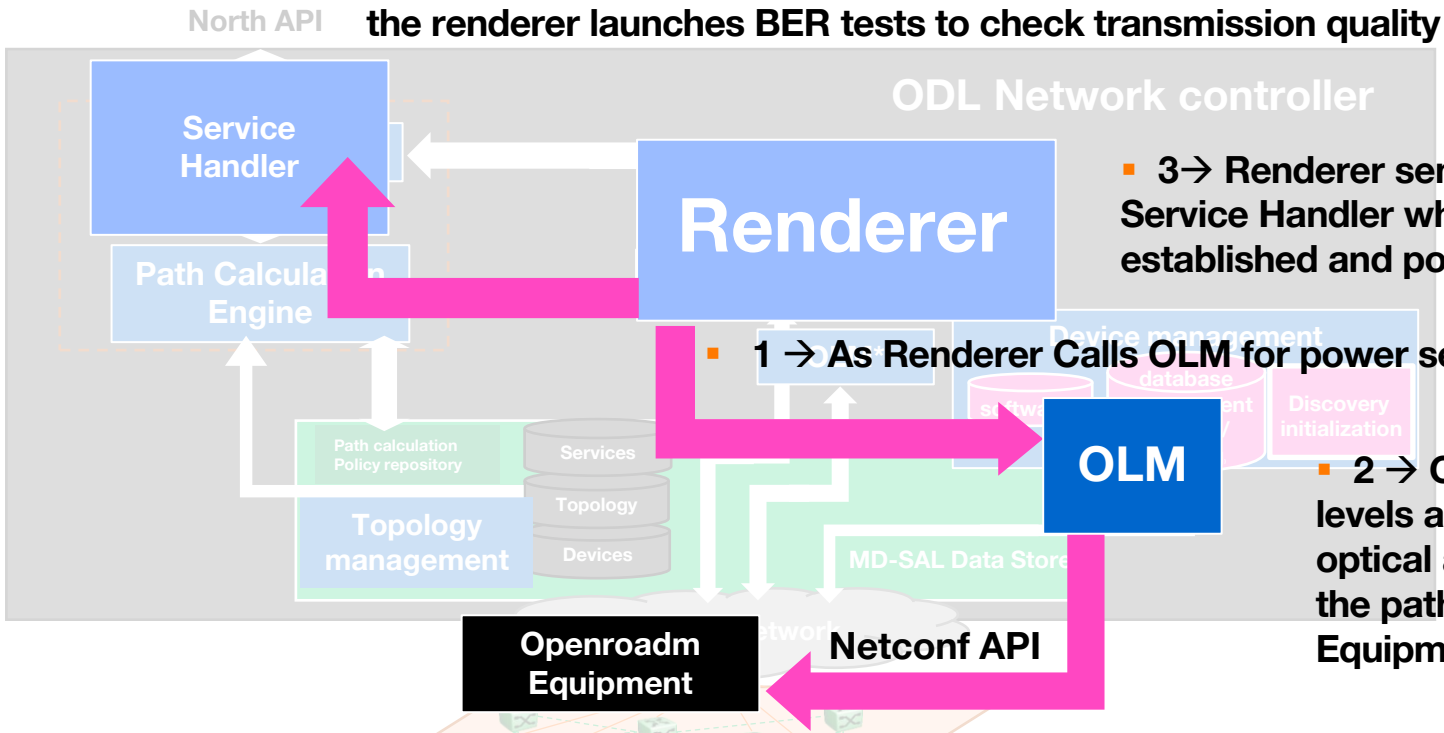
XPDR-NETWORK1 Logical Connection Point allocated to physical port a physical port M over another specific Circuit-Pack 1-1-3 (network pluggable module)



orange™

# OLM : Optical Line management module

- 4→ After interfaces, connections & power levels has been configured on the NEs, the renderer launches BER tests to check transmission quality in both directions




- 3→ Renderer sends notifications to Service Handler when connection is established and power levels are set
- 1 → As Renderer Calls OLM for power settings and adjustment
- 2 → OLM Sets power levels and connection optical attributes along the path through Equipment Netconf API

**OLM is responsible for setting and controlling optical power levels**

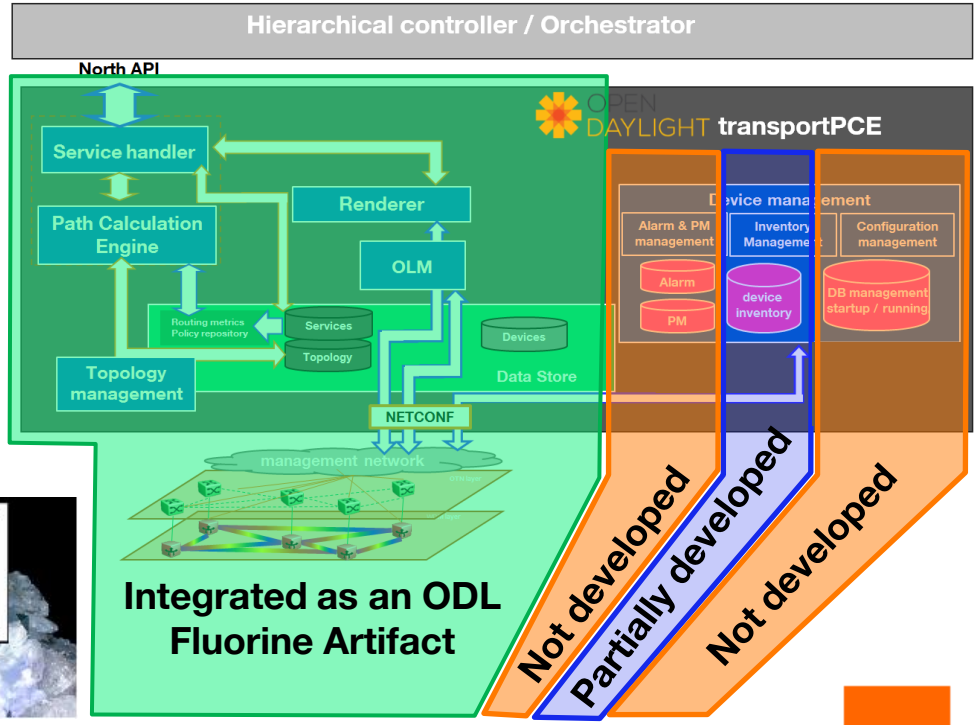
- After interfaces & connection are created, OLM manages power settings of the different optical elements



orange™

# Transport PCE : where are we today ?

- **Most of the bricks defined in the controller architecture are available**
- **Junit and functional tests have been developed for the available modules**
  - Allows checking code robustness
  - Needed for integration in ODL
  - Continuous integration eases collaboration between contributors in different countries, entities & companies
- **Transport PCE code now available as an ODL Fluorine Artifact**




orange™

# Added in Neon ODL Release

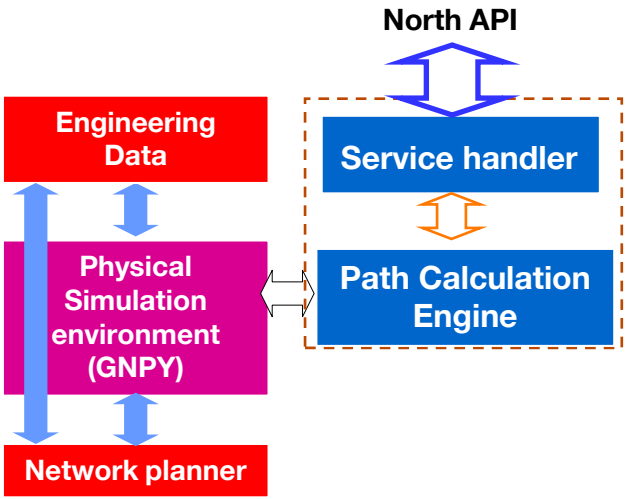
The main features added in Neon release are the following

- Add support for notification in RPC handling
- Extension of the coverage for OpenROADM Service RPC handling: service-reroute, service-restoration, temp-service-create/delete
- Impairment aware path calculation in PCE (OSNR calculation)
- Management of unidirectional ports in path calculation and path configuration
- ROADM to ROADM service creation, for resource reservation when transponders are not present
- Introduction of transportpce-service-path 1.6
- Device version management (up to release 2.2.1)




orange™

# Coming soon… interconnection to GNPY




- **Focus on GNPY interconnection**
  - We plan to use GNPY tool as an external PCE with the capability to handle bookended transponders, low noise amplifiers, and high rate transponders in the future
    - GNPY, a project of the OOPT/PSE working group of the TIP is an open-source, community-developed library for building route planning and optimization tools in real-world mesh optical networks.
    - It uses complex algorithm and will be maintained to include additional features (Raman Amp is an example)
  - We are currently exporting topology and providing the results of a path computation performed by the PCE to GNPY tool
    - Path computed by tpce is converted to routing constraints
  - GNPY returns the result of the performance evaluation on the path
    - feasible or not, evaluated OSNR value

- **Complement interconnection with GNPY tool:**
  - Manage cases where path provided by PCE is not valid in terms of performances
  - Crank back → GNPY proposes a new valid path which does not follow the same route as the original path computed by PCE, but respects original routing constraints provided in service-create → PCE checks path conformance with initial constraints



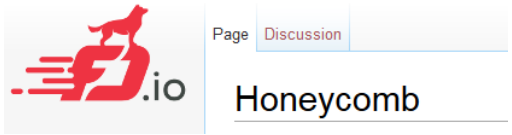
orange™

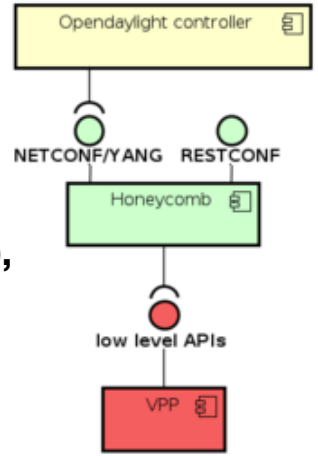
# Mid Term forecasted developments

- **Main priority for mid term developments is Service Assurance**
  - Today, if we handle service creation and deletion, service health is not monitored, meaning that the controller is not able to trigger corrective actions in case of failures or degradation of the service quality.
- **OTN is also identified as a high priority development**
  - Current implementation provides service handling for the WDM layer. OTN services can not be provisioned
  - Performance monitoring and associated proactive service maintenance could be developed in a second step
- **FLEXGRID and higher rates**
  - As per today, if NMC and MC have been introduced with Device 2.2.1 version management, the controller still manage service creation according to a fix grid
  - The target is to handle services of 100G and beyond according to a flexgrid
- **PCE enhancement**
  - Complement constraints handling : soft, co-routing/include  constraints
  - Complement the set of metrics handled by the PCE : distance, load, TE metric…

# Continuous integration and deployment (CICD)

- **Contributors to transportPCE are located in different countries and belong to different companies or different entities in the same company**
  - Some of the contributors can test their code on equipment platforms, some others not
  - Platforms do not include the same equipment /releases
- **Therefore, a set of common tools that can be used by any of the contributors to test the code they push and verify it does not introduce regressions is absolutely needed**
- **Unitary tests allow checking code implementation / semantic**
- **Functional tests allow extending the scope of testing and test functions implemented**
  - They are developed using tools from other projects
    - **ODL Netconf Testtool**
      - suffers from limitations : no possibility to write and modify configurations
    - **fd.io Honeycomb project**
      - Can easily be customized
      - Development of HoneyNode used for functional tests
      - Allows handling configuration and operational Data Stores

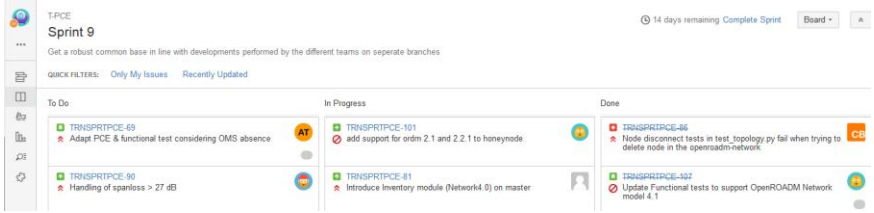



- **Xtesting is used to customize the test environment (Docker, minions, Jenkins, etc...),**
  - Offload functional tests outside the integration framework with a bot accounting with -1/0/+1 verified privilege, and a gerrit-stream events connection

# Thank You

- **For more information :**
  - General information on the project : transportPCE wiki:
    - https://wiki.opendaylight.org/view/TransportPCE:Main
    - still under work , thanks for your comprehension!
  - Documentation :
  - https://docs.opendaylight.org/projects/transportpce/en/stable-fluorine/index.html
    https://docs.opendaylight.org/projects/transportpce/en/latest/
    - User guide
    - Development guide
  - Last code available on Gerrit :
    - https://git.opendaylight.org/gerrit/#/q/project:transportpce
  - Planned developments, bug tracking :
    https://jira.opendaylight.org/secure/RapidBoard.jspa?rapidView=62&projectKey=TRNSPRTPCE




## TransportPCE:Main

### Contents

[hide]


- 1 Description
- 2 Yang models
- 3 Module description and requirements
- 4 Meetings and Action points
- 5 Dependencies on other projects
- 6 Contributing to TransportPCE
- 7 Proof of Concept and demonstrations
- 8 Tests and continuous integration
- 9 Identified bugs & workarounds, Coding enhancements