SDN Platform Wired Installation and User Guide

01.01.02.0021 Version

9034818 Rev 02

Published April 2015

Table of Contents

Preface....................4
Conventions....................4
Providing Feedback to Us....................5
Getting Help....................5

Chapter 1: Overview....................7
Extreme Networks SDN Platform....................7
OneController....................8
Virtual Tenant Network....................9
OpenFlow....................10
OpenStack....................10
Hyperglance....................10

Chapter 2: SDN Network Topology....................11

Chapter 3: Enabling and Configuring OpenFlow....................15

Chapter 4: Setting Up OneController Virtual Tenant Network....................16
Setting Up OneController Virtual Tenant Network....................16

Chapter 5: Setting Up OpenStack....................25
OpenStack/Virtual Tenant Network Support....................25
OpenStack Controller and Compute VM Requirements....................25
Deploying OpenStack OVA Files....................26
Starting OpenStack....................26
Automatic OpenStack to VTN Mapping....................29
Verifying OpenStack Configuration....................30
Verifying Open vSwitch Configuration....................31
Verifying Web Interface....................32
OpenStack VM to VM Communication Using Manual User Mapping....................32
Understanding Automatic VTN Naming Convention....................36

Chapter 6: Setting Up Hyperglance....................37
Installing Hyperglance....................37
Using Hyperglance....................40

Preface

Conventions

This section discusses the conventions used in this guide.

Text Conventions

The following tables list text conventions that are used throughout this guide.

Table 1: Notice Icons

Icon	Notice Type	Alerts you to...
📝	Note	Important features or instructions.
⚡	Caution	Risk of personal injury, system damage, or loss of data.
⚠	Warning	Risk of severe personal injury.
🆕	New	This command or section is new for this release.

Table 2: Text Conventions

Convention	Description
`Screen displays`	This typeface indicates command syntax, or represents information as it appears on the screen.
The words **enter** and **type**	When you see the word "enter" in this guide, you must type something, and then press the Return or Enter key. Do not press the Return or Enter key when an instruction simply says "type."
[Key] names	Key names are written with brackets, such as **[Return]** or **[Esc]**. If you must press two or more keys simultaneously, the key names are linked with a plus sign (+). Example: Press **[Ctrl]**+**[Alt]**+**[Del]**
Words in italicized type	Italics emphasize a point or denote new terms at the place where they are defined in the text. Italics are also used when referring to publication titles.

Platform-Dependent Conventions

Unless otherwise noted, all information applies to all platforms supported by ExtremeXOS software, which are the following:

- BlackDiamond® X series switch
- BlackDiamond 8800 series switches

- Cell Site Routers (E4G-200 and E4G-400)
- Summit® family switches
- SummitStack™

When a feature or feature implementation applies to specific platforms, the specific platform is noted in the heading for the section describing that implementation in the ExtremeXOS command documentation. In many cases, although the command is available on all platforms, each platform uses specific keywords. These keywords specific to each platform are shown in the Syntax Description and discussed in the Usage Guidelines.

Terminology

When features, functionality, or operation is specific to a switch family, the family name is used. Explanations about features and operations that are the same across all product families simply refer to the product as the "switch."

Providing Feedback to Us

We are always striving to improve our documentation and help you work better, so we want to hear from you! We welcome all feedback but especially want to know about:

- Content errors or confusing or conflicting information.
- Ideas for improvements to our documentation so you can find the information you need faster.
- Broken links or usability issues.

If you would like to provide feedback to the Extreme Networks Information Development team about this document, please contact us using our short online feedback form. You can also email us directly at InternalInfoDev@extremenetworks.com.

Getting Help

If you require assistance, contact Extreme Networks Global Technical Assistance Center using one of the following methods:

Web	www.extremenetworks.com/support
Phone	1-800-872-8440 (toll-free in U.S. and Canada) or 1-603-952-5000 For the Extreme Networks support phone number in your country: www.extremenetworks.com/support/contact
Email	support@extremenetworks.com To expedite your message, enter the product name or model number in the subject line.

Before contacting Extreme Networks for technical support, have the following information ready:

- Your Extreme Networks service contract number
- A description of the failure
- A description of any action(s) already taken to resolve the problem (for example, changing mode switches or rebooting the unit)
- The serial and revision numbers of all involved Extreme Networks products in the network

- A description of your network environment (such as layout, cable type, other relevant environmental information)
- Network load and frame size at the time of trouble (if known)
- The device history (for example, if you have returned the device before, or if this is a recurring problem)
- Any previous Return Material Authorization (RMA) numbers

1 Overview

Extreme Networks SDN Platform
OneController
Virtual Tenant Network
OpenFlow
OpenStack
Hyperglance

This guide provides step-by-step instructions for installing and configuring an example wired SDN developer setup using Extreme Networks OneController™ (based on the OpenDaylight Helium release).

This document assumes that you are familiar with an ESXi operating system. A network diagram is included that depicts the exact physical connections between the switches and ESXi server (see SDN Network Topology on page 11). Switch configurations are included and are the recommended working configurations.

Extreme Networks SDN Platform

Extreme Networks SDN platform is an evolutionary software defined networking (SDN) platform to promote community-led innovation. We believe customers should be able to migrate their existing (brownfield) networks to SDN without expensive forklift upgrades. Seamless SDN migration requires both an open, standards-based, and comprehensive approach with strong community mindshare and partner ecosystem support. Extreme Networks' SDN platform is based on a comprehensive, hardened OpenDaylight (ODL) controller that uniquely includes: network management, network access control, application analytics, and wireless controller technology. Extreme Networks comprehensive approach preserves the integrity of the open API provided by ODL, while extending data center orchestration, automation and provisioning to the entire network under a single pane of glass.

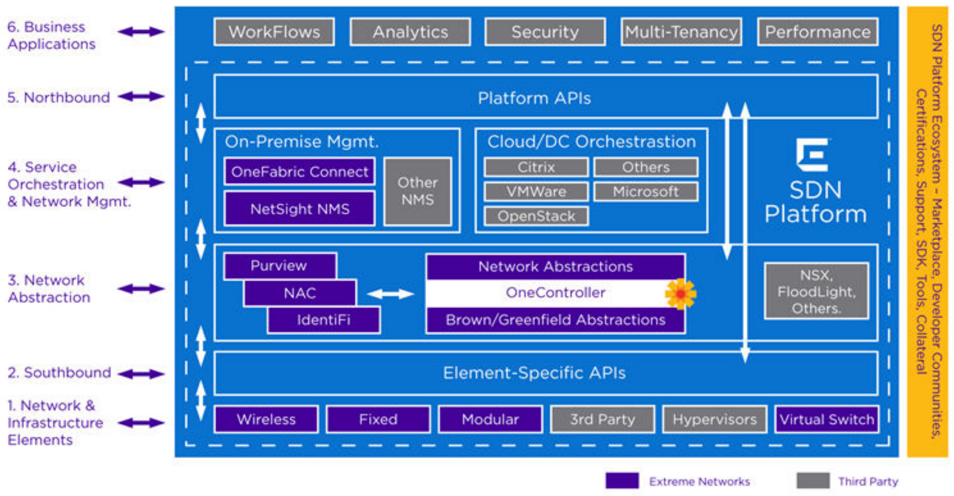


Figure 1: Extreme Networks' SDN Platform

OneController

OneController leverages the OpenDaylight SDN Controller to provide an open, fully pluggable and scalable platform to enable SDN and NFV for networks at any size and scale. OneController's REST (web-based) API provides you with the ability to create applications using the programming language of your choice to program and control your network. These applications use OneController to gather network intelligence, run algorithms to perform analytics, and then use OneController to orchestrate the new rules, if any, throughout the network. Additionally, OneController is based on the modular OpenDaylight platform that allows multiple Java modules to run concurrently within the OSGI framework, and lets the modules access Java APIs exposed by other modules using the OpenDaylight Service Layer Abstraction framework. The southbound interface is capable of supporting multiple protocols (as separate plugins), for example: OpenFlow 1.0, OpenFlow 1.3, SNMP, SOAP/XML, etc. These modules are dynamically linked into a Service Abstraction Layer (SAL). The SAL exposes device services to which the modules north of it are written. The SAL determines how to fulfill the requested service regardless of the underlying protocol used between the controller and the network devices. The OneController framework contains a collection of dynamically pluggable modules to provide network services such as:

- Host and node service
- Flow service
- Physical and overlay (flow-based) topology service
- Path service to setup and manage a path based on specified constraints such as bandwidth between a given source and destination
- Multi-tenant network virtualization service
- Network statistics service

In addition, platform-oriented services and other extensions can also be inserted into the controller platform for enhanced SDN functionality.

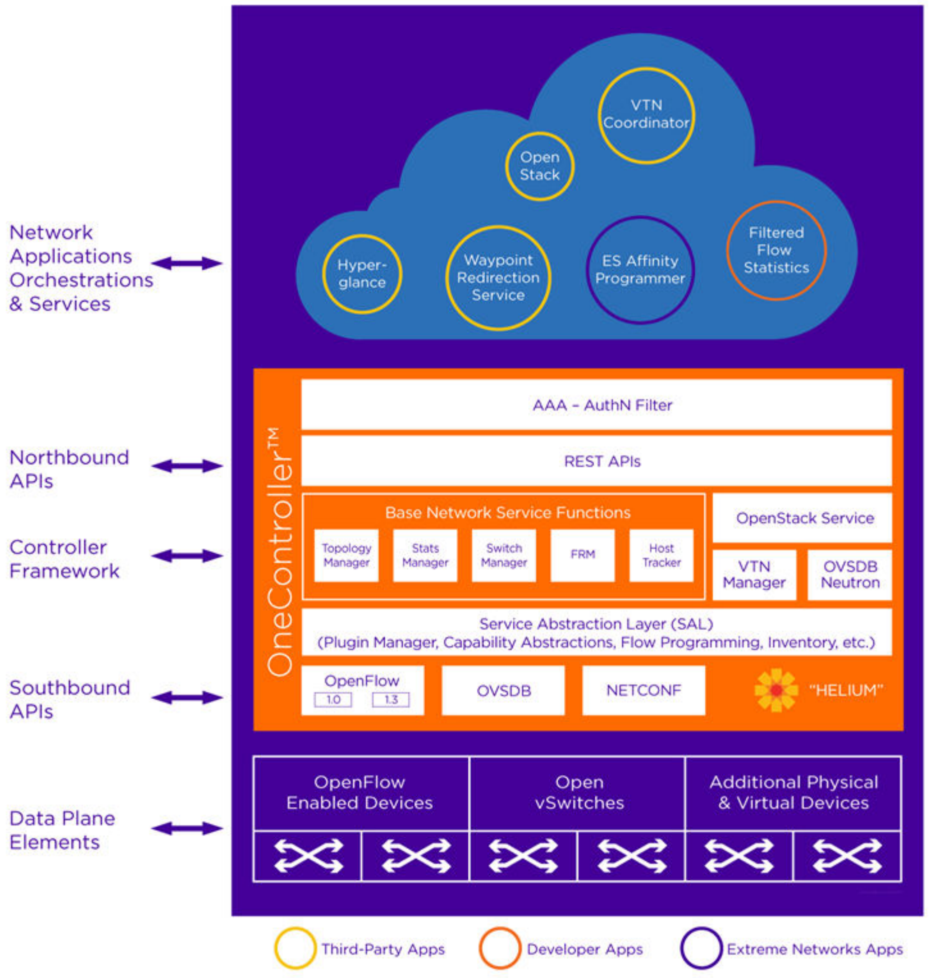


Figure 2: Extreme Networks' OneController

Virtual Tenant Network

Virtual Tenant Network (VTN) is a set of technology components that provides a multi-tenant virtual network through a controller. Traditionally, physical networks have been configured as silos for each department within an organization (or for each customer) by a service provider. This incurs huge, unneeded hardware investments and operating expenses due to the underutilized, redundant network equipment required to implement this scheme. The VTN addresses this problem by providing an abstraction that enables the complete separation of logical plane from physical plane of the network. This allows users to design and deploy virtual networks for their customers without needing to know the physical network topology or underlying operating characteristics. Once a network is defined using VTN, it is automatically mapped onto the underlying physical network by the controller. VTN in the Extreme Networks SDN Platform is an open source product from the OpenDaylight project (for more information, see https://wiki.opendaylight.org/view/OpenDaylight_Virtual_Tenant_Network_(VTN)).

OpenFlow

OpenFlow enables remote controllers to determine the path of network packets through the network of switches, rather than the switches making this determination. This separation of the control logic (handled by the remote controller) from the forwarding decisions (handled by switches) allows for more sophisticated traffic management than is feasible using access control lists (ACLs) and routing protocols. Also, OpenFlow allows switches from different suppliers—often each with their own proprietary interfaces and scripting languages—to be managed remotely using a single, open protocol.

OpenStack

OpenStack is a free and open-source cloud computing software platform. Used primarily as an infrastructure as a service (IaaS) solution, it offers to customers computers—physical, or more often, virtual machines—and other resources according to the customers' varying requirements, providing the ability to scale services up and down. The technology consists of a series of interrelated projects that control pools of processing, storage, and networking resources throughout a data center, which users manage through a web-based dashboard, command-line tools, or a RESTful API.

Hyperglance

Hyperglance provides a GUI-based, 3-D cloud visibility solution to simplify managing networks by aggregating and dynamically synchronizing data for real-time, multidimensional visualization, navigation, analysis, and control at scale. Hyperglance in the Extreme Networks SDN Platform is a commercial product from the Real Status (for more information, see http://www.real-status.com/product).

2 SDN Network Topology

Important
Diagram connections are exact; if you use different connections you need to change switch configurations and several commands provided in this document. It is recommended that you use the same port numbers, addresses, and naming conventions whenever possible.

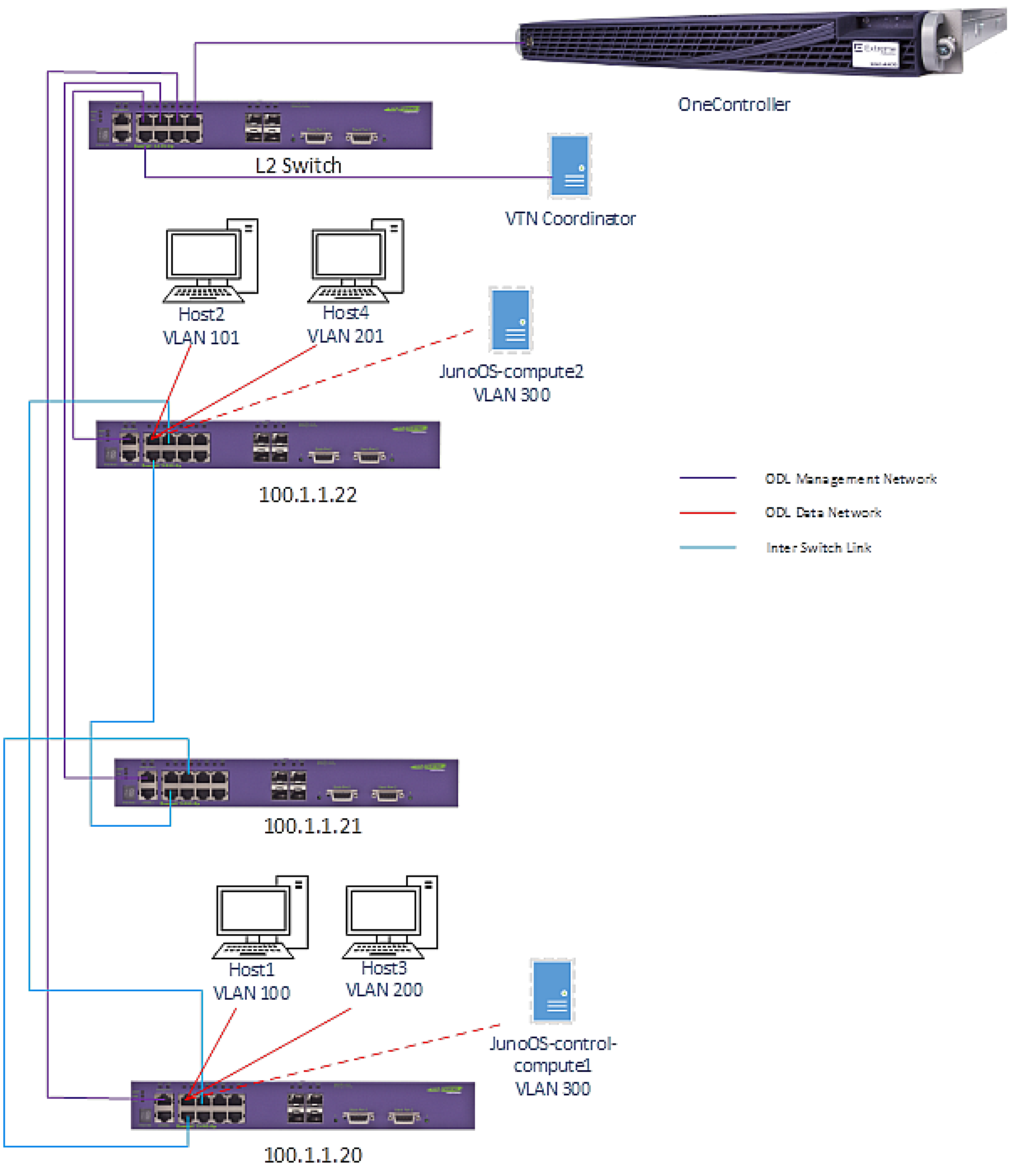


Figure 3: SDN Network Topology

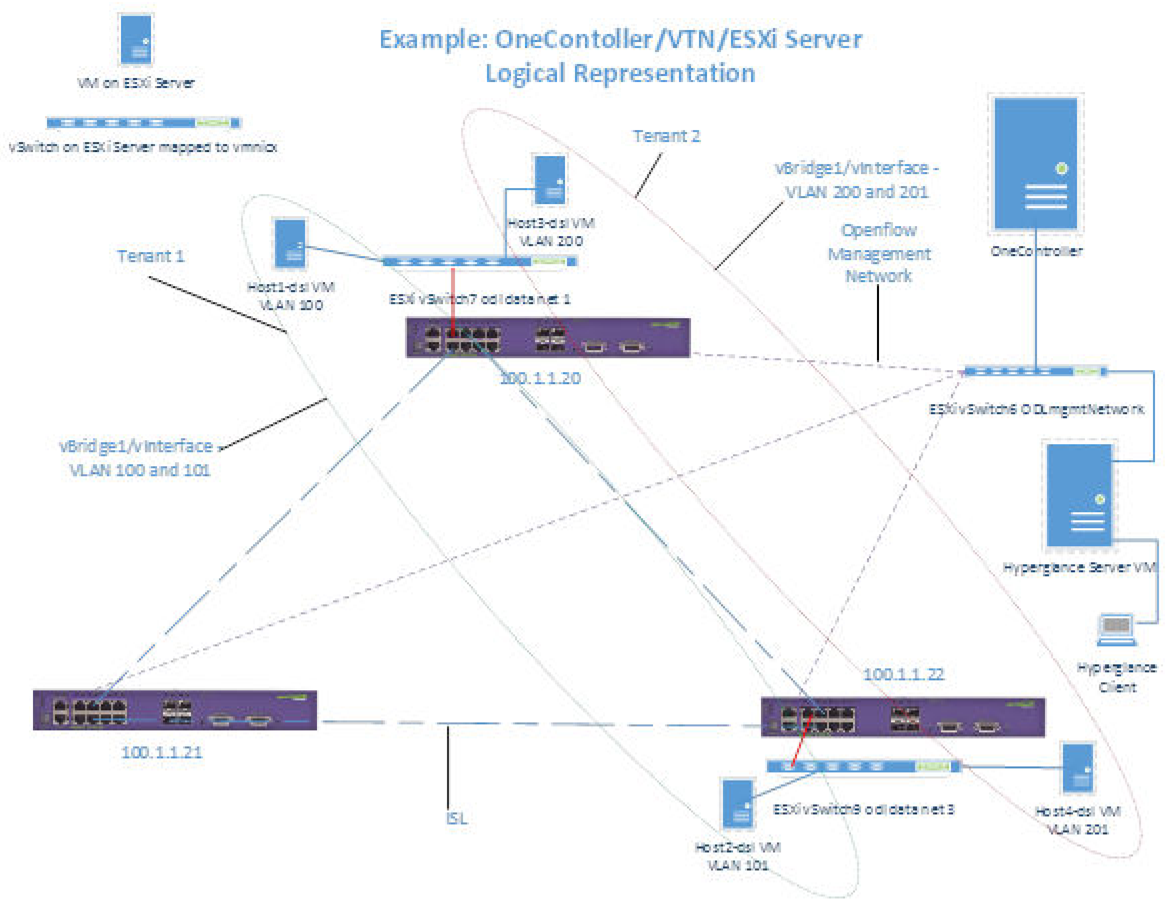

Example: OneContoller/VTN/ESXi Server Logical Representation
VM on ESXi Server
vSwitch on ESXi Server mapped to vmnicx
Tenant 2
vBridge1/vInterface - VLAN 200 and 201
Openflow Management Network
OneController
Host3-dsl VM VLAN 200
Tenant 1
Host1-dsl VM VLAN 100
ESXi vSwitch7 odl data net 1
100.1.1.20
ESXi vSwitch6 ODLmgmtNetwork
vBridge1/vInterface VLAN 100 and 101
Hypergiance Server VM
Hypergiance Client
100.1.1.22
100.1.1.21
ISL
ESXi vSwitch9 odl data net 3
Host2-dsl VM VLAN 101
Host4-dsl VM VLAN 201

Figure 4: OneController/VTN/ESXi Server Logical Representation

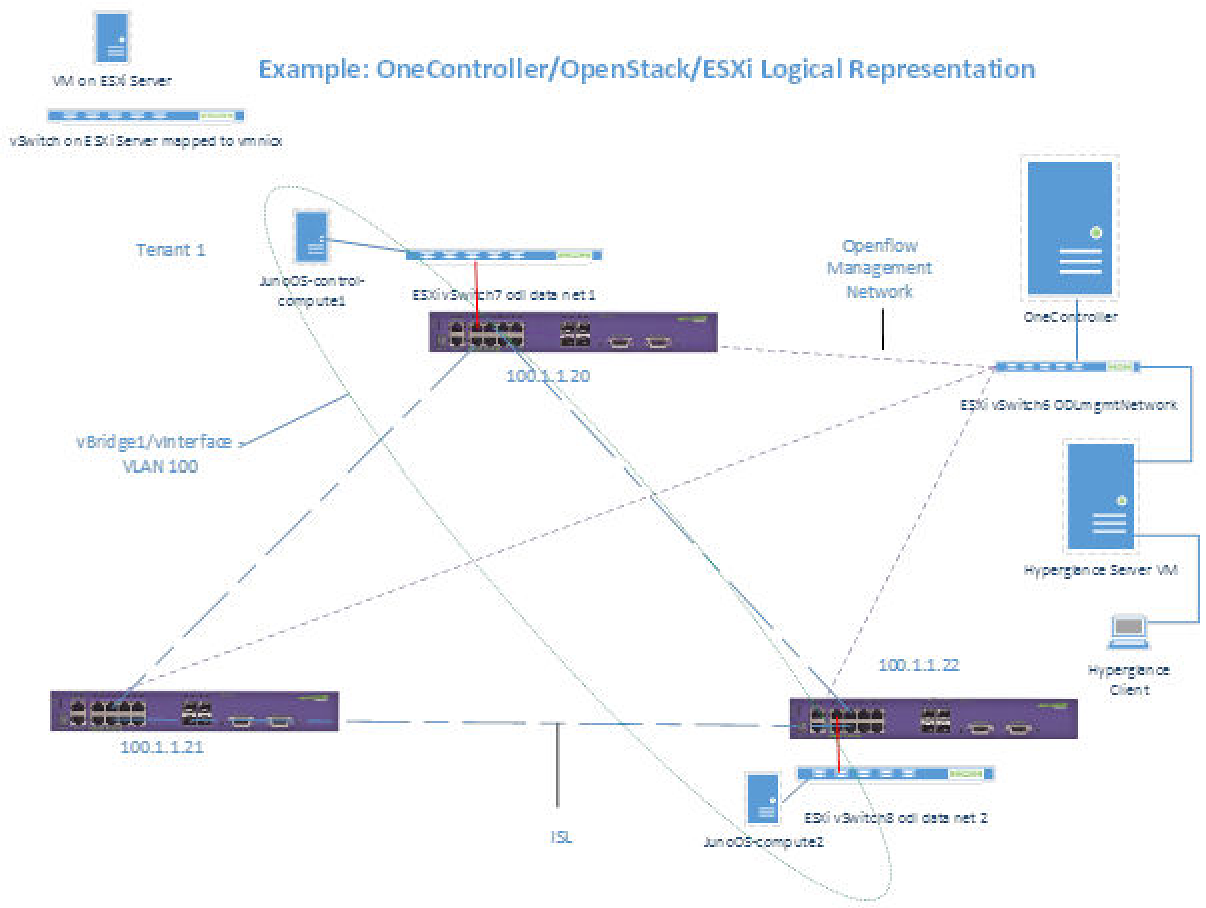

VM on ESXi Server
vSwitch on ESXi Server mapped to vmnics
Example: OneController/OpenStack/ESXi Logical Representation
Tenant 1
JunoOS-control-compute1
ESXi vSwitch7 odl data net 1
Openflow Management Network
OneController
100.1.1.20
ESXi vSwitch6 ODLmgmtNetwork
vBridge1/vInterface VLAN 100
Hypergiance Server VM
Hypergiance Client
100.1.1.22
100.1.1.21
ESXi vSwitch8 odl data net 2
JunoOS-compute2
ISL

Figure 5: OneController/OpenStack/ESXi Server Logical Representation

3 Enabling and Configuring OpenFlow

This section demonstrates how to successfully configure OpenFlow on ExtremeXOS operating system. You must have previously installed the OpenFlow XMOD file and enabled it with a license key.

Note
For complete OpenFlow configuration options, see the *ExtremeXOS User Guide* at www.extremenetworks.com/documentation.

To enable and configure OpenFlow:

1 Configure the VLANs that you wish OpenFlow to be enabled on.

Note
Use the VLAN numbers specified in the topology diagrams (see SDN Network Topology on page 11).

2 Type the command **`configure access-list`** ***`width double`***.

3 Enable OpenFlow by entering the command **`enable openflow`**.

4 Configure the OpenFlow OneController parameters on the switch:

   a Type the command **`configure openflow controller primary out-of-band active ipaddress`** `x.x.x.x` **`vr-x`**.

   b Type the command **`enable openflow vlan`** `vlanx` (enables OpenFlow on your selected VLANs).

   c Type the command **`show openflow`**. The switch should show a status of `ACTIVE` if the switch is connected to the OneController. A status of `BACKOFF` defines an unconnected switch.

4 Setting Up OneController Virtual Tenant Network

Setting Up OneController Virtual Tenant Network

This section details the installation and configuration settings required to successfully deploy OneController with Virtual Tenant Network (VTN).

You must have a working centOS7 VM installed on an ESXi VM server along with OneController and the VTN coordinator.

The following network topology depicts the physical interconnections. Note that the port, VLAN, and switch MAC addresses are examples.

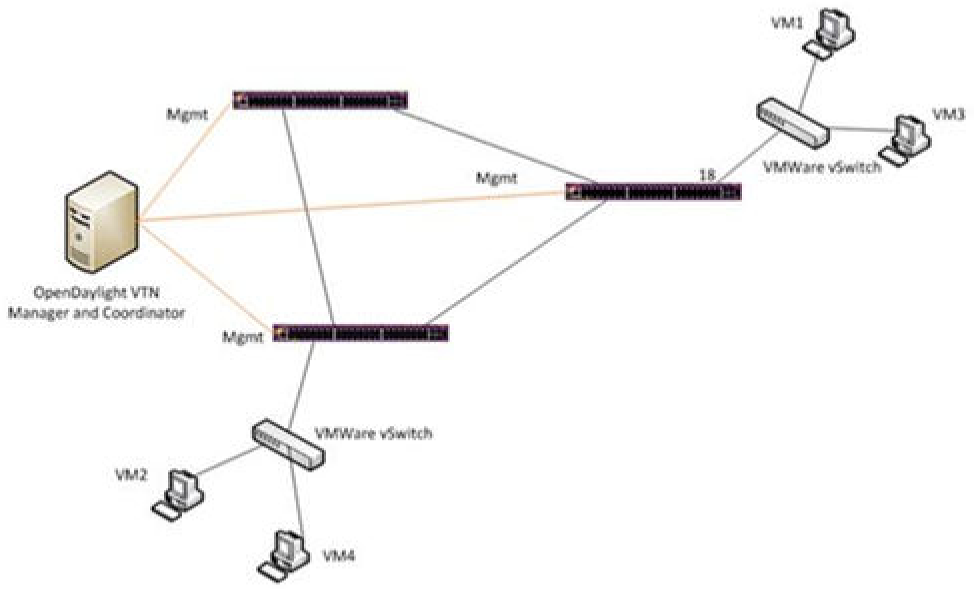


Figure 6: OneController/VTN Topology

Setting Up OneController Virtual Tenant Network

To set up OneController VTN:

1 After deploying (EXTR-OneC-V-DA-1.0.ova), power on of OneController using the ESXi server.
2 Start OneController.
3 Log on to OneController as an administrator (user name = admin, password = abc123).

4 Log on to Karaf (password = karaf).

```
admin@OC.extremenetworks.com:~$ ssh karaf@127.0.0.1 -p 8101
Password authentication
Password: (karaf)
```

5 Install the following features:

```
opendaylight-user@root>feature:install odl-adsal-compatibility-all
odl-nsf-all odl-vtn-manager-all
```

6 Verify that all features have been installed:

```
opendaylight-user@root>feature:list –i
```

7 After features have been installed, log on to the DLUX web GUI through a web browser (user name = admin, password = abc123):

```
http://<your-ip-address-here>:8181/dlux/index.html
```

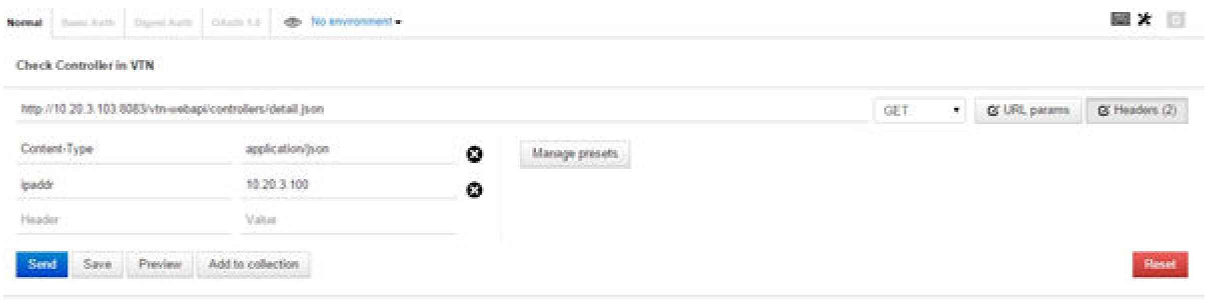
8 After deploying (EXTR-VTNCoordinator-DAKit.ova), power on of the VTN Coordinator using the ESXi server.

9 Start VTN coordinator and log on as root (user name = root, password = extreme).

10 Start and check VTN coordinator:

```
./usr/local/vtn/bin/vtn_start
curl -X GET -H 'content-type: application/json' -H 'username: admin' -
H
'password: adminpass' -H 'ipaddr:127.0.0.1'
http://127.0.0.1:8083/vtn-webapi/api_version.json | python -mjson.tool
```

Postman configuration, substitute 10.20.3.103 for the address of the VTN coordinator.



Check Controller in VTN

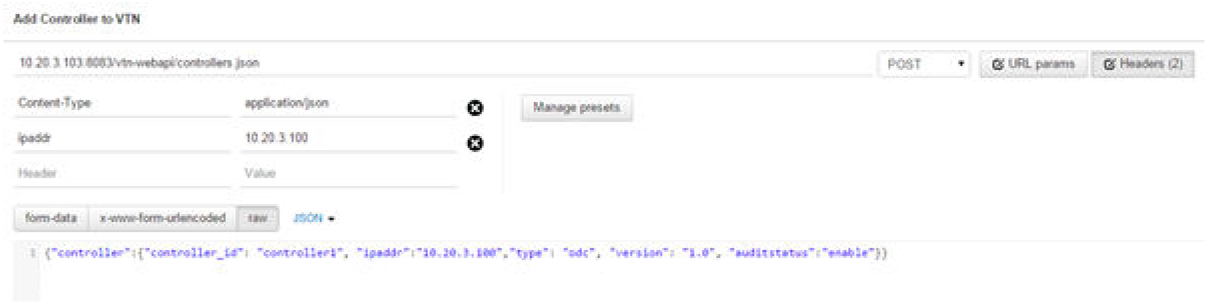
http://10.20.3.103:8083/vtn-webapi/controllers/detail.json GET

Header	Value
Content-Type	application/json
ipaddr	10.20.3.100

Figure 7: Start and Check VTN Coordinator

11 Add OneController to the VTN coordinator and show the configured controller:

```
curl -v -X POST -H 'content-type: application/json' -H 'username:
admin' -H
'password: adminpass' -H 'ipaddr:<onecontroller-ip>' -d
'{"controller":
{"controller_id": "controller1", "ipaddr":"<onecontroller-ip>",
"type": "odc",
"version": "1.0", "auditstatus":"enable"}}' http://127.0.0.1:8083/vtn-
webapi/controllers.json
```



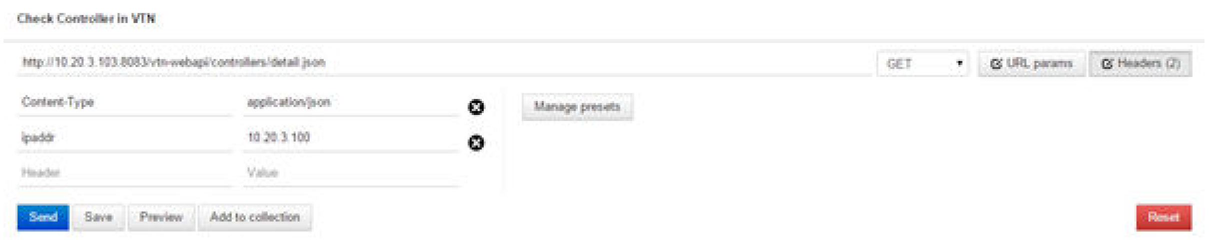
10.20.3.103:8083/vtn-webapi/controllers.json

Content-Type	application/json
ipaddr	10.20.3.100
Header	Value

```
{"controller":{"controller_id": "controller1", "ipaddr":"10.20.3.100","type": "odc", "version": "1.0", "auditstatus":"enable"}}
```

Figure 8: Add OneController to VTN Coordinator

```
curl -X GET -H 'content-type: application/json' -H 'username: admin' -
H
 'password: adminpass'   -H 'ipaddr:<onecontroller-ip>'
http://127.0.0.1:8083/vtn-webapi/controllers/detail.json | python -
mjson.tool
```



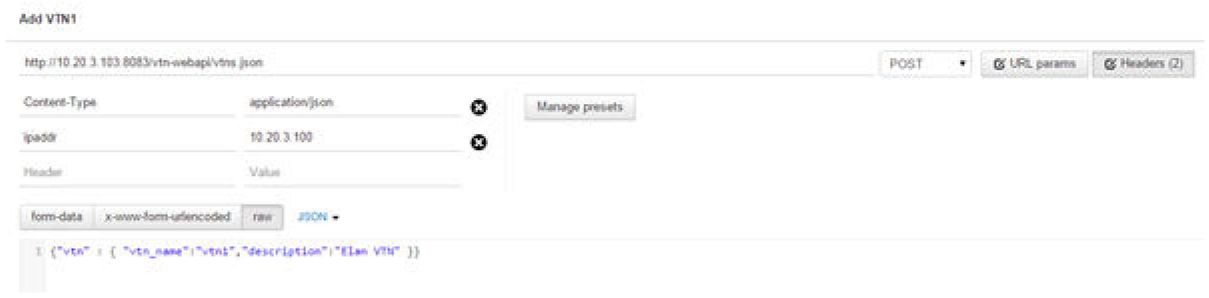
http://10.20.3.103:8083/vtn-webapi/controllers/detail.json

Content-Type	application/json
ipaddr	10.20.3.100
Header	Value

Figure 9: Check OneController in VTN

12 Create VTN VTN1:

```
curl -v -X POST -H 'content-type: application/json' -H 'username:
admin' -H
 'password: adminpass' -H 'ipaddr:<onecontroller-ip>' -d '{"vtn" :
{"vtn_name":"vtn1","description":"Elan VTN" }}' http://127.0.0.1:8083/
vtn-webapi/vtns.json
```



http://10.20.3.103:8083/vtn-webapi/vtns.json

Content-Type	application/json
ipaddr	10.20.3.100
Header	Value

```
{"vtn" : { "vtn_name":"vtn1","description":"Elan VTN" }}
```

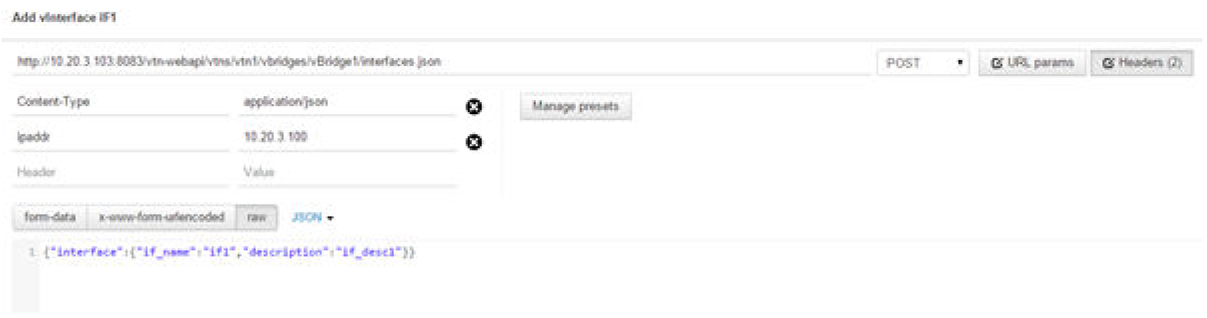
Figure 10: Add VTN1

13 Add vbridge vBridge1:

```
curl -v -X POST -H 'content-type: application/json' -H 'username:
admin' -H
'password: adminpass' -H 'ipaddr:<onecontroller-ip>' -d '{"vbridge" :
{"vbr_name":"vBridge1","controller_id":"controller1","domain_id":"(DEF
AULT)" }}
' http://127.0.0.1:8083/vtn-webapi/vtns/vtn1/vbridges.json
```

14 Add vInterface IF1 vBridge1:

```
curl -v -X POST -H 'content-type: application/json' -H 'username:
admin' -H
'password: adminpass' -H 'ipaddr:<onecontroller-ip>' -d
'{"interface":
{"if_name": "if1","description": "if_desc1"}}'
http://127.0.0.1:8083/vtn-webapi/vtns/vtn1/vbridges/vBridge1/
interfaces.json
```


Add vInterface IF1

http://10.20.3.103:8083/vtn-webapi/vtns/vtn1/vbridges/vBridge1/interfaces.json

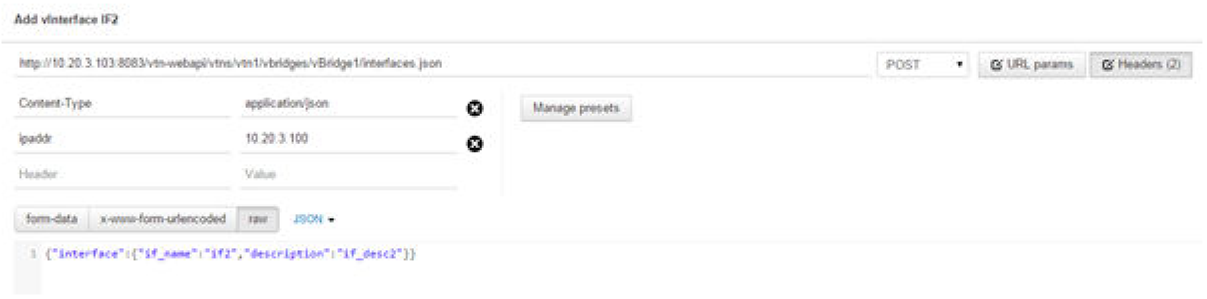
Content-Type	application/json
ipaddr	10.20.3.100
Header	Value

```
{"interface":{"if_name":"if1","description":"if_desc1"}}
```

Figure 11: Add vInterface IF1

15 Add vInterface IF2 to vBridge1:

```
curl -v -X POST -H 'content-type: application/json' -H
'username: admin' -H 'password: adminpass' -H 'ipaddr:<onecontroller-
ip>' -d
 '{"interface": {"if_name": "if2","description": "if_desc2"}}'
http://127.0.0.1:8083/vtn-webapi/vtns/vtn1/vbridges/vBridge1/
interfaces.json
```


Add vInterface IF2

http://10.20.3.103:8083/vtn-webapi/vtns/vtn1/vbridges/vBridge1/interfaces.json

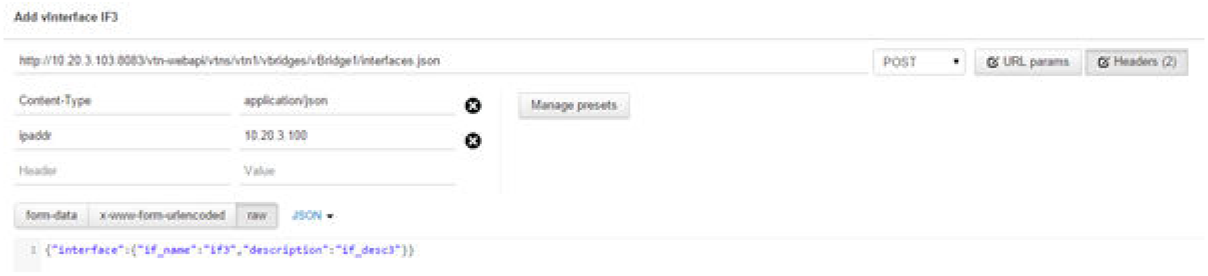
Content-Type	application/json
ipaddr	10.20.3.100
Header	Value

```
{"interface":{"if_name":"if2","description":"if_desc2"}}
```

Figure 12: Add vInterface IF2

16 Add vInterface IF3 vBridge1:

```
curl -v -X POST -H 'content-type: application/json' -H 'username:
admin' -H
'password: adminpass' -H 'ipaddr:<onecontroller-ip>' -d
'{"interface":
{"if_name": "if3","description": "if_desc3"}}'
http://127.0.0.1:8083/vtn-webapi/vtns/vtn1/vbridges/vBridge1/
interfaces.json
```



Add vInterface IF3

http://10.20.3.103:8083/vtn-webapi/vtns/vtn1/vbridges/vBridge1/interfaces.json

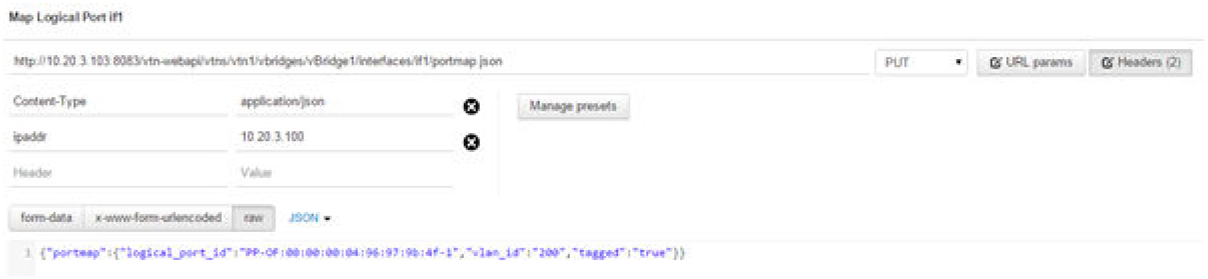
Content-Type	application/json
ipaddr	10.20.3.100
Header	Value

```
{"interface":{"if_name":"if3","description":"if_desc3"}}
```

Figure 13: Add vInterface IF3

17 Map the logical port to interface IF1:

```
curl -v -X PUT -H 'content-type: application/json' -H 'username:
admin' -H
 'password: adminpass' -H 'ipaddr:<onecontroller-ip>' -d
'{"portmap":{"logical_port_id": "PP-OF:00:00:<switch-mac-address-
here>-<port id here>",
"vlan_id": "100","tagged": "true"}}'
http://127.0.0.1:8083/vtn-webapi/vtns/vtn1/vbridges/vBridge1/
interfaces/if1/portmap.json
```



Map Logical Port if1

http://10.20.3.103:8083/vtn-webapi/vtns/vtn1/vbridges/vBridge1/interfaces/if1/portmap.json

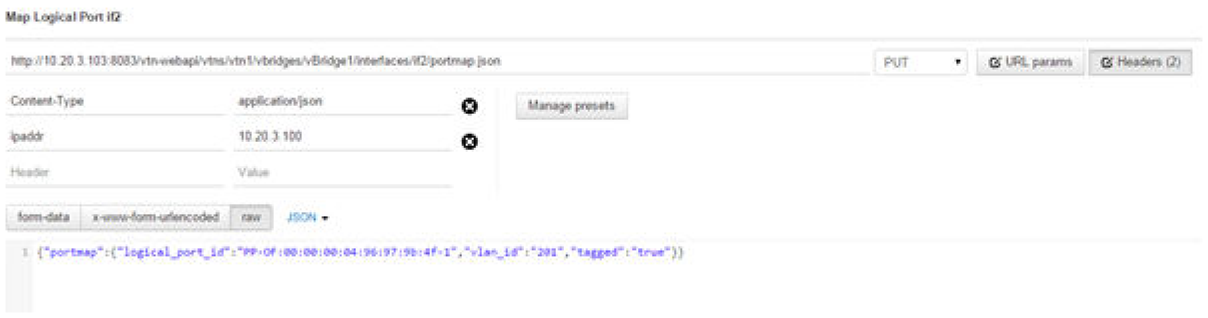
Content-Type	application/json
ipaddr	10.20.3.100
Header	Value

```
{"portmap":{"logical_port_id":"PP-OF:00:00:00:04:96:97:9b:4f-1","vlan_id":"200","tagged":"true"}}
```

Figure 14: Map Logical Port to Interface IF1

18 Map the logical port to interface IF2:

```
curl -v -X PUT -H 'content-type: application/json' -H
 'username: admin' -H 'password: adminpass' -H 'ipaddr:<onecontroller-
ip>' -d
 '{"portmap":{"logical_port_id":
"PP-OF:00:00:<switch-mac-address-here>-<port id here>","vlan_id":
"101","tagged": "true"}}'
http://127.0.0.1:8083/vtn-webapi/vtns/vtn1/vbridges/vBridge1/
interfaces/if2/portmap.json
```



Map Logical Port if2

http://10.20.3.103:8083/vtn-webapi/vtns/vtn1/vbridges/vBridge1/interfaces/if2/portmap.json

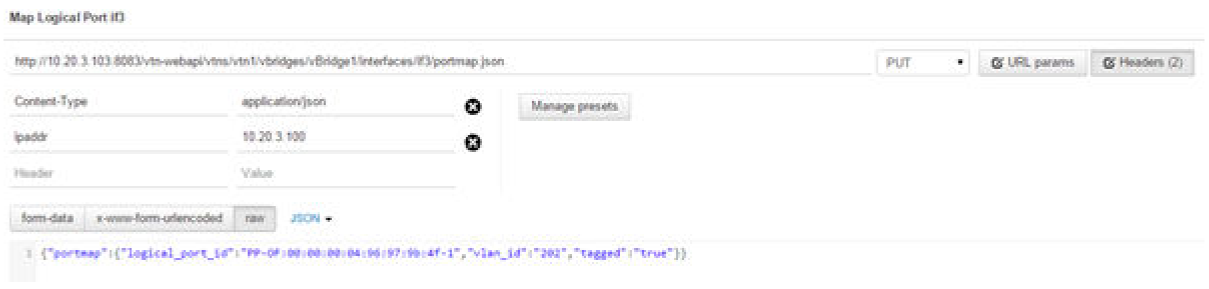
Content-Type	application/json
ipaddr	10.20.3.100
Header	Value

```
{"portmap":{"logical_port_id":"PP-OF:00:00:00:04:96:97:9b:4f-1","vlan_id":"201","tagged":"true"}}
```

Figure 15: Map Logical Port to Interface IF2

19 Map the logical port to interface IF3:

```
curl -v -X PUT -H 'content-type: application/json' -H 'username:
admin' -H
 'password: adminpass' -H 'ipaddr:<onecontroller-ip>' -d
'{"portmap":{"logical_port_id": "PP-OF:00:00:<switch-mac-address-
here>-<port id here>",
"vlan_id": "101","tagged": "true"}}'
http://127.0.0.1:8083/vtn-webapi/vtns/vtn1/vbridges/vBridge1/
interfaces/if3/portmap.json
```



Map Logical Port IF3

http://10.20.3.103:8083/vtn-webapi/vtns/vtn1/vbridges/vBridge1/interfaces/if3/portmap.json — PUT

Content-Type	application/json
ipaddr	10.20.3.100

raw — JSON

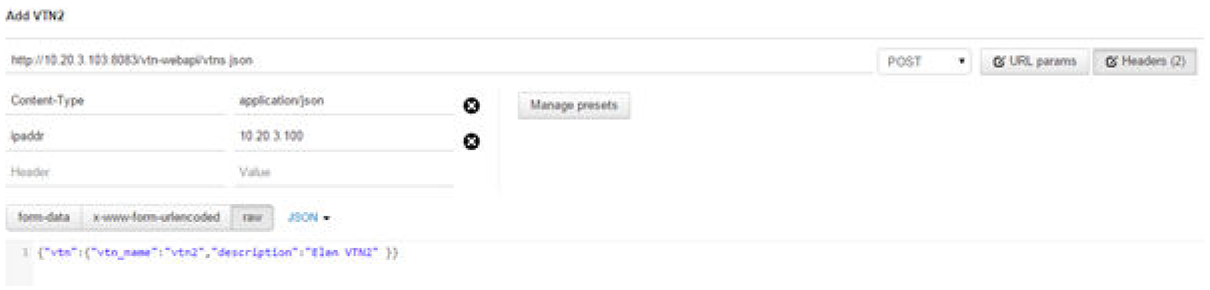
```
{"portmap":{"logical_port_id":"PP-OF:00:00:00:04:96:97:9b:4f-1","vlan_id":"202","tagged":"true"}}
```

Figure 16: Map Logical Port to Interface IF3

IF1 is mapped to port 1 with VLAN 100 on the Summit X440 (OF|02:77) and IF2 is mapped to port 1 with VLAN 101 on a different Summit X440 (OF|ee:7c).

20 Create VTN VTN2:

```
curl -v -X POST -H 'content-type: application/json' -H
 'username: admin' -H 'password: adminpass' -H 'ipaddr:<onecontroller-
ip>' -d
 '{"vtn" : {"vtn_name":"vtn2","description":"Elan VTN 2" }}'
http://127.0.0.1:8083/vtn-webapi/vtns.json
```



Add VTN2

http://10.20.3.103:8083/vtn-webapi/vtns.json — POST

Content-Type	application/json
ipaddr	10.20.3.100

raw — JSON

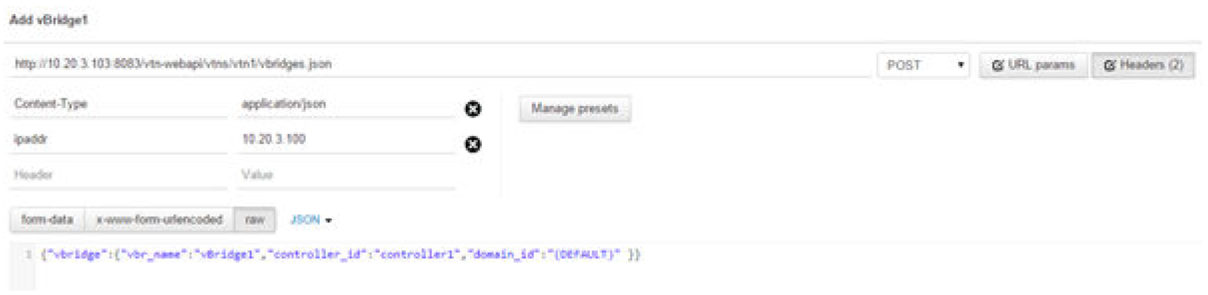
```
{"vtn":{"vtn_name":"vtn2","description":"Elan VTN2" }}
```

Figure 17: Add VTN2

21 Add vbridge vBridge1:

```
curl -v -X POST -H 'content-type: application/json' -H 'username:
admin' -H
 'password: adminpass' -H 'ipaddr:<onecontroller-ip>' -d
'{"vbridge" :
{"vbr_name":"vBridge1","controller_id":"controller1","domain_id":"(DEF
AULT)" }}'
http://127.0.0.1:8083/vtn-webapi/vtns/vtn2/vbridges.json
```

Figure 18: Add vBridge1



Add vBridge1

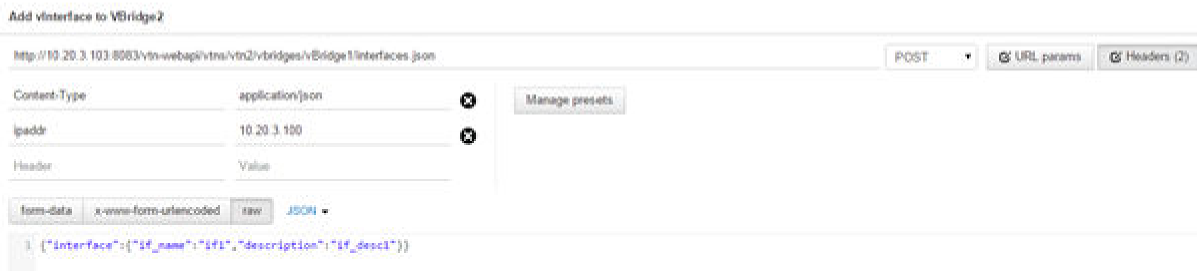
http://10.20.3.103:8083/vtn-webapi/vtns/vtn1/vbridges.json

Content-Type	application/json
ipaddr	10.20.3.100
Header	Value

```
{"vbridge":{"vbr_name":"vBridge1","controller_id":"controller1","domain_id":"(DEFAULT)" }}
```

22 Add vInterface IF1 to vBridge1:

```
curl -v -X POST -H 'content-type: application/json' -H 'username:
admin' -H
 'password: adminpass' -H 'ipaddr:<onecontroller-ip>' -d
'{"interface":
{"if_name": "if1","description": "if_desc1"}}'
http://127.0.0.1:8083/vtn-webapi/vtns/vtn2/vbridges/vBridge1/
interfaces.json
```



Add vInterface to VBridge2

http://10.20.3.103:8083/vtn-webapi/vtns/vtn2/vbridges/vBridge1/interfaces.json

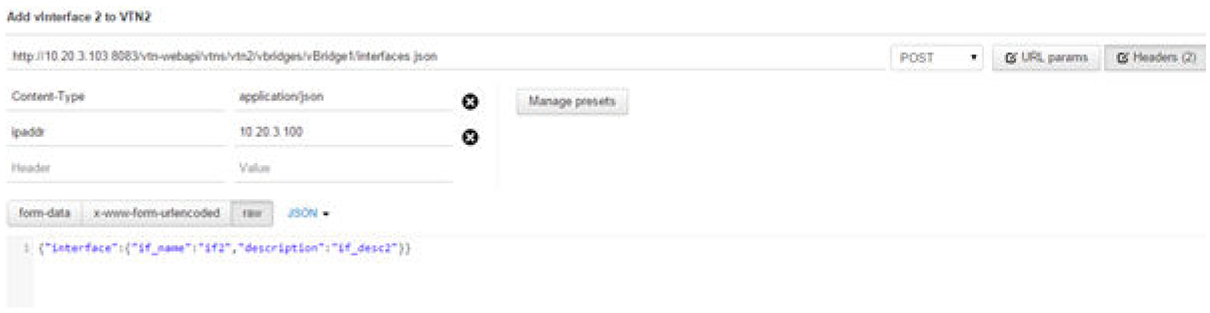
Content-Type	application/json
ipaddr	10.20.3.100
Header	Value

```
{"interface":{"if_name":"if1","description":"if_desc1"}}
```

Figure 19: Add vInterface IF1 to vBridge1

23 Add vInterface IF2 to vBridge1:

```
curl -v -X POST -H 'content-type: application/json' -H 'username:
admin' -H
 'password: adminpass' -H 'ipaddr:<onecontroller-ip>' -d
'{"interface":
{"if_name": "if2","description": "if_desc2"}}'
http://127.0.0.1:8083/vtn-webapi/vtns/vtn2/vbridges/vBridge1/
interfaces.json
```



Add vInterface 2 to VTN2

http://10.20.3.103:8083/vtn-webapi/vtns/vtn2/vbridges/vBridge1/interfaces.json

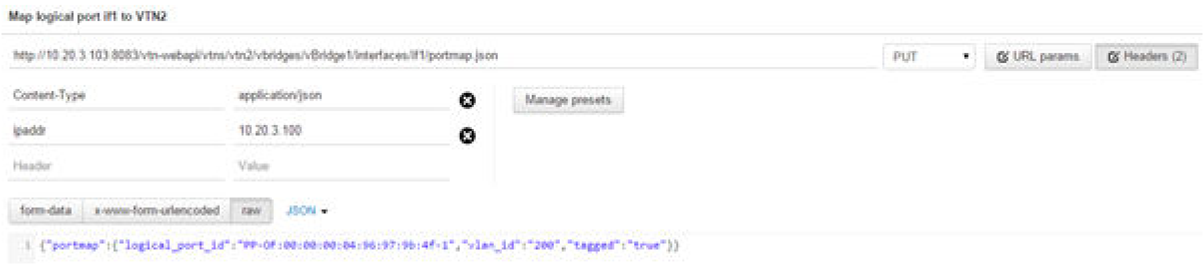
Content-Type	application/json
ipaddr	10.20.3.100
Header	Value

```
{"interface":{"if_name":"if2","description":"if_desc2"}}
```

Figure 20: Add vInterface IF2 to vBridge1

24 Map the logical ports to the interface IF1 (same ports as VTN1)

```
curl -v -X PUT -H 'content-type: application/json' -H
 'username: admin' -H 'password: adminpass' -H 'ipaddr:<onecontroller-
ip>' -d
 '{"portmap":{"logical_port_id": "PP-OF:00:00:<switch-mac-address-
here>-<port id here>",
"vlan_id": "200","tagged": "true"}}'
http://127.0.0.1:8083/vtn-webapi/vtns/vtn2/vbridges/vBridge1/
interfaces/if1/portmap.json
```


Map logical port if1 to VTN2

http://10.20.3.103:8083/vtn-webapi/vtns/vtn2/vbridges/vBridge1/interfaces/if1/portmap.json PUT

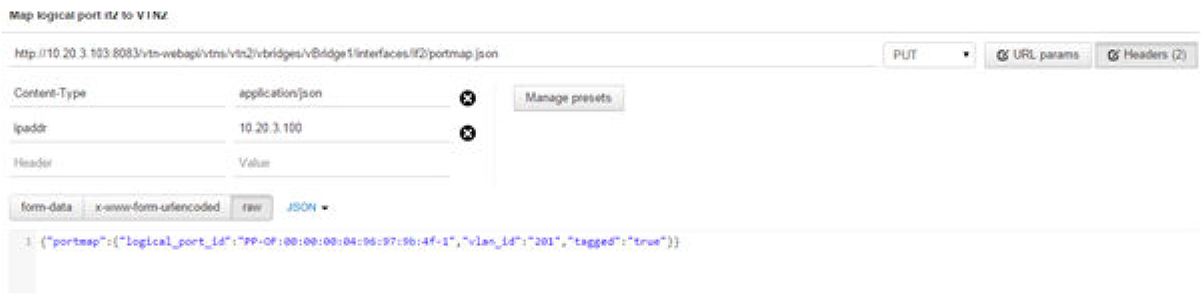
Content-Type	application/json
ipaddr	10.20.3.100

```
{"portmap":{"logical_port_id":"PP-OF:00:00:00:04:96:97:9b:4f-1","vlan_id":"200","tagged":"true"}}
```

Figure 21: Map logical ports to Interface IF1

25 Map the logical ports to the interface IF2 (same ports as VTN1)

```
curl -v -X PUT -H 'content-type: application/json' -H 'username:
admin' -H
 'password: adminpass' -H 'ipaddr:<onecontroller-ip>' -d '{"portmap":
{"logical_port_id":
" PP-OF:00:00:<switch-mac-address-here>-<port id here>","vlan_id":
"201","tagged": "true"}}
' http://127.0.0.1:8083/vtn-webapi/vtns/vtn2/vbridges/vBridge1/
interfaces/if2/portmap.json
```


Map logical port if2 to VTN2

http://10.20.3.103:8083/vtn-webapi/vtns/vtn2/vbridges/vBridge1/interfaces/if2/portmap.json PUT

Content-Type	application/json
ipaddr	10.20.3.100

```
{"portmap":{"logical_port_id":"PP-OF:00:00:00:04:96:97:9b:4f-1","vlan_id":"201","tagged":"true"}}
```

Figure 22: Map logical ports to Interface IF2

IF1 is mapped to port 1 with VLAN 100 on the Summit X440 (OF|02:77) and IF2 is mapped to port 1 with VLAN 101 on a different Summit X440 (OF|ee:7c).

26 Open VM Host1-dsl and open terminal. This host is connected to port 1 of the summit 440 (OF|02:77):

```
sudo ping  200.1.1.101
```

Ping is successful. You should see that ODL has pushed new flows on both the Summit X440 series switches, so that it can strip and add VLAN tags to get the packets to their destination.

```
show openflow flows
```

27 Open VM Host2-dsl and open terminal (ping VM Host1-dsl):

```
sudo ping 200.1.1.100
```

Ping is successful.

28 Ping VM Host3-dsl:

```
sudo ping 200.1.1.200
```

Should not ping as expected as this host belongs to VTN2.

29 Ping VM Host4-dsl:

```
sudo ping 200.1.1.201
```

Should not ping as expected as this host belongs to VTN2.

30 Open VM Host3-dsl and open terminal (ping VM Host4-dsl):

```
sudo ping 200.1.1.201
```

Ping is successful. You should see that ODL has pushed new flows on both the Summit X440 series switches, so that it can strip and add VLAN tags to get the packets to their destination.

31 Refresh OneController GUI.

The four hosts should be visible and connected to the data ports in the Summit X440 series switches.

32 While the pings are going, disable a port link on the Summit switch.

```
telnet 100.1.1.20
disable port 3
```

33 Refresh OneController GUI.

The new topology should be visible. OneController pushes new flows to the switches for the new topology.

34 Enable port 3.

```
OneController discovers topology again, and pushes new flows again.
```

5 Setting Up OpenStack

OpenStack/Virtual Tenant Network Support
OpenStack Controller and Compute VM Requirements
Deploying OpenStack OVA Files
Starting OpenStack
Automatic OpenStack to VTN Mapping
Verifying OpenStack Configuration
Verifying Open vSwitch Configuration
Verifying Web Interface
OpenStack VM to VM Communication Using Manual User Mapping
Understanding Automatic VTN Naming Convention

This section details the installing and configuring of OpenStack. Details about supported and unsupported features are also listed.

OpenStack/Virtual Tenant Network Support

Currently Supported

- Tenant creation
- vBridge creation
- vInterface creation
- Manual ExtremeXOS port/VLAN mapping to vinterface

Conflicting Support

- OpenStack VLAN maps to VTN VLAN maps to vbridge conflicts with VM mapping to vinterface. You cannot have both on the same bridge (causes an error).

Not Supported

- ExtremeXOS working with VXLAN

OpenStack Controller and Compute VM Requirements

Two NICs (one for external communication that has ability to connect to the Internet, and one for internal VM communication). See below for centos:

- ens33 = external nic on each openstack vm (configure IP address or use dhcp)
- ens34 = internal nic on each openstack vm connected to EXOS data ports (no ip address)

Deploying OpenStack OVA Files

Prior to deploying the OpenStack OVA files, you must have a working centOS6.4 VM installed on an ESXi VM server with OneController Virtual Tenant Network (VTN) running. The VM must be able to communicate with repositories for package installs.

To install OpenStack, deploy the following VMs in the same manner as the other previous virtual appliances:

- EXTR-Openstack-Juno-control-compute1-DAKit.ova
- EXTR-Openstack-Juno-compute2-DAKit.ova

Starting OpenStack

To start OpenStack:

1 Set hostname and add hosts of other OpenStack computes/control (important for OpenStack that this is configured):

```
vi /etc/hosts
```

```
[root@junoos-control-compute1 ~]# cat /etc/hosts
127.0.0.1    localhost.localdomain localhost junoos-control-compute1
::1          localhost6.localdomain6 localhost6
100.1.1.201  junoos-compute2
[root@junoos-compute2 devstack]#  cat /etc/hosts
127.0.0.1    localhost.localdomain localhost junos-compute2 junoos-
compute2
::1          localhost6.localdomain6 localhost6
100.1.1.200  junoos-control-compute1
```

2 Copy or edit the appropriate `local.conf` in the `/opt/stack/devstack` folder (`local.conf` appropriate to whether server is intended as compute or control). Fill in appropriate IP addresses and hostname.

```
vi local.conf
```

EXTR-Openstack-Juno-control-compute1-DAKit.ova local.conf file

```
[[local|localrc]]
#LOGFILE=stack.sh.log
#SCREEN_LOGDIR=/opt/stack/data/log
LOG_COLOR=False
OFFLINE=True
#RECLONE=yes
MULTI_HOST=1
disable_service rabbit
enable_service qpid
enable_service nova
enable_service n-api
enable_service n-nova
enable_service n-cpu
enable_service n-cond
enable_service n-novnc
disable_service n-net
enable_service q-agt
enable_service q-svc
enable_service q-dhcp
```

```
disable_service q-l3
enable_service q-meta
enable_service neutron
disable_service tempest
enable_service odl-compute
HOST_NAME=<you-hostname-here>
SERVICE_HOST_NAME=${HOST_NAME}
SERVICE_HOST=<your-ipaddress-here>
Q_HOST=$SERVICE_HOST
HOST_IP=<your-ipaddress-here>
Q_PLUGIN=ml2
#Q_ML2_PLUGIN_MECHANISM_DRIVERS=opendaylight,logger
#NEUTRON_REPO=https://github.com/CiscoSystems/neutron.git
#NEUTRON_BRANCH=odl_ml2
ODL_MGR_IP=<controller-ipaddress-here>
#FLOATING_RANGE=100.1.1.0/24
#PUBLIC_NETWORK_GATEWAY=100.1.1.1
VNCSERVER_PROXYCLIENT_ADDRESS=<your-ipaddress-here>
VNCSERVER_LISTEN=<your-ipaddress-here>
DATABASE_TYPE=mysql
MYSQL_HOST=$SERVICE_HOST
RABBIT_HOST=$SERVICE_HOST
GLANCE_HOSTPORT=$SERVICE_HOST:9292
KEYSTONE_AUTH_HOST=$SERVICE_HOST
KEYSTONE_SERVICE_HOST=$SERVICE_HOST
MYSQL_PASSWORD=mysql
RABBIT_PASSWORD=rabbit
QPID_PASSWORD=rabbit
SERVICE_TOKEN=service
SERVICE_PASSWORD=admin
ADMIN_PASSWORD=admin
[[post-config|/etc/cinder/cinder.conf]]
[DEFAULT]
use_syslog=True
iscsi_target_prefix=iqn.2010-10.org.openstack:
iscsi_ip_address=<your-ipaddress-here>
iscsi_port=3260
[[post-config|/etc/nova/nova.conf]]
[DEFAULT]
vif_plugging_timeout = 600
vif_plugging_is_fatal = False
use_syslog=True
instance_build_timeout = 600
[[post-config|/etc/neutron/plugins/ml2/ml2_conf.ini]]
[ml2]
type_drivers = local
tenant_network_types = local
mechanism_drivers = opendaylight,logger
#[ml2_type_vxlan]
#vni_ranges = 65537:69999
#[ml2_type_gre]
#tunnel_id_ranges = 32769:34000
[ovs]
#local_ip = 172.172.172.2
#tunnel_type = vxlan
#tunnel_bridge = br-tun
#integration_bridge = br-int
#tunnel_id_ranges = 65537:69999
#tenant_network_type = flat
#enable_tunneling = true
[agent]
```

```
root_helper = sudo neutron-rootwrap /etc/neutron/rootwrap.conf
#tunnel_types = vxlan
#vxlan_udp_port = 4789
#l2_population = False
minimize_polling=True
[ml2_odl]
url=http://<controller-ipaddress-here>:8080/controller/nb/v2/neutron
username=admin
password=abc123
```

EXTR-Openstack-Juno-compute2-DAKit.ova local.conf file

```
[[local|localrc]]
#LOGFILE=stack.sh.log
LOG_COLOR=False
#SCREEN_LOGDIR=/opt/stack/data/log
OFFLINE=true
#RECLONE=yes
disable_all_services
enable_service n-api
enable_service n-cpu
enable_service c-vol
enable_service quantum
enable_service n-nova
enable_service qpid
enable_service odl-compute
enable_service n-novnc
enable_service q-agt
HOST_NAME=<your-hostname-here>
HOST_IP=<your-ipaddress-here>
SERVICE_HOST_NAME=<Control-hostname-here>
SERVICE_HOST=<control-ipaddress-here>
VNCSERVER_PROXYCLIENT_ADDRESS=<your-ipaddress-here>
VNCSERVER_LISTEN=<your-ipaddress-here>

Q_PLUGIN=ml2
#Q_ML2_PLUGIN_MECHANISM_DRIVERS=opendaylight,logger
ODL_MGR_IP=<controller-ipaddress-here>
#FLOATING_RANGE=100.2.1.0/24
DATABASE_TYPE=mysql
MYSQL_HOST=$SERVICE_HOST
RABBIT_HOST=$SERVICE_HOST
GLANCE_HOSTPORT=$SERVICE_HOST:9292
KEYSTONE_AUTH_HOST=$SERVICE_HOST
KEYSTONE_SERVICE_HOST=$SERVICE_HOST
MYSQL_PASSWORD=mysql
RABBIT_PASSWORD=rabbit
QPID_PASSWORD=rabbit
SERVICE_TOKEN=service
SERVICE_PASSWORD=admin
ADMIN_PASSWORD=admin
[[post-config|/etc/cinder/cinder.conf]]
[DEFAULT]
use_syslog=True
iscsi_target_prefix=iqn.2010-10.org.openstack:
iscsi_ip_address=<your-ipaddress-here>
iscsi_port=3260
my_ip = <your-ipaddress-here>
[[post-config|/etc/nova/nova.conf]]
[DEFAULT]
vif_plugging_timeout = 600
```

```
vif_plugging_is_fatal = False
use_syslog=True
instance_build_timeout = 600

[[post-config|/etc/neutron/plugins/ml2/ml2_conf.ini]]
[ml2]
type_drivers = local
tenant_network_types = local
mechanism_drivers = opendaylight,logger
#[ml2_type_vxlan]
#vni_ranges = 65537:69999
#[ml2_type_gre]
#tunnel_id_ranges = 32769:34000
[ovs]
#local_ip =
#tunnel_type = vxlan
#tunnel_bridge = br-tun
#integration_bridge = br-int
#tunnel_id_ranges = 65537:69999
#tenant_network_type = flat
#enable_tunneling = true
[agent]
root_helper = sudo neutron-rootwrap /etc/neutron/rootwrap.conf
#tunnel_types = vxlan
#vxlan_udp_port = 4789
#l2_population = False
minimize_polling=True
[ml2_odl]
url=http:// <controller-ipaddress-here>:8080/controller/nb/v2/neutron
username=admin
password=abc123
```

3 After deploying `Junoos-control-compute1.ova`, power on of the Openstack Control Compute1 using the ESXi server. Start OpenStack on the OpenStack Control VM (Openstack-Juno-controlcompute1.ova). Starting OpenStack takes around 10 minutes.

4 Log on as root (user name = root, password = extreme).

```
cd /root/openstack_scripts
./stack-openstack
```

5 After deploying (Junoos-compute2.ova), power on of the Openstack Compute 2 using the ESXi server. After the Openstack Control VM (Junoos-control-compute1.ova) is completely finished the stack process, start OpenStack on OpenStack Compute VM (Junoos-compute2.ova). Starting OpenStack takes around 5 minutes.

6 Log on as root (user name = root, password = extreme).

```
cd /root/openstack_scripts
./stack-openstack
```

Automatic OpenStack to VTN Mapping

- Project = tenant gets automatically created
- Network under Project = Vbridge under the tenant automatically created
- VM in Network under Project = vInterface in vBridge under the tenant automatically created

Verifying OpenStack Configuration

To verify the OpenStack configuration:

1 Log on using command line from the Devstack directory:

```
[root@junoos-control-compute1 ~]# cd /opt/stack/devstack/
```

2 Enable OpenStack command line interface:

```
[root@junoos-control-compute1 devstack]# . ./openrc admin admin
```

3 Verify running services on each server (dependent on synced clocks):

```
[root@junoos-control-compute1 devstack]# nova-manage service list
Binary           Host
Zone             Status     State Updated_At
nova-conductor   junoos-control-compute1
internal         enabled    :-)   2015-02-26 16:33:41
nova-cert        junoos-control-compute1
internal         enabled    :-)   2015-02-26 16:33:34
nova-scheduler   junoos-control-compute1
internal         enabled    :-)   2015-02-26 16:33:37
nova-consoleauth junoos-control-compute1
internal         enabled    :-)   2015-02-26 16:33:36
nova-compute     junoos-control-compute1
nova             enabled    :-)   2015-02-26 16:33:33
nova-compute     junoos-compute2
nova             enabled    :-)   2015-02-26 16:33:37
[root@junoos-compute2 devstack]# nova-manage service list
Binary           Host
Zone             Status     State Updated_At
nova-conductor   junoos-control-compute1
internal         enabled    :-)   2015-02-26 16:35:11
nova-cert        junoos-control-compute1
internal         enabled    :-)   2015-02-26 16:35:14
nova-scheduler   junoos-control-compute1
internal         enabled    :-)   2015-02-26 16:35:17
nova-consoleauth junoos-control-compute1
internal         enabled    :-)   2015-02-26 16:35:16
nova-compute     junoos-control-compute1
nova             enabled    :-)   2015-02-26 16:35:13
nova-compute     junoos-compute2
nova             enabled    :-)   2015-02-26 16:35:17
```

4 Verifying hypervisors:

```
[root@junoos-control-compute1 devstack]# nova hypervisor-list
+----+-------------------------+
| ID | Hypervisor hostname     |
+----+-------------------------+
| 1  | junoos-control-compute1 |
| 2  | junoos-compute2         |
+----+-------------------------+
[root@junoos-compute2 devstack]# nova hypervisor-list
+----+-------------------------+
| ID | Hypervisor hostname     |
+----+-------------------------+
| 1  | junoos-control-compute1 |
| 2  | junoos-compute2         |
+----+-------------------------+
```

Verifying Open vSwitch Configuration

In order for the Compute VMs on one server to talk to the Compute VMs of another server, a physical NIC must be added to bridge "br-int". By default, ens34 is added automatically to bridge “br-int” after the stack-openstack command execution. This bridge is where all VMs virtual NICs are created upon starting up a VM.

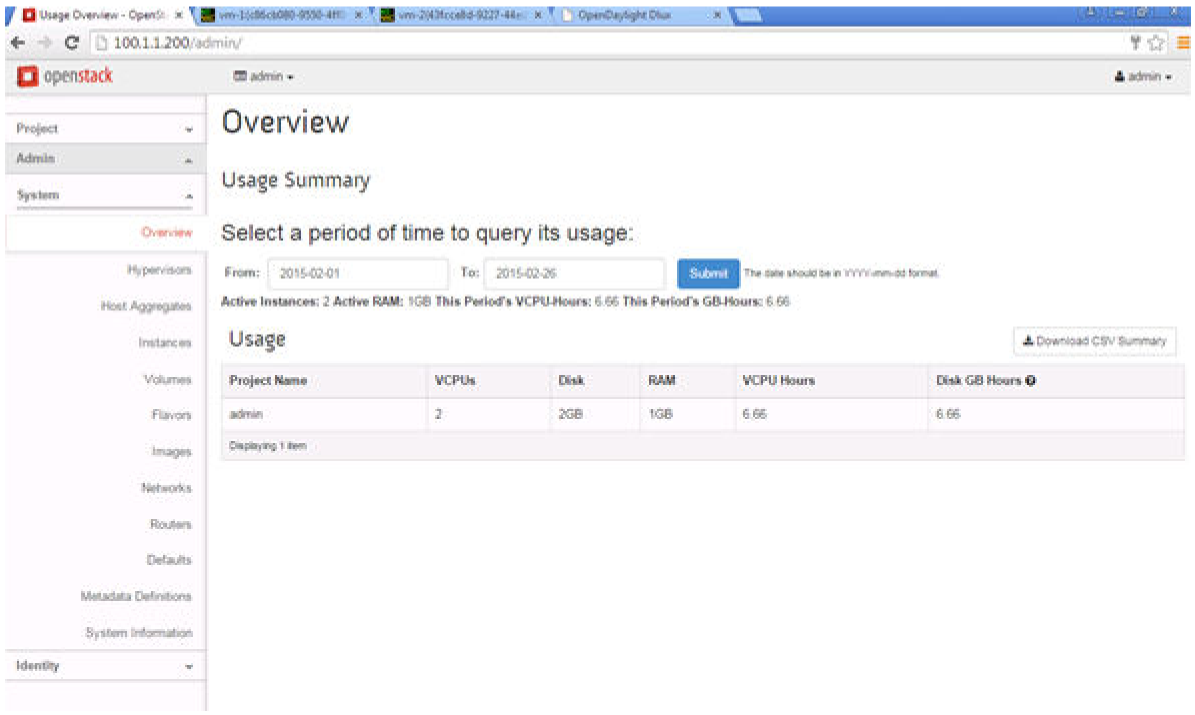
To verifying Open vSwitch configuration:

```
[root@junoos-control-compute1 ~]# sudo ovs-vsctl show
cdf580b3-ac43-41f8-aeda-d6c701b15535
    Bridge br-tun
        Port patch-int
            Interface patch-int
                type: patch
                options: {peer=patch-tun}
        Port br-tun
            Interface br-tun
                type: internal
    Bridge br-int
        fail_mode: secure
        Port "ens34"
            tag: 1
            Interface "ens34"
        Port patch-tun
            Interface patch-tun
                type: patch
                options: {peer=patch-int}
        Port br-int
            Interface br-int
                type: internal
    ovs_version: "2.1.3"
[root@junoos-compute2 devstack]# sudo ovs-vsctl show
cf3e3a09-4fb0-4ee5-a9bb-155d275f7f43
Bridge br-tun
    Port br-
tun
Interface br-tun
                type: internal
        Port patch-int
            Interface patch-int
                type: patch
                options: {peer=patch-tun}
    Bridge br-int
        fail_mode: secure
        Port patch-tun
            Interface patch-tun
                type: patch
                options: {peer=patch-int}
        Port "ens34"
            tag: 1
            Interface "ens34"
        Port br-int
            Interface br-int
                type: internal
    ovs_version: "2.1.3"
```

Verifying Web Interface

> **Note**
> If web interface is not accessible, turn off firewall on the server.

In a web browser, log on to the web interface (http://<openstack-control-ipaddress-here>:80) with credentials admin/admin.



Overview

Usage Summary

Select a period of time to query its usage:

From: 2015-02-01 To: 2015-02-26 The date should be in YYYY-mm-dd format.

Active Instances: 2 **Active RAM:** 1GB **This Period's VCPU-Hours:** 6.66 **This Period's GB-Hours:** 6.66

Usage

Project Name	VCPUs	Disk	RAM	VCPU Hours	Disk GB Hours
admin	2	2GB	1GB	6.66	6.66

Displaying 1 item

Figure 23: OpenStack Web Interface

OpenStack VM to VM Communication Using Manual User Mapping

Ping VM created on Compute from VM created on Controller.

Prerequisites:

- Have OneController open in web browser.
- Have SSH Karaf command line interface session open. This is needed to get the VTN names created by OpenStack so that you can manually map port/VLANs of Summit 440 series switches to appropriate vInterface of VM.

```
admin@OC.extremenetworks.com:~$ ssh karaf@127.0.0.1 -p 8101
Password authentication
Password: (karaf)
opendaylight-user@root>log:tail |grep vtn
```

- Have an SSH session of VTN coordinator VM to input cURL commands.
- Have proper cURL commands ready (see cURL Commands on page 33)

1 Project > Network > Network Topology.
2 In upper-right corner, click Create Network.
3 Create the network, and then add a subnet.
4 Click Project > Compute > Instance.
5 Create VM1:
    a Use Flavor == m1.tiny and image == cirros.
    b Click the Network tab, and then use the network created in step 3 on page 33.
6 Verify the VTN names in SSH session.
7 Map vInterface of VM to VLAN/port that VM is directly connected to on the Summit X440 switch.
8 Create VM2:
    a Use Flavor == m1.tiny and image == cirros.
    b Click the Network tab, and then use the network created in step 3 on page 33.
9 Verify the VTN names in SSH session.
10 Map vInterface of VM to VLAN/port that VM is directly connected to on the Summit X440 switch using the cURL commands (see cURL Commands on page 33).
11 Verify that a VM resides on each server (Junoos-control-compute1.ova and Junoos-compute2).
12 Open the VM console in a new browser tab.
13 After the port/VLAN/VTN mapping has occured, the VMs can obtina a DHCP address. On each VM console execute:

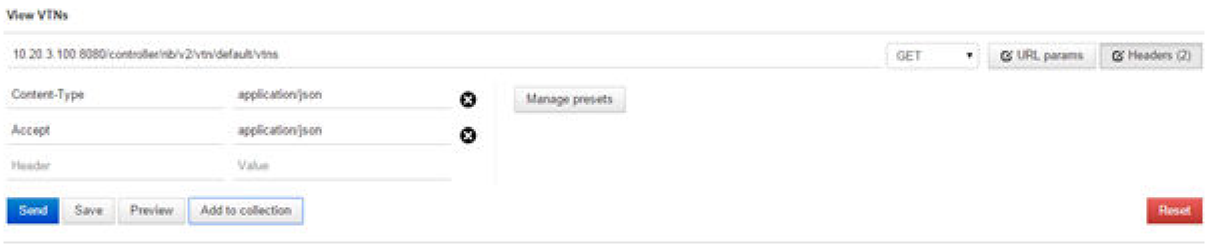
```
sudo ifup eth0
sudo ifconfig
```

14 Ping VM1 to VM2.
15 Ping VM2 to VM1.

cURL Commands

- View VTNs:

```
curl --user "admin":"abc123" -H "Accept: application/json" -H
 "Content-type: application/json" -X GET
http://<OneController-ip>:8080/controller/nb/v2/vtn/default/vtns |
python -mjson.tool
```


View VTNs

10.20.3.100:8080/controller/nb/v2/vtn/default/vtns

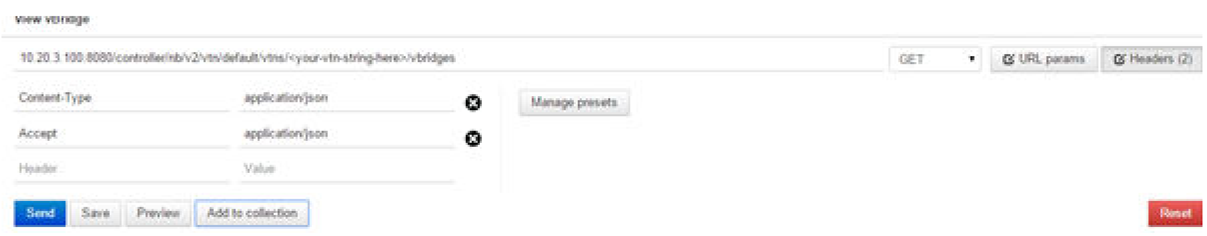
GET

Content-Type	application/json
Accept	application/json
Header	Value

Figure 24: Viewing VTNs

- View vBridge:

```
curl --user "admin":"abc123" -H "Accept: application/json" -H
 "Content-type: application/json" -X GET
http:// <OneController-ip>:8080/controller/nb/v2/vtn/default/vtns/
<your-vtn-string-here>/vbridges | python -mjson.tool
```


View vBridge

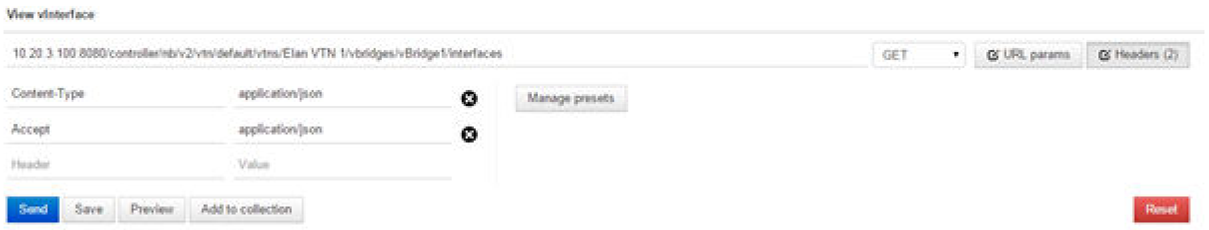
10.20.3.100:8080/controller/nb/v2/vtn/default/vtns/<your-vtn-string-here>/vbridges

Content-Type	application/json
Accept	application/json
Header	Value

Figure 25: Viewing vBridge

- View vInterface:

```
curl --user "admin":"abc123" -H "Accept: application/json" -H
 "Content-type: application/json" -X GET
http:// <OneController-ip>:8080/controller/nb/v2/vtn/default/vtns/
<your-vtn-string-here>/vbridges/<you-bridge-name-here>/interfaces |
python -mjson.tool
```


View vInterface

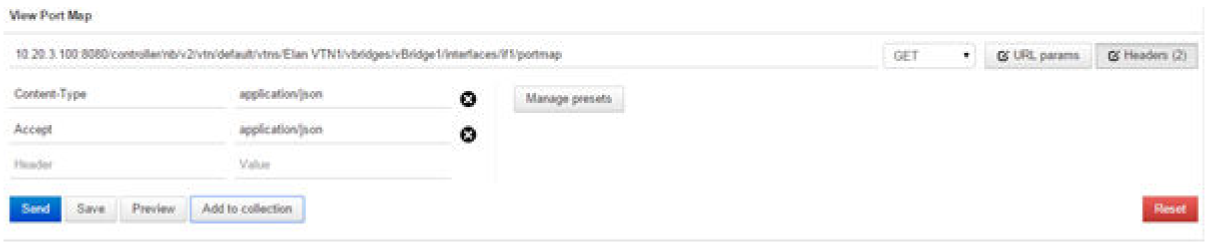
10.20.3.100:8080/controller/nb/v2/vtn/default/vtns/Elan VTN 1/vbridges/vBridge1/interfaces

Content-Type	application/json
Accept	application/json
Header	Value

Figure 26: Viewing vInterface

- View port maps:

```
curl --user "admin":"abc123" -H "Accept: application/json" -H
 "Content-type: application/json" -X GET
http:// <OneController-ip>:8080/controller/nb/v2/vtn/default/vtns/
<your- vtn -string-here>/vbridges/<you-bridge-name-here>/interfaces/
<you-vinterface1-name-here>/portmap | python -mjson.tool
```

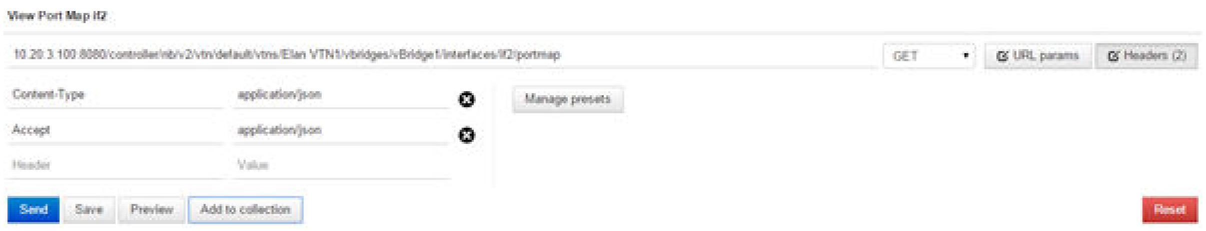

View Port Map

10.20.3.100:8080/controller/nb/v2/vtn/default/vtns/Elan VTN1/vbridges/vBridge1/interfaces/if1/portmap

Content-Type	application/json
Accept	application/json
Header	Value

Figure 27: Viewing Port Map IF1

```
curl --user "admin":"abc123" -H "Accept: application/json" -H
 "Content-type: application/json" -X GET
http:// <OneController-ip>:8080/controller/nb/v2/vtn/default/vtns/
<your- vtn -string-here>/vbridges/<you-bridge-name-here>/interfaces/
<you-vinterface2-name-here>/portmap | python -mjson.tool
```


View Port Map If2

10.20.3.100:8080/controller/nb/v2/vtn/default/vtns/Elan VTN1/vbridges/vBridge1/interfaces/If2/portmap

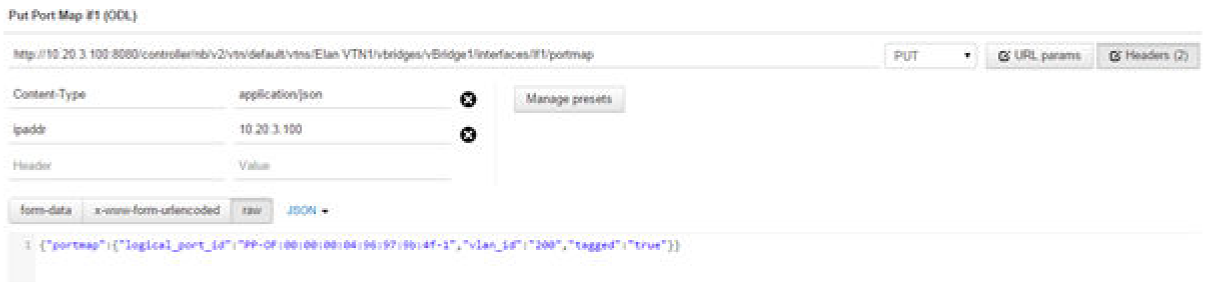
GET

Content-Type	application/json
Accept	application/json

Figure 28: Viewing Port Map IF2

- Put port map:

```
curl --user "admin":"abc123" -H "Accept: application/json" -H
 "Content-type: application/json" -X PUT
http:// <OneController-ip>:8080/controller/nb/v2/vtn/default/vtns/
<your- vtn -string-here>/vbridges/<you-bridge-name-here>/interfaces/
<you-vinterface1-name-here>/portmap -d '{"vlan": "<vlan-number>",
"node": {"type": "OF", "id": "00:00:<switch-mac-address-here>"},
"port":
{"name": "<port-number>"}}'
```


Put Port Map If1 (ODL)

http://10.20.3.100:8080/controller/nb/v2/vtn/default/vtns/Elan VTN1/vbridges/vBridge1/interfaces/If1/portmap

PUT

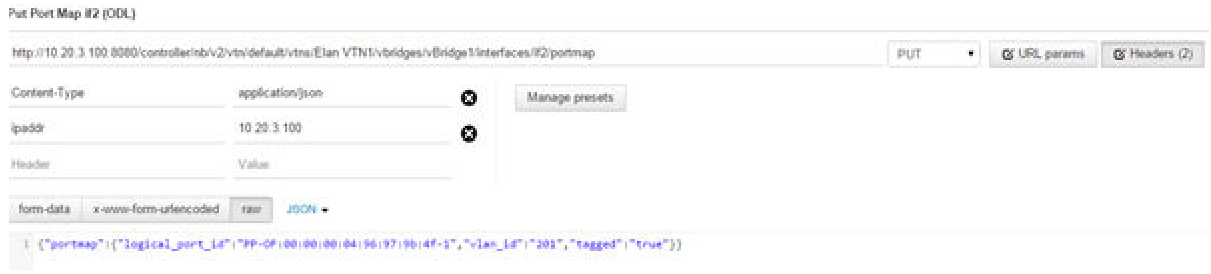
Content-Type	application/json
ipaddr	10.20.3.100

raw JSON

```
{"portmap":{"logical_port_id":"PP-OF:00:00:00:04:96:97:9b:4f-1","vlan_id":"200","tagged":"true"}}
```

Figure 29: Put Port Map IF1

```
curl --user "admin":"abc123" -H "Accept: application/json" -H
 "Content-type: application/json" -X PUT
http:// <OneController-ip>:8080/controller/nb/v2/vtn/default/vtns/
<your- vtn -string-here>/vbridges/<you-bridge-name-here>/interfaces/
<you-vinterface2-name-here>/portmap -d '{"vlan": "<vlan-number>",
"node": {"type": "OF", "id": "00:00: <switch-mac-address-here>"},
"port": {"name": "<port-number>"}}'
```


Put Port Map If2 (ODL)

http://10.20.3.100:8080/controller/nb/v2/vtn/default/vtns/Elan VTN1/vbridges/vBridge1/interfaces/If2/portmap

PUT

Content-Type	application/json
ipaddr	10.20.3.100

raw JSON

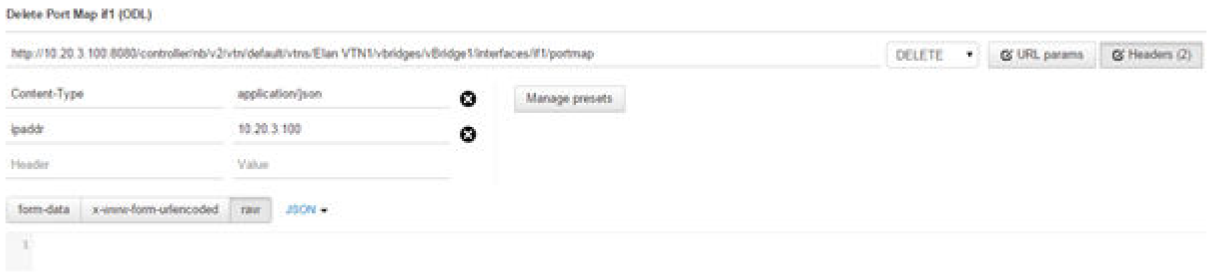
```
{"portmap":{"logical_port_id":"PP-OF:00:00:00:04:96:97:9b:4f-1","vlan_id":"201","tagged":"true"}}
```

Figure 30: Put Port Map IF2

- Delete port map:

```
curl --user "admin":"abc123" -H "Accept: application/json" -H
 "Content-type: application/json" -X DELETE
http:// <OneController-ip>:8080/controller/nb/v2/vtn/default/vtns/
```

```
<your- vtn -string-here>/vbridges/<you-bridge-name-here>/interfaces/
<you-vinterface1-name-here>/portmap
```


Delete Port Map #1 (ODL)

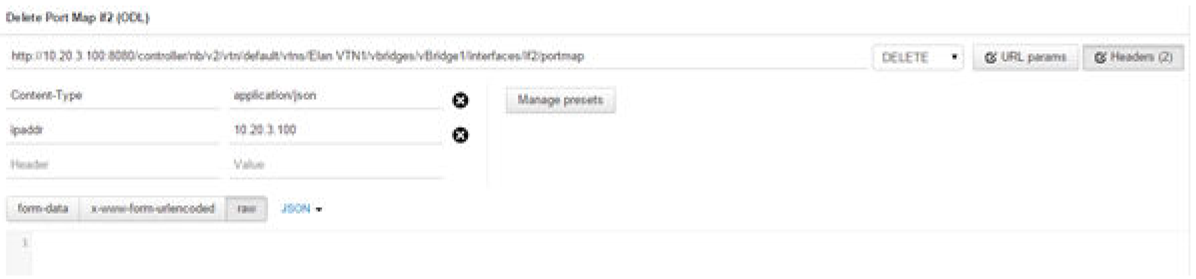
http://10.20.3.100:8080/controller/nb/v2/vtn/default/vtns/Elan VTN1/vbridges/vBridge1/interfaces/if1/portmap

DELETE

Content-Type	application/json
ipaddr	10.20.3.100

Figure 31: Delete Port Map IF1

```
curl --user "admin":"abc123" -H "Accept: application/json" -H
 "Content-type: application/json" -X DELETE
http:// <OneController-ip>:8080/controller/nb/v2/vtn/default/vtns/
<your- vtn -string-here>/vbridges/<you-bridge-name-here>/interfaces/
<you-vinterface2-name-here>/portmap
```


Delete Port Map #2 (ODL)

http://10.20.3.100:8080/controller/nb/v2/vtn/default/vtns/Elan VTN1/vbridges/vBridge1/interfaces/if2/portmap

DELETE

Content-Type	application/json
ipaddr	10.20.3.100

Figure 32: Delete Port Map IF2

Understanding Automatic VTN Naming Convention

2014-06-04 12:23:38.397 GMT-05:00 [VTN Task Thread: default] INFO o.o.v.m.internal.VTNManagerImpl - default: 0c40c6fffb269a2b6dacd880a0686a9.0446c92c6b5219d96afbdd03c9cc142.37e420c4c3d4c80ae4e13d72d1694d0: Bridge interface added: VInterface[name=37e420c4c3d4c80ae4e13d72d1694d0,desc=,enabled,state=UNKNOWN,entityState=UNKNOWN]

Tenant name: 0c40c6fffb269a2b6dacd880a0686a9

VBridge Name: 0446c92c6b5219d96afbdd03c9cc142

Name: 37e420c4c3d4c80ae4e13d72d1694d0

6 Setting Up Hyperglance

Installing Hyperglance
Using Hyperglance

Installing Hyperglance

This section details the installation and configuration settings required to successfully deploy Hyperglance. You must have a working centOS6.4 VM installed on an ESXi VM server with OneController virtual tenant network (VTN) running.

Hyperglance is based off a server/client architecture with the server communicating directly with the ODL controller and the client communicating directly with the Hyperglance server. Two interfaces can be used if desired. You can configure an “inside” static interface to interconnect to the OneController and an “outside” interface interconnecting the Hyperglance client application. Hyperglance communicates with OneController to populate its tables.

Installing the Hyperglance Virtual Machine Server

You need to download the Hyperglance installation software to the ESXi server prior to installation. Go to www.real-status.com, and click Download for Free.

To install the Hyperglance server:

1. From the vSphere client, click File > Deploy OVF Template.
2. Go to the directory where the downloaded Hyperglance server VM resides.
3. Click Next through the OVF Template Details.
4. At the Name and Location section, name the VM.
5. Use the default settings in Disk Format.
6. Click Next to go to the Network Mapping section. Under Destination Networks, select the destination your sever interface will reside on (eth0 configured for DHCP).
7. Click Next to finish the configuration. Importing may take several minutes to finish.
8. Start the Hyperglance VM from the vSphere client by right-clicking, selecting power, power on, or by selecting the VM, and then clicking the green arrow icon on the vSphere client toolbar.
9. Log on to the server console.

   Logon = hyperglance/hyperglance

   root = sudo su/password = hyperglance

   Manual config = /etc/sysconfig/network-scripts/ifcfg-eth0

10 Enter the following command to see what IP address the server retained: `<ifconfig>` as eth0 is configured for DHCP initially. If a static is needed the following steps can be used for manual configuration.

11 If a second interface is needed for your configuration:

Note

- Manual configuration of the interface(s) can be done by vi of `/etc/sysconfig/network-scripts/ifcfg-eth0`.
- If an additional interface is required user can cp ifcfg-eth0 to cfg-eth1. DHCP or static can be configured. A mapping on the ESXi server is also required in order for eth1 to become functional.

a Select Hyperglance Server VM through the vSphere client.

b In the right pane click Edit virtual machine settings, and then Add a Network adapter.

c Select the correct adapter type E1000 and configured Network connection for that interface. You may need to reboot the VM.

d If problems occur,enter the following command `<rm /etc/udev/rules.d/70-persistent-net.rules>`, and then reboot.

12

Configuring the Hyperglance VM Server

To configure the Hyperglance VM server:

1 Enter the Hyperglance server VM IP address in a web browser, and then click Proceed Anyway for certificate authentication.

Logon = admin/admin

2 In Collectors Setup there are several controller types you can point the Hyperglance server to. The following are two shown as examples and cannot be configured to run simultaneously:

OpenStack:

1 Account Alias can be an arbitrary name.

2 Keystone URL is at the end of the "stacking" output on the OpenStack control. This is usually the IP address of the interface on the control plane followed by port 5000/v2.0 (for example: http://100.1.1.18:5000/v2.0/).

3 Username/password is admin/admin.

4 Click submit.

OneController:

1 OneController URL is the IP address of the controller's UI. (for example; http://100.1.1.24:8080).

2 Username/password is admin/abc123.

3 Click submit.

4 To check connectivity, click Validate.

Installing the Hyperglance Client from the Hyperglance Server

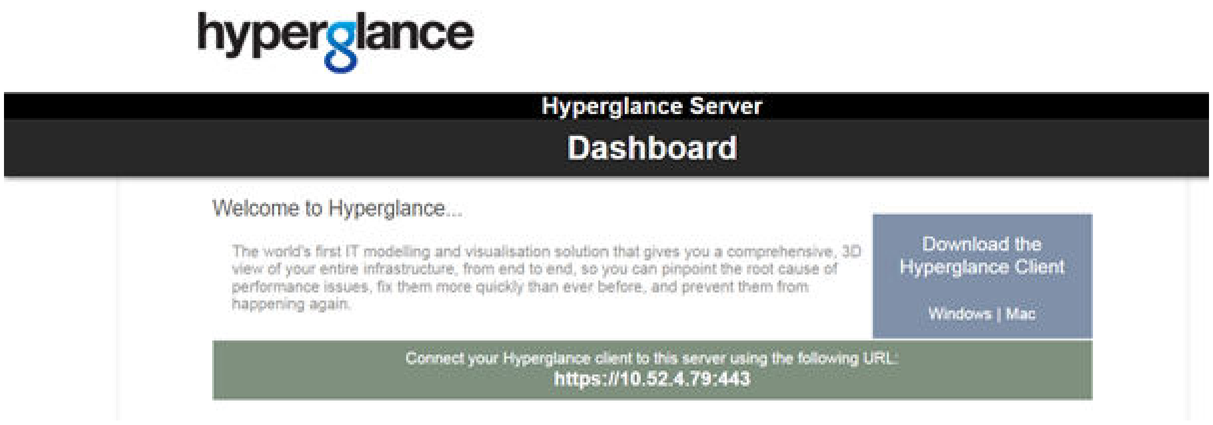
To install the Hyperglance client from the Hyperglance server:

1 In a browser enter the Hyperglance server VM IP address, and select Proceed Anyway for certificate authentication.

  Logon = admin/admin.

2 On the upper-right of the Hyperglance Server dashboard, select the appropriate operating system for your client install within the (Download the Hyperglance Client) box.

3 Note the IP address and port to connect your client to the server when the download finishes on the Hyperglance Server Dashboard (https://10.52.4.254:443). (http://10.52.4.254 also works from the client.)


hyperglance

Hyperglance Server

Dashboard

Welcome to Hyperglance...

The world's first IT modelling and visualisation solution that gives you a comprehensive, 3D view of your entire infrastructure, from end to end, so you can pinpoint the root cause of performance issues, fix them more quickly than ever before, and prevent them from happening again.

Download the Hyperglance Client

Windows | Mac

Connect your Hyperglance client to this server using the following URL:
https://10.52.4.79:443

Figure 33: Hyperglance Server Dashboard

4 Run the .exe.

5 When installation finishes, start the Hyperglance client from the shortcut or the Start menu. This may take some time.

  A dialog box may appear asking to enter the Hyperglance server address. Either address example given above will work. There is a console pane on the right side of the client dashboard that shows the status.

6 To connect to a different server, double-click the server URL at the bottom of dashboard border, or click the Hyperglance icon in the lower left, and then click File > Connect to server.

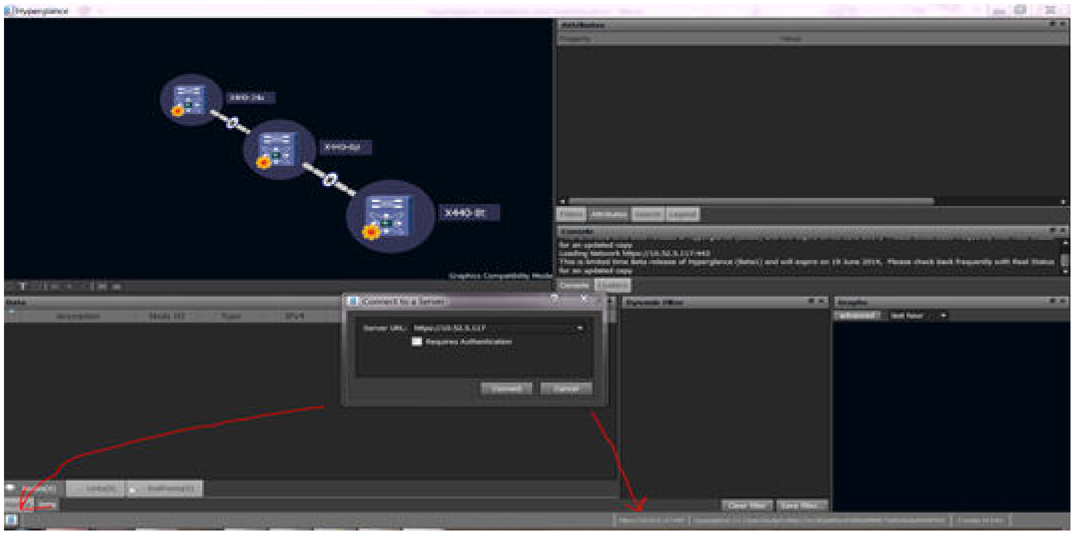


Figure 34: Hyperglance—Connecting to a Different Server

Using Hyperglance

This section details how to successfully navigate Hyperglance. There are two requirements:

- A working centOS6.4 VM installed on an ESXi VM server running OneController.
- Hyperglance server installed and running with two interfaces configured (inside/outside) (see Installing Hyperglance on page 37). Hyperglance is based on a server/client architecture with the server communicating directly with the OneController.

Using Hyperglance Graphics Compatibility Mode

You can re-adjust the discovered network image by right-clicking anywhere in upper, left panel and dragging the pointer to the desired position.

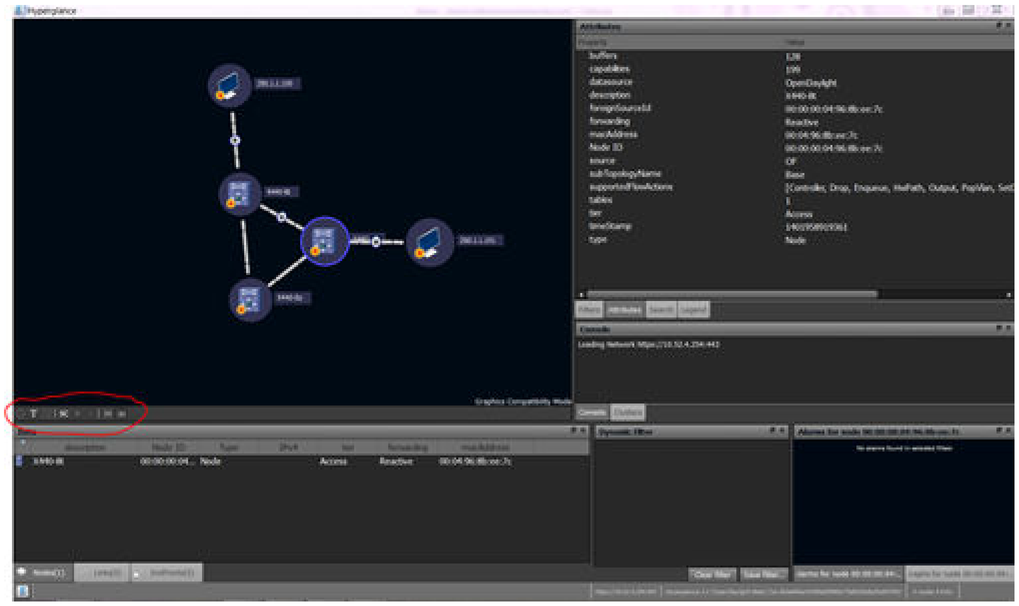
Use the middle button on your mouse to zoom in or out on an image.

You can select a switch in network image to show details under the Data and Attributes panels (lower left and upper right, respectively).

The area circled in red in Figure 35: Hyperglance on page 41 contains the following icons, in order:

- Center (centers network in view)
- Node labels
- Full screen
- Show connected mode
- Add and subtract connected nodes
- 2-D and 3-D graphics modes

By double-clicking a switch, a 3-D cube envelops the switch, and the network is centered with respect to that switch. To undo, click the center icon again.



Property	Value
buffers	128
capabilities	199
datasource	OpenDaylight
description	X440-8t
foreignSourceId	00:00:00:04:96:8b:ee:7c
forwarding	Reactive
macAddress	00:04:96:8b:ee:7c
Node ID	00:00:00:04:96:8b:ee:7c
source	OF
subTopologyName	Base
supportedFlowActions	[Controller, Drop, Enqueue, HwPath, Output, PopVlan, Set
tables	1
tier	Access
timeStamp	1401958919361
type	Node

Loading Network https://10.52.4.254:443

description	Node ID	Type	IPv4	tier	forwarding	macAddress
X440-8t	00:00:00:04...	Node		Access	Reactive	00:04:96:8b:ee:7c

Figure 35: Hyperglance

Right-clicking a selected switch, accesses a menu. You can also access this menu by right-clicking the selected switch on the Data Panel (lower left).

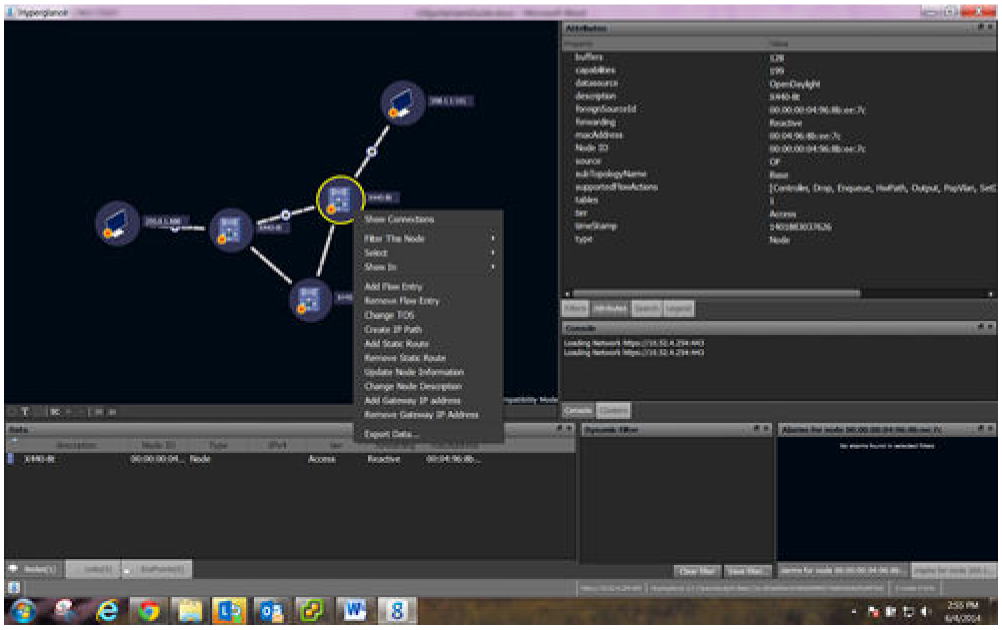
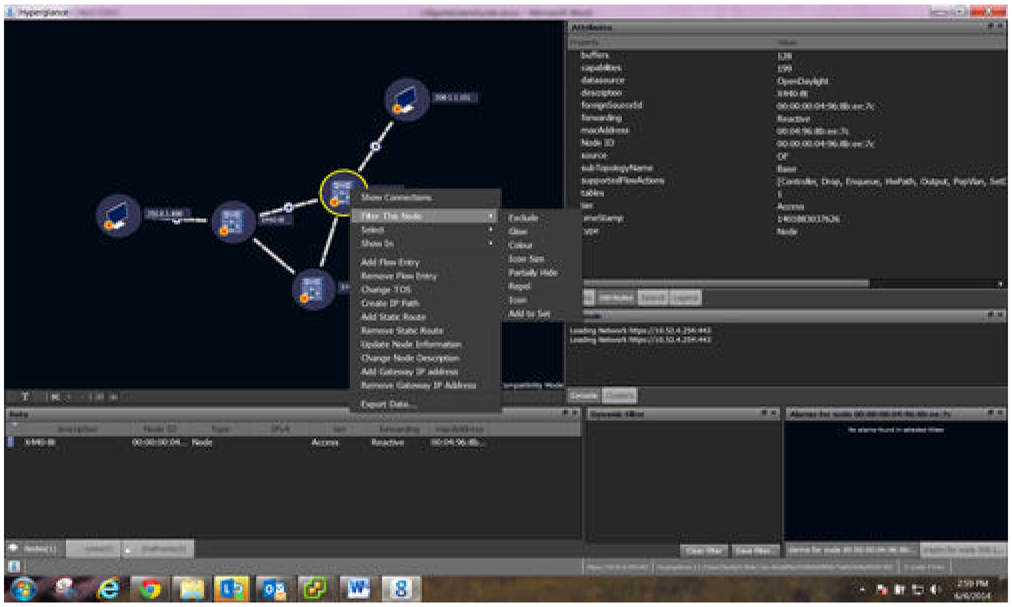


Figure 36: Hyperglance

The menu includes every feature allowed in OneController with the addition of filters, select connected end points, and show in graph form. Hyperglance also allows you to export the switches’ attribute information to a file.



Attributes

Property	Value
buffers	128
capabilities	199
datasource	OpenDaylight
description	X440-8t
foreignSourceId	00:00:00:04:96:8b:ee:7c
forwarding	Reactive
macAddress	00:04:96:8b:ee:7c
Node ID	00:00:00:04:96:8b:ee:7c
source	OF
subTopologyName	Base
supportedFlowActions	[Controller, Drop, Enqueue, HwPath, Output, PopVlan, SetC
tables	1
tier	Access
timeStamp	1401883037626
type	Node

```
Loading Network https://10.52.4.254:443
Loading Network https://10.52.4.254:443
```

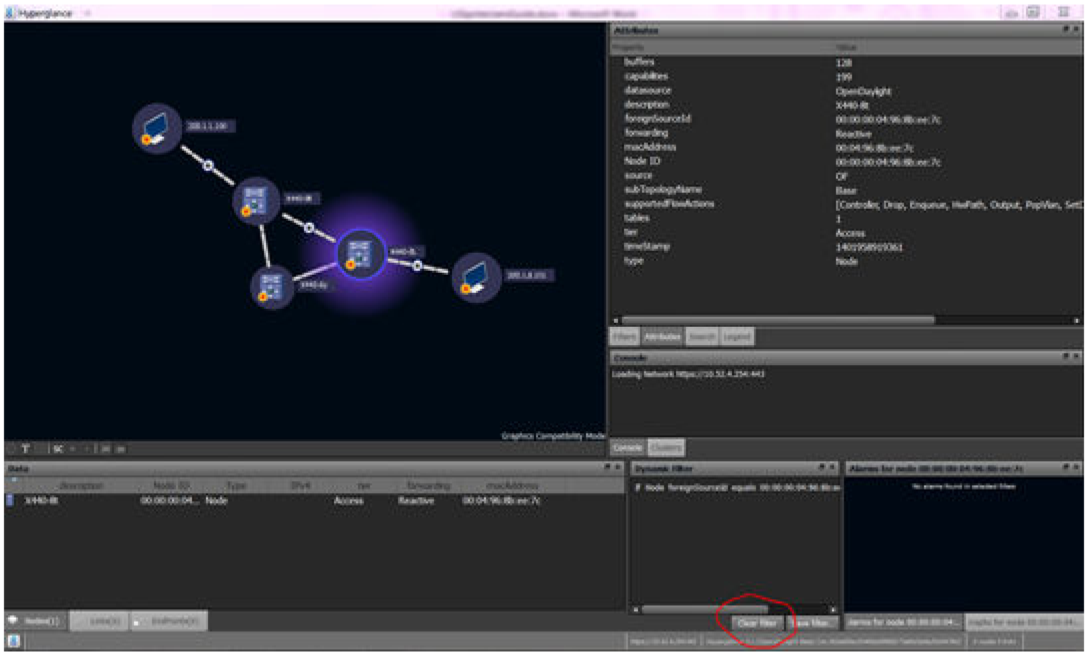
Data

description	Node ID	Type	IPv4	tier	forwarding	macAddress
X440-8t	00:00:00:04...	Node		Access	Reactive	00:04:96:8b...

No alarms found in selected filter.

Figure 37: Hyperglance

Figure 38: Hyperglance on page 42 shows the selected glow filter, which is helpful with large networks containing hundreds of switches. You can clear filters by clicking the Clear Filter button on the lower middle panel.



Attributes

Property	Value
buffers	128
capabilities	199
datasource	OpenDaylight
description	X440-8t
foreignSourceId	00:00:00:04:96:8b:ee:7c
forwarding	Reactive
macAddress	00:04:96:8b:ee:7c
Node ID	00:00:00:04:96:8b:ee:7c
source	OF
subTopologyName	Base
supportedFlowActions	[Controller, Drop, Enqueue, HwPath, Output, PopVlan, SetC
tables	1
tier	Access
timeStamp	1401958919361
type	Node

```
Loading Network https://10.52.4.254:443
```

Data

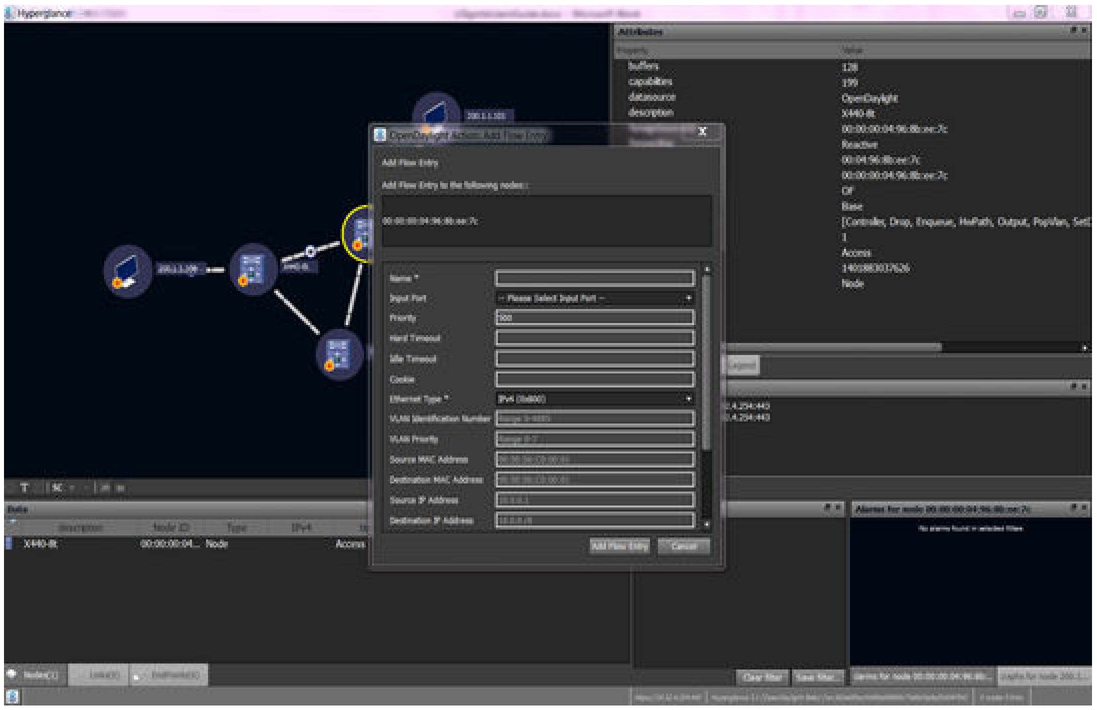
description	Node ID	Type	IPv4	tier	forwarding	macAddress
X440-8t	00:00:00:04...	Node		Access	Reactive	00:04:96:8b:ee:7c

Node foreignSourceId equals 00:00:00:04:96:8b:ee

No alarms found in selected filter.

Figure 38: Hyperglance

You can also add or remove flows using the menu.



Add Flow Entry

Add Flow Entry to the following nodes :

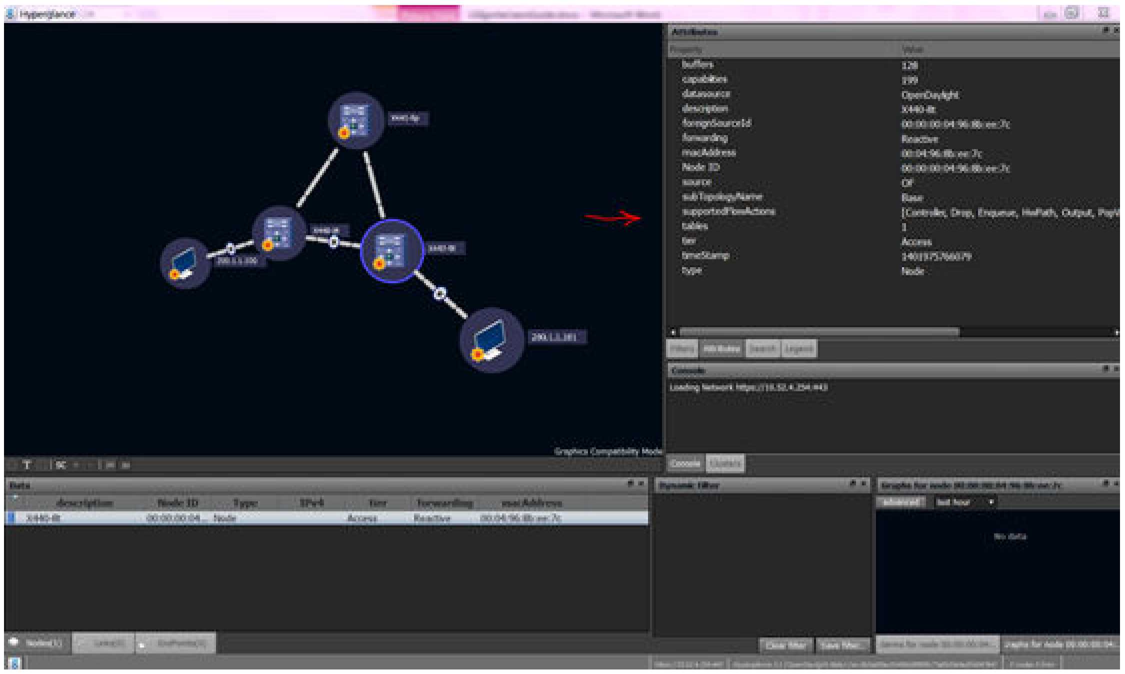
00:00:00:04:96:8b:ee:7c

Name *	
Input Port	-- Please Select Input Port --
Priority	500
Hard Timeout	
Idle Timeout	
Cookie	
Ethernet Type *	IPv4 (0x800)
VLAN Identification Number	Range 0-4095
VLAN Priority	Range 0-7
Source MAC Address	00:00:00:C0:00:00
Destination MAC Address	00:00:00:C0:00:00
Source IP Address	10.0.0.1
Destination IP Address	10.0.0.8

Add Flow Entry · Cancel

Figure 39: Hyperglance

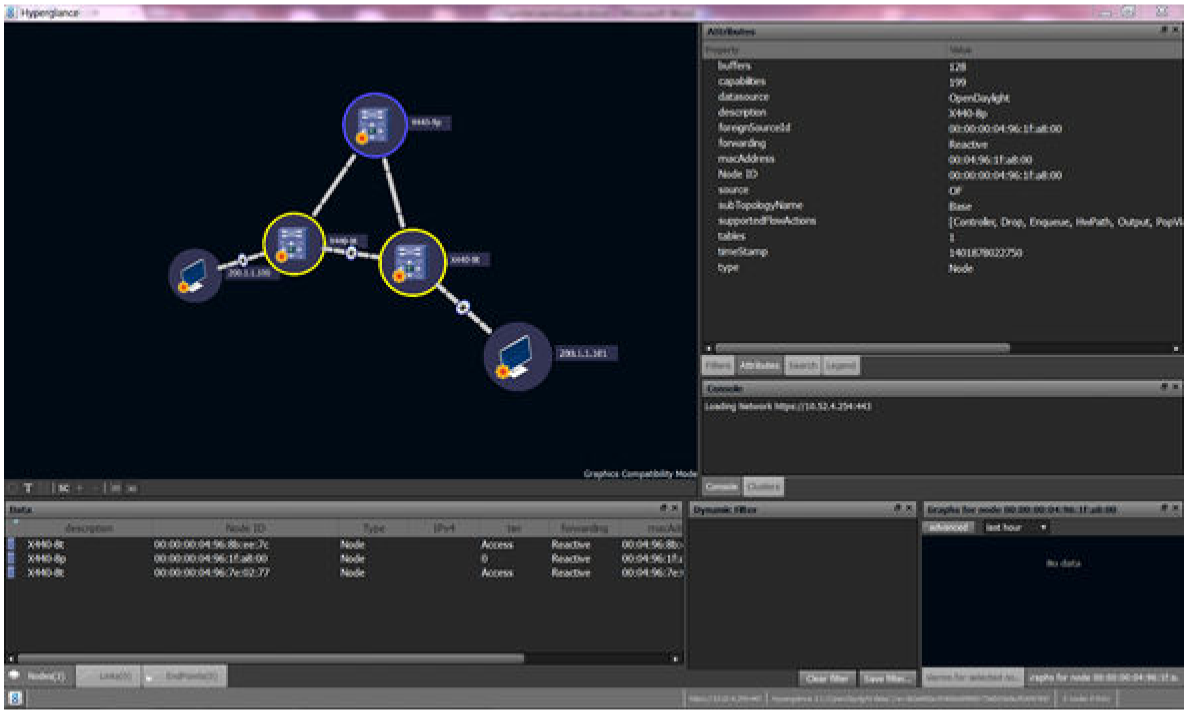
Flows (proactive default flows, reactive, and learned) appear in the Attributes Panel (upper right). Additional flows pushed from Affinity also appear.



Property	Value
buffers	128
capabilities	199
datasource	OpenDaylight
description	X440-8t
foreignSourceId	00:00:00:04:96:8b:ee:7c
forwarding	Reactive
macAddress	00:04:96:8b:ee:7c
Node ID	00:00:00:04:96:8b:ee:7c
source	OF
subTopologyName	Base
supportedFlowActions	[Controller, Drop, Enqueue, HwPath, Output, PopV
tables	1
tier	Access
timeStamp	1401975766079
type	Node

Figure 40: Hyperglance

You can select all switches by pressing CTRL + click. Switches then populate the Data Panel.



Attributes

Property	Value
buffers	128
capabilities	199
datasource	OpenDaylight
description	X440-8p
foreignSourceId	00:00:00:04:96:1f:a8:00
forwarding	Reactive
macAddress	00:04:96:1f:a8:00
Node ID	00:00:00:04:96:1f:a8:00
source	OF
subTopologyName	Base
supportedFlowActions	[Controller, Drop, Enqueue, HwPath, Output, PopVl
tables	1
timeStamp	1401878022750
type	Node

Console

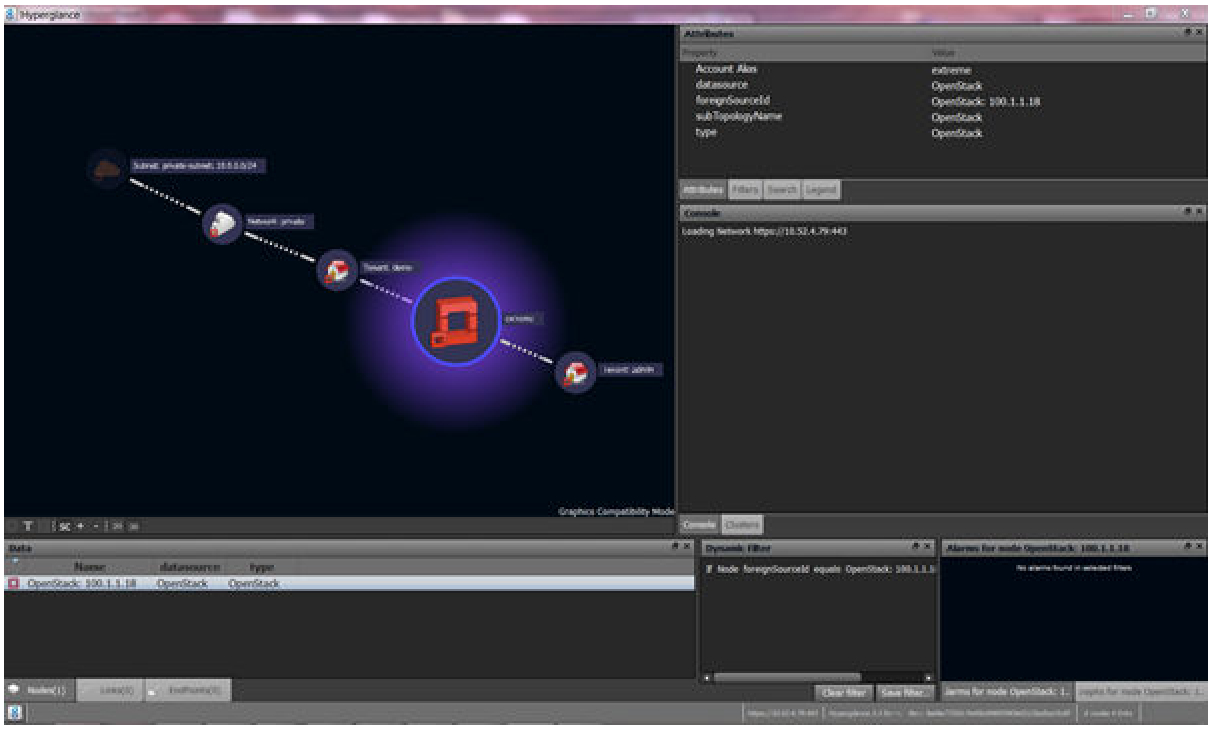
```
Loading Network https://10.52.4.254:443
```

Data

description	Node ID	Type	IPv4	tier	forwarding	macAd
X440-8t	00:00:00:04:96:8b:ee:7c	Node		Access	Reactive	00:04:96:8b:
X440-8p	00:00:00:04:96:1f:a8:00	Node		0	Reactive	00:04:96:1f:a
X440-8t	00:00:00:04:96:7e:02:77	Node		Access	Reactive	00:04:96:7e:

Figure 41: Hyperglance

The following image shows an example of an OpenStack discovered network:



Attributes

Property	Value
Account Alias	extreme
datasource	OpenStack
foreignSourceId	OpenStack: 100.1.1.18
subTopologyName	OpenStack
type	OpenStack

Console

```
Loading Network https://10.52.4.79:443
```

Dynamic Filter

```
Node foreignSourceId equals OpenStack: 100.1.1.18
```

Data

Name	datasource	type
OpenStack: 100.1.1.18	OpenStack	OpenStack

Figure 42: Hyperglance