

# **Major Project Report on** **“REST API Verification and 2.5.3 monitoring feature”**

In partial fulfillment of requirements for the degree of  
**Bachelor of Technology (B. Tech.)**  
in  
**Computer Science and Engineering**



**Submitted by**  
Ms. Divya Aswani (160328)

**Under the Guidance of**  
Mr. Praveen Kumar Arora

*Department of Computer Science and Engineering*  
**SCHOOL OF ENGINEERING AND TECHNOLOGY**  
**Mody University and Science and Technology**  
**Lakshmangarh, Dist. Sikar-332311**

July 2020

## ACKNOWLEDGEMENT

It is my immense pleasure to work as an intern in **Hewlett Packard Enterprise**. It was great learning and professional development experience and I would like to express my deepest gratitude to HPE for providing such kind of opportunity for students to broaden their perspective on professional level working in the fields of Computer Science Engineering and also for organizing the whole internship program. Also a great thanks to **Mody University of Science and Technology** for providing us with such internships and supporting us throughout the internship period

I would also like to extend my supreme gratitude to **Mr. Praveen Kumar Arora**, Manager , Aruba Campus Switching for the guidance and constant supervision and encouragement also for providing necessary information regarding the project & also for their support in completing the projects.

I would be failing in my duty if I do not express my deep sense of gratitude to **Dr. V.K. Jain**, Dean of Academics MU and **Dr. A. Senthil** , Head of Department SET , Mody University for his support and corporation throughout the internship program and also all other faculty members for encouragement and contributed directly or indirectly in completing the project.

**Divya Aswani (160328)**

## **CERTIFICATE**

This is to certify that the major project report entitled “REST API Verification and 2.5.3 monitoring feature” submitted by Ms. Divya Aswani, as a partial fulfillment for the requirement of B. Tech. VIII Semester examination of the School of Engineering and Technology, Mody University of Science and Technology, Lakshmangarh for the academic session 2019-2020 is an original project work carried out under the supervision and guidance of Mr. Praveen Kumar Arora has undergone the requisite duration as prescribed by the institution for the project work.

### **PROJECT GUIDE:**

**Signature:**

**Name: Mr. Praveen Kumar Arora**

**Date:**

### **HEAD OF DEPARTMENT**

**Signature:**

**Name: Dr. A. Senthil**

**Date:**

### **EXAMINER-I:**

**Signature:**

**Name:**

**Date:**

### **EXAMINER-II**

**Signature:**

**Name:**

**Date:**

## **ABSTRACT**

The project mainly involved tasks. The major tasks were verifying and documenting the FG guide for MSTP and LLDP protocols, STP-NI and understanding 2.5.2 Monitoring feature. In task 1, the present documentation only had the functionality guide with CLI commands which was not the best way to check REST APIs , so studying the working of CURL commands and looking for attributes in suitable table so that the CURL command can be generated was the main task. Not only testing but also updating the same commands in existing FG guide was also a part of the task , the documents then went for review , further the document was reviewed and verified , it was merged with the existing document. The next task was solo project, the STP-NI project which stands for Spanning-tree protocol-Net Insight project. What was expected from me was to understand the TCN (Topology Change Notification) generation locally on PMG set up and look for syslog events to find which switch generated the TCN first. After accomplishing this task, the next task was to do the same but on Central by onboarding the switches, and flapping one of the ports so the TCN is generated on switches and retrieving them by creating a new alert template in Alerts and Events Page. Further the code changes has been suggested and waiting for approval so that further NI-MOCKUP can be designed and developed. The current task is mainly related to the new release 2.5.3 which is yet to be launched, once this release would be launched the monitoring feature and all the IFDs and CFDs related to monitoring would come under our team. So, understanding the existing release was a task, each team member was provided with a module, and were given a generic template of what should be covered. The module assigned to me was VLANs, from its UI frontend workflow to its backend workflow everything including databases was covered and also a demo was given by me to the team. Later, some defects were assigned to me which were resolved successfully.

## Table of Contents

<b>Sr.no.</b>	<b>Topics</b>	<b>Page No.</b>
<b>1.</b>	<b>Introduction.....</b>	<b>1-5</b>
1.1	-Present System.....	1-3
1.2	-Proposed System.....	4-5
<b>2.</b>	<b>System Design.....</b>	<b>6-8</b>
2.1	-System flowchart.....	6-8
<b>3.</b>	<b>Hardware and Software Details/ Standards.....</b>	<b>9</b>
3.1	-Software Requirements.....	9
<b>4.</b>	<b>Implementation Work Details.....</b>	<b>10-22</b>
4.1	-Real life applications.....	10-11
4.2	-Data implementation and program execution.....	11-22
<b>5.</b>	<b>Source Code/ Simulation Code/ Commands.....</b>	<b>23-29</b>
<b>6.</b>	<b>Input/output Screens.....</b>	<b>30-33</b>
<b>7.</b>	<b>System Testing.....</b>	<b>34-35</b>
<b>8.</b>	<b>Individual Contribution.....</b>	<b>36</b>
<b>9.</b>	<b>Conclusion.....</b>	<b>37</b>
9.1	-Limitations and Future Scope.....	37
<b>10.</b>	<b>Bibliography.....</b>	<b>38</b>
<b>11.</b>	<b>Annexures.....</b>	<b>39</b>
	-Plagiarism Report.....	39

### **Table of Figures**

<b>Fig.no.</b>	<b>Figure Title</b>	<b>Page no.</b>
2.1	REST API verification and documentation	6
2.2	STP-NI design	7
2.3	Message Flow	8
6.1	Topology for REST API testing	30
6.2	Topologies before and after flapping of ports	30
6.3	Syslog events	31
6.4	Events reflected on central	31
6.5	Alert for a switch generating multiple TCN in short time	31
6.6	NI Mock up	32
6.7	Git Bash screenshot for UI API	32
6.8	Mock UI for running UI API	33
6.9	The elastic search for VLAN page	33
7.1	NBAPI for Vlan Info	35

### **Table of Tables**

<b>Fig.no.</b>	<b>Table Name</b>	<b>Page no.</b>
4.1	login parameters	11
4.2	logout parameters	11

# Chapter1: Introduction

---

## 1. INTRODUCTION

Throughout the internship, the project was given in the form of several tasks of different releases. The tasks were all related to Aruba Campus Switching. The list of all the tasks is as follows:

### **I. REST API documentation and verification of protocols:**

A RESTful API also referred to as a RESTful web service or REST API -- is based on representational state transfer (REST). The REST\_API provides a management interface to interact with a switch. You can utilize the API to retrieve status and statistics information of the switch, as well as to set and change the configuration of the switch.

The REST API can be tested using 2 features:

1. Using CLI commands
2. Using CURL commands

The testing with CLI includes testing the REST API on switch. This API call allows us to execute almost any command on the switch. If the switch information has a direct API call we should probably use that.

REST supports four operations that must be documented if available for specific features:

- GET
- POST
- PUT
- DELETE

The protocols which would be tested:

1. MSTP (Multiple spanning-tree) protocol
2. LLDP (link layer discovery) protocol



## **II. STP NI**

STP-NI refers to Spanning Tree protocol- Net Insights .The task involved the Setting up the topology using (4-5) switches, (2) some of them connected to yellow LAN to get the IP. Configuring each switch with Spa protocol commands .Creating and configuring the VLANs for the connected ports. And getting their event logs to check the generation of TCN (topology change notifications) events. After generating the events, checking which switch generated the TCN first. Performing the same task but by onboarding the devices and generating the alerts on the centralite.

## **III. 2.5.3 Monitoring feature:**

This task involved the deep understanding of Switch Monitoring feature of 2.5.2 release to manage the upcoming 2.5.3 Monitoring feature's CFDs (Customer Found defects) or IFDs( Internal found defects)

The module assigned to me was VLANs .

The overall workflow which includes the frontend flow , backend flow , kafka topics , NBAPI testing etc.

## **1.1 PRESENT SYSTEM**

### **I. REST API documentation and verification of protocols:**

The present documentation only had the functionality guide with CLI commands. The only way a controller was configured was using command line interface (CLI) or controller user interface (Web UI). This created hindrance to automation because the CLIs typically changed over time and Web UI could not be automated easily because most of the pages were hand crafted. The Web UI also used CLI to communicate to the back end, which was hard coded and not easily extensible.

So, the verification of REST APIs using CURL commands was required and hence, the testing of each MSTP or LLDP attribute was done and documented in the existing functionality guide.

### **II. STP-NI**

The present system didn't had the time guideline to check which port will generate the TCN first if some certain port of some specific switch is flapping. So, checking which port generated the TCN and at what time , also the time difference between the TCN alert generation was not there. Also the centralite didn't had the alert for TCN .

### **III. 2.5.3 Monitoring Feature**

The present monitoring feature belongs to 2.5.2 release. The 2.5.3 release would be launched in july and hence, the monitoring feature would be under our team. The previous Monitoring feature was managed by some other team but after 2.5.3 release the MON feature with all the CFDs/IFDs would be managed and resolved by Switch CX team.

## **1.2 PROPOSED SYSTEM**

### **I. REST API documentation and verification of protocols:**

The proposed system is of checking REST API with CURL commands and not just CLI commands. The CURL commands will include:

- \* `GET`: Read a resource. To retrieve the data/values of attributes on calling APIs.
- \* `POST`: Create a resource. To initialize the values of non-existing attributes. Also it must specify all the pre-required fields.
- \* `PUT`: Update a resource. To modify the value of existing attributes.
- \* `DELETE`: Delete a resource. To delete a complete instance.

Verifying and documenting all these commands to update the existing functionality guide.

### **II. STP-NI**

The proposed system included the creation of alert template for Switch RPVST Topology Change Notification with severity levels. It was needed to be configured to get those alerts. The flapping of any ports would generate the alert with the name of the switch which generated the TCN first. It was included in 2 phases:

#### **Phase 1:**

- The switch sends syslog event to central every 5 minutes.
- The syslog events specific to STP can be used to generate alerts on central.
- Based on specific events and corresponding alerts, the user can be suggested some solutions for problem in stp.

#### **Phase 2:**

- Use stp and lldp to show topology on central.

As every switch has only node level information for stp, central need to combine stp and lldp info to generate viewable.

### **III. 2.5.3 Monitoring Feature**

The proposed system would be understanding the overall workflow of 2.5.2 release to get the gist of 2.5.3 release. The Vlan Module included:

- UI Construction Info
- CX switch data
- Central-CE module
- Central- Kafka Message streaming
- Central-Monitoring App
- Central – UI and NB API's
- Central – Login to databases
- Central - Rendering module/feature information to central UI

## Chapter2: System Design

### 2.1. System Flowchart

#### I. REST API verification and testing:

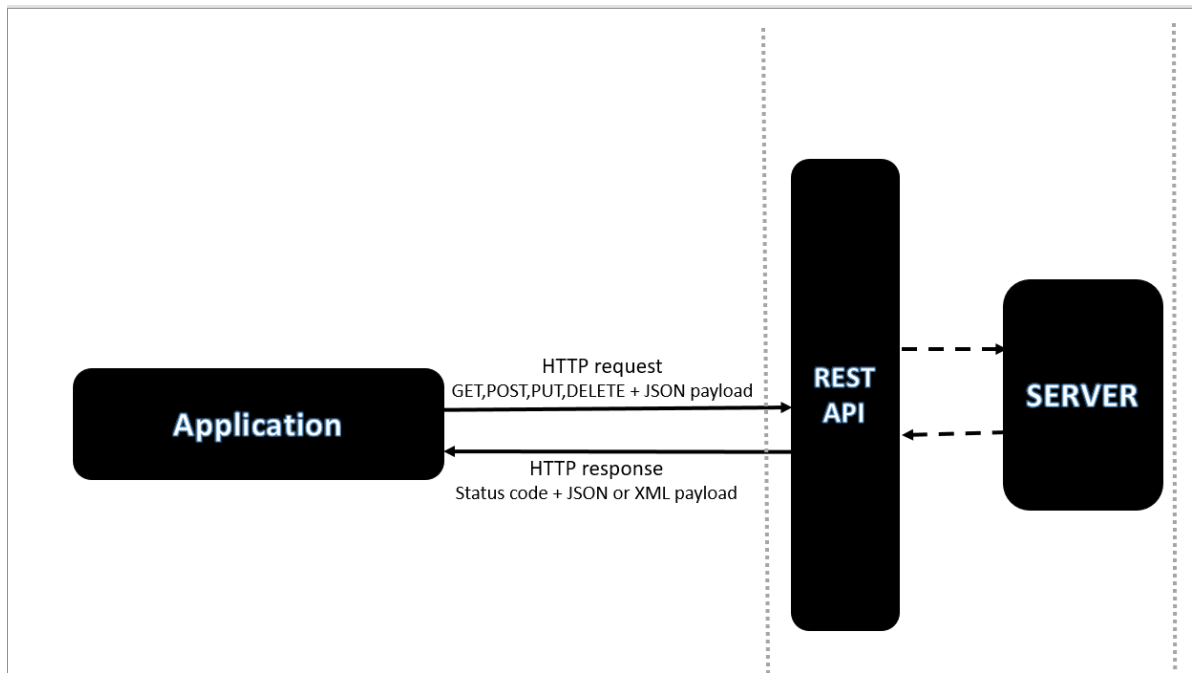


Fig. 2.1 REST API verification and documentation

The Application sends the HTTP request like GET , POST , PUT , DELETE request calls to the server and the server would respond with the status code as well as the required output in JSON or XML format

## II. STP NI

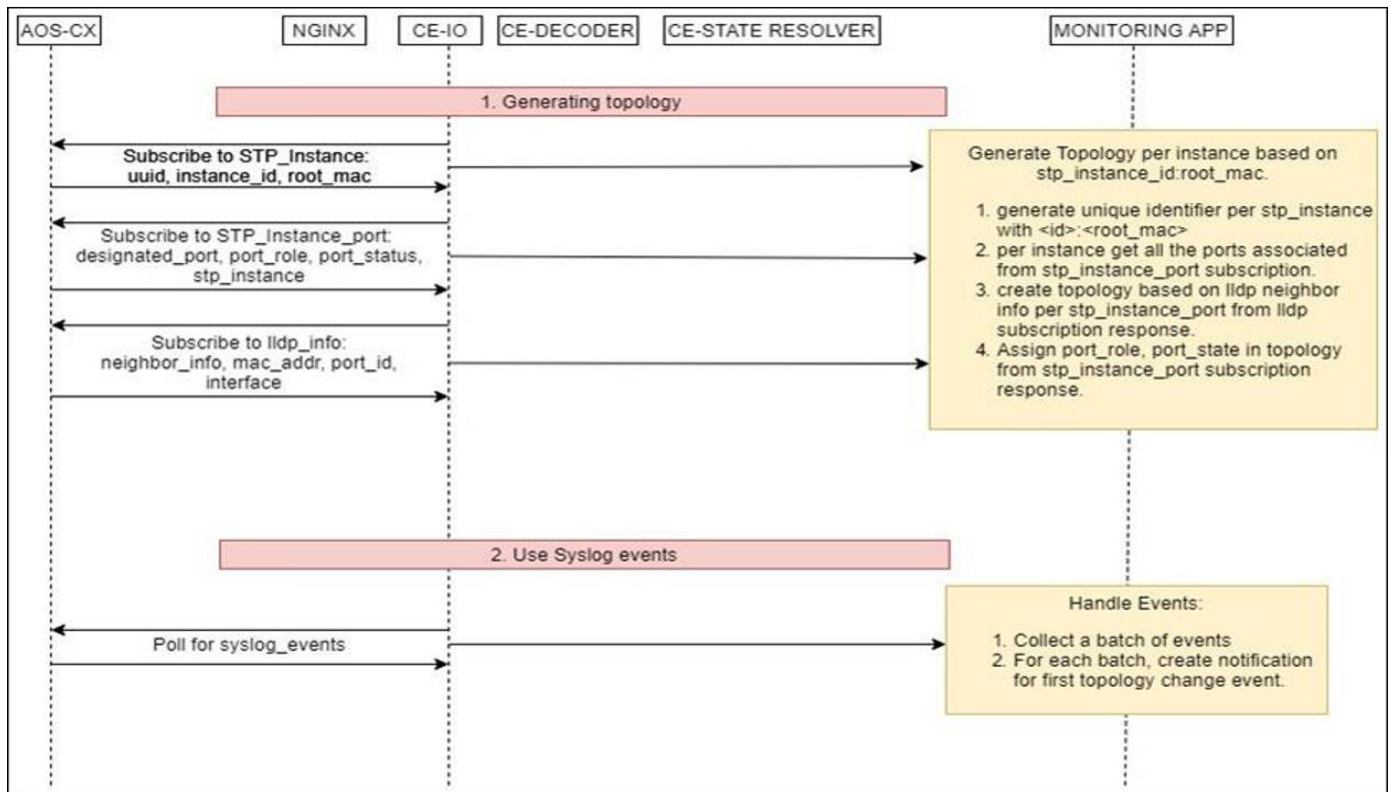


Fig. 2.2 STP-NI design

The 2 phases are as following:

1. Generating topology - For this we shut down and toggled a link in an stp-instance and monitored the set of events on central.
2. Use Syslog events - For the events a meaningful alert was created for user to be notified.

### III.2.5.3 Monitoring feature

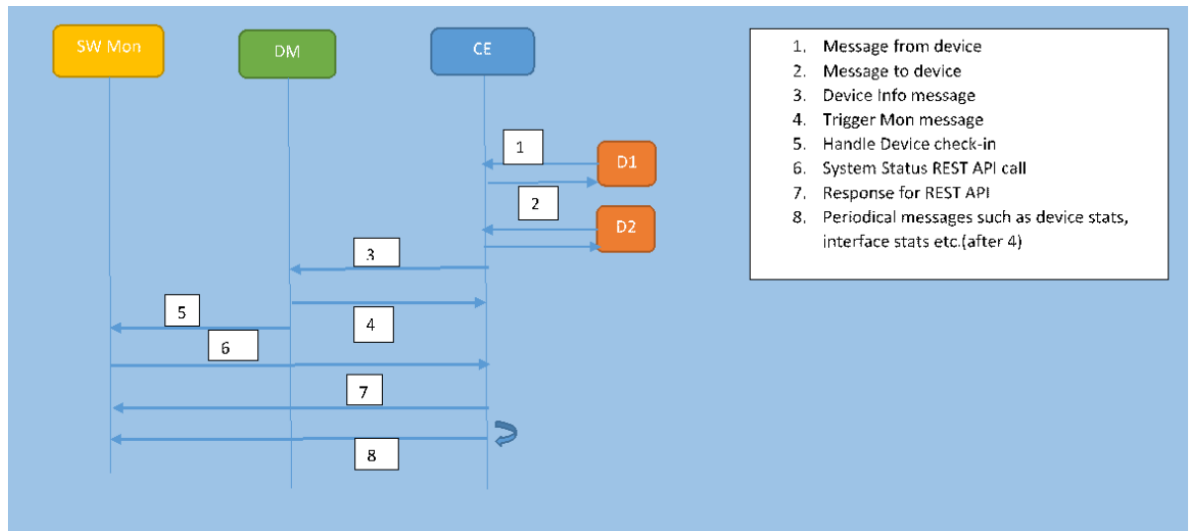


Fig. 2.3 Message Flow

The UI component will have UI-API which will call for switch apis in views files and hence, these api calls will further call functions which would return json response having all data required from server. The following data will pass from kafka queues and would be passes to databases and then would be delivered.

## Chapter3: Hardware and Software Details

---

### 3.1. Software Required :

- 1) Aruba AD account : Fundamental Requirement
- 2) Aruba Confluence web pages/JIRA : To get the access to HPE exclusive study content
- 3) Cloud-cop : To enable/disable cl instance
- 4) QA Activate : To add device to activate server
- 5) CI Jenkins access : To create cl instance
- 6) Central ACP training Documents : Content related to ARUBA cloud platform
- 7) VM access (India) : To access pod-manager
- 8) VM access (Roseville) : To access pod-manager
- 9) X2goClient : Platform for pod manager
- 10) Mobaxterm : To use it as terminal
- 11) Central lite : Aruba Central Lite
- 12) Curl Commands : POST, GET, PUT, DELETE
- 13) CLI commands : Of mstp and lldp protocols
- 14) Angular JS : For scripts



## Chapter4: Implementation Work Details

---

### 4.1. Real Life Application :

#### I. REST API Verification and Testing

REST technology is generally preferred over the more robust Simple Object Access Protocol (SOAP) technology because REST uses less bandwidth, making it more suitable for efficient internet usage. The REST used by browsers can be thought of as the language of the internet. With cloud use on the rise, APIs are being used by cloud consumers to expose and organize access to web services. REST is a logical choice for building APIs that allow users to connect to, manage and interact with cloud services flexibly in a distributed environment.

#### II.STP NI

To have a big picture , to see what it takes from the ArubaCX-OS id to provide basic NetInsight Support on par with ArubaOS.

Characteristics used by NetInsight to monitor Switch :

- 1.System Health – CPU/memory based on baseline thresholds
- 2.Port errors/counters (errors/discards, etc)
- 3.Port flaps
- 4.PoE analytics to identify issues in allocation

Under these characteristics NetInsight Apps subscribe to specific topics which have dedicated fields that help in providing Insights for the Switch.

### III. 2.5.3 MON Feature

All Monitoring features comes under Monitoring app and Monitoring App is an integral part of Aruba Central. Its purpose is to process the periodic AMON messages (Protobuf from IAP) from the devices and present the state and statistical data to the customer, appropriately aggregated and formatted. In addition, the Monitoring App disseminates processed data for other apps consumption.

## 4.2. Data Implementation and Program Execution

### I. REST API Verification and Testing

**MSTP Protocol:** Without spanning tree, having more than one active path between a pair of nodes causes loops in the network, which can result in duplication of messages, leading to a “broadcast storm” that can bring down the network.

Multiple-Instance spanning tree operation (802.1s) ensures that only one active path exists between any two nodes in a spanning-tree instance. A spanning-tree instance comprises a unique set of VLANs, and belongs to a specific spanning-tree region. A region can comprise multiple spanning-tree instances (each with a different set of VLANs), and allows one active path among regions in a network.

### Config context commands

#### #### Curl Login ####

To access any configuration element -- whether it is GET or SET on the object, the user first has to login to the Mobility Master.

The following is a sample CURL command used by the user to log in to the Mobility Master:

#### <syntax>

```
curl --insecure -c "aruba-cookie" -d "username=&password=" https://:4343/v1/api/login
```

*<The --insecure option can be used with the curl command if the certificate of the Mobility Master cannot be validated.>*

Example:

```
curl --noproxy 10.102.149.165 -k -X POST -c /tmp/auth_cookie -H 'Content-Type: application/x-www-form-urlencoded' -d 'username=admin&password=admin' https://10.102.149.165/rest/v1/login
```

Table 4.1 : login parameters

Parameters	Description
<username>	Username of the user.
<password>	Password of the user.
<controller-ip>	IPv4 address of the Mobility Master.

#### #### Curl Logout ####

To close all the interactions, you need to logout from the Mobility Master.

The following is a sample CURL command used by the user to log out of the Mobility Master:

<syntax>

```
curl -c "aruba-cookie" https://:4343/v1/api/logout
```

*<The --insecure option can be used with the curl command if the certificate of the Mobility Master cannot be validated.>*

Example:

```
curl -k -X POST -b /tmp/auth_cookie --header 'Content-Type:application/json' --header 'Accept: application/json' https://172.25.0.2/rest/v1/logout
```

table 4.2 : logout parameter

Parameters	Description
<controller-ip>	IPv4 address of the Mobility Master.

## Example : Vlan to an Instance

VLAN to an instance

### Syntax

```
`spanning-tree instance <instance-id> vlan <VLAN-ID>`
```

### Description

This command create a new instance with VLAN(s) mapped or map VLAN(s) to an existing one. Each instance must have at least one VLAN mapped to it.

When VLANs are mapped to an instance they are automatically unmapped from the instance they were mapped to before.

Any MSTP instance can have all the VLANs configured in the switch.

---

### Authority

Admin users.

### Parameters

Parameter	Status	Syntax	Description
:-----	:-----	:-----	:-----
*instance-id*	Required	<1-64>	Specifies the MSTP instance number
*VLAN-ID*	Required	<1-4094>	Specifies the VLAN-ID number

### Examples of CLI

---

```
switch# configure terminal
switch(config)# spanning-tree instance 1 vlan
<1-4094> Configure VLAN to an MST instance
switch(config)# spanning-tree instance 1 vlan 2
---
```

### Curl command

#### #GET#

```
curl -k -X GET -b /tmp/auth_cookie -H 'accept: application/json'
'https://10.100.73.254/rest/v10.04/system/stp_instances/mstp,1?attributes=vlan_ids'
```

#### #POST#

```
curl -k -X POST -b /tmp/auth_cookie -H 'Content-Type:application/json' --header
'Accept:application/json' 'https://10.102.165.233/rest/v10.04/system/stp_instances' --data
'{"instance_type": "mstp", "stp_instance_id": 1, "origin": "configuration", "vlan_ids": [2]}
```

#### #PUT#

```
curl -k -X PUT -b /tmp/auth_cookie -H 'Content-Type:application/json' --header
'Accept:application/json' 'https://10.102.165.233/rest/v10.04/system/stp_instances/mstp,1' --
data '{"vlan_ids": [2]}
```

### Remove VLAN from instance

#### Syntax

```
`no spanning-tree instance <instance-id> vlan <VLAN-ID>`
```

#### Description

This command removes the VLAN-ID from the MSTP instance.

#### Authority

Admin users.

#### Parameters

Parameter	Status	Syntax	Description
:----- :----- :----- :-----			
*instance-id*	Required	<1-64>	Specifies the MSTP instance number
*VLAN-ID*	Required	<1-4094>	Specifies the VLAN-ID number

#### Examples of CLI

...

switch# configure terminal

switch(config)# no spanning-tree instance 1 vlan

<1-4094> Configure VLAN to an MST instance

switch(config)# no spanning-tree instance 1 vlan 2

Curl command

#GET#

```
curl -k -X GET -b /tmp/auth_cookie -H 'accept: application/json'
'https://10.100.73.254/rest/v10.04/system/stp_instances/mstp,1?attributes=vlan_ids'
```

#DELETE#

```
curl -k -X DELETE -b /tmp/auth_cookie --header 'Content-Type:application/json' --header
'Accept:application/json' https://10.101.157.126/rest/v10.04/system/stp_instances/mstp/1
<!-- DELETE command is used to delete the complete instance -->
```

#PUT#

Not Configurable

...

## LLDP PROTOCOL

The Link Layer Discovery Protocol (LLDP) is an industry-standard, vendor-neutral method to allow networked devices to advertise capabilities, discover and identify other LLDP enabled devices and gather information in a LAN. A maximum of 4 neighbors can be identified on a particular interface.

LLDP is supported on physical interfaces and Out-Of-Band Management (OOBM) interfaces and it is not supported on logical interfaces.

The following bullet list contains some of the information gathered by LLDP:

1. System name and description
  - a. Port name and description

2. VLAN name and identifier
  - a. IP network management address
3. Device Capabilities (for example, switch, router, or server)
4. MAC address and physical layer information
5. Power information

#### ### Curl Login ###

```
curl --no-proxy 10.102.149.165 -k -X POST -c /tmp/auth_cookie -H 'Content-Type: application/x-www-form-urlencoded' -d 'username=admin&password=admin' https://10.102.149.165/rest/v1/login
```

#### ### Curl Logout ###

```
curl -k -X POST -b /tmp/auth_cookie --header 'Content-Type:application/json' --header 'Accept: application/json' "https://172.25.0.2/rest/v1/logout"
```

All LLDP configuration commands except ``lldp transmit`` and ``lldp receive`` work in the global config context. The ``lldp transmit`` and ``lldp receive`` commands set the interface level tx/rx permission for LLDP packets.

Hence, these commands can only be configured in the interface config context.

### Set LLDP txdelay

#### Syntax

```
`lldp txdelay <time>`
```

#### Description

This command sets the amount of time (in seconds) to wait before sending LLDP advertisements from any interface.

The `lldp txdelay` should be lesser than or equal to  $(0.25 \times \text{timer})$  value.

## Authority

Admin Users

## Parameters

Parameter	Status	Syntax	Description
*time*	Required	1-8192	Select txdelay between 1 and 8192 seconds.

## Examples

---

```
switch# configure terminal
```

```
switch(config)# lldp txdelay 5
```

---

## Curl Commands

### # GET #

```
curl -k -X GET -b /tmp/auth_cookie -H 'accept:application/json'
'https://10.100.40.240/rest/v1/system/?attributes=other_config'
```

### #POST#

Not configurable

### #PUT#

```
curl -k -X PUT -b /tmp/auth_cookie -H 'Content-Type:application/json' --header
'Accept:application/json' 'https://10.100.48.240/rest/v1/system/' --data
'{"other_config":{"lldp_txdelay":5}}'
```

## Set LLDP txdelay to default

### Syntax

```
`no lldp txdelay`
```



## Description

This command resets to default value the amount of time to wait before sending LLDP broadcast from any interface. The default value is 2 seconds.

## Authority

Admin Users

## Parameters

No parameters.

## Examples

```

switch# configure terminal

switch(config)# no lldp txdelay

```

## Curl Commands

### # GET #

```
curl -k -X GET -b /tmp/auth_cookie -H 'accept:application/json'
'https://10.100.40.240/rest/v1/system/?attributes=other_config'
```

### #DELETE#

Not configurable

### #PUT#

```
curl -k -X PUT -b /tmp/auth_cookie -H 'Content-Type:application/json' --header
'Accept:application/json' 'https://10.100.48.240/rest/v1/system/' --data
'{"other_config":{"lldp_txdelay":2}}'
```

## II.STP NI

Steps to generate TCN:

1. Set up the topology using (4-5) switches , (2) some of them connected to yellowLAN to get the IP.
1. Configure each switch as:
  - a. #config
  - b. #spa
  - c. #spa mode rpvt(in case the mvrp mode is enabled , first use command : no mvrp to configure)
2. Create vlan of each port and configure each port.
3. Check the status of spanning-tree interfaces using : sh spa
4. Try pinging sw4 from sw2 in bash as :
5. We can confirm the traffic path by looking at the MAC address table: **show mac address-table dynamic**
  - a. Show mac address-table dynamic
6. Now unplug cable between sw1 and sw3 as:
  - a. Sw1(config)#int 1/1/2
  - b. Sw1(config-if)#shutdown
  - c. Swi(config-if)#exit
7. Check the event log files as
  - a. #start-shell
  - b. \$cd /var/log
  - c. vim event.log
8. Event.log file for each switch
9. TCN Generating Time Difference

The cable between SW2 and SW4 was disabled. So the tcN is generated in the following manner:

Example:

Sr. no.	Switch	Time Stamp	Order	Time Difference(ref. SW2)
1	SW1	04:35:36	2nd	00:00:01
2	SW2	04:35:35	1st	00:00:00
3	SW3	04:37:14	3rd	00:01:39
4	SW4	04:39:20	4th	00:02:06

The next step was Generate the TCN alerts on Centralite so the first step is to onboard a device on central and activate. The device to be onboarded was 6300 with filters and corresponding image built on it. Adding the required certificate on the switch .Then providing the vrf and cl instance locations for connectivity. Adding the switch infi in frm5 / activate server and also on device inventory. At the end restarting demon and checking the connectivity.

### III. 2.5.3 Monitoring feature

#### UI Construction Info:

- Information related to how UI components are built.
- Specific mention of pages related to your module.
- Connected components from where you pull and render values to components/widgets.

#### CX Switch Data

- REST API request & response. Nothing but Incoming data from Switch.
- Investigate whether the switch info is being retrieved via Polling or via Subscription based mechanisms.

#### Central – CE Module

- Entry point/CE function responsible to accept incoming data from CX Switch.

- Logic description in brief about what all attributes/info of CX Raw data is being absorbed by CE module

### **Central – GPB Message format and Schema File for CX message transformation**

- Specify message-name and it's schema file here.
- Mention about the mapping from CX Switch data to GCP message format.
- Explain how the new re-enveloped CX Switch data is sent out from CE Module. (via ACP Events/Kafka Topics related to this message – is it a Switch State/Stats information etc. )

### **Central – Kafka Message Streaming**

- Explain what kind of the message is being sent from CE module in Kafka queue - Switch state/stats information. What is the frequency at which the messages are arriving to the upstream app.
- Depending on the rate at which the Kafka messages are coming, understand and explain if an additional aggregation is necessary/handled already in cx-switch-monitoring app.
- If an additional aggregation logic is already handled, explain the logic and mention the related adapter classes used for processing to consolidate all related messages and send to upstream app.

### **Central – Monitoring App [ CX Switch Specific]**

- Explain where the Kafka messages are first intercepted in the App. Explain how the Kafka topic looks like – header and message body. What is the data payload content and what all other information is sent to App along with this message.
- Specify it is a State or a Stats message arrived and where it will be handled in Kafka-apps package under monitoring app.
- Explain in brief the code flow between – Kafka-App ->Adapter Class->Service Class [file-names and entry function points is a MUST here]
- Explain the data bases used to store and retrieve your message/component value.
- Explain if you either directly writing to DB or deleting it to other worker job/app to push to DB.

### **Central – UI and NB API 's**

- Mention the main file where - UI API's are defined.  
List all the available UI API's and the important attributes they accept.
- Mention the base-path of the file where the NB API is defined (in Swagger.yaml file)  
List all the available NB API's and the important attributes they accept.

### **Central – Login to databases**

- Specify details of how you login to the databases (Redis/PostGRES/Cassandra/ES) using CL –CLI or using any REST API Client like PostMan and provide sample example of querying the database and any important DB commands (if any)

### **Central - Rendering module/feature information to central UI**

- Explain how the UI API responses (in json dict format) are parsed and UI pages/components value is getting rendered in central UI

### **Other Generic Info to capture:**

- All associated Pod Names
- CX Switch OVSDB Table Name to check dump

## Chapter5: Source Code/Simulation Code/Commands

---

### I. REST API Verification and Testing:

#### 5.1. CLI commands:

##### MSTP

6300# login username admin

Password <blank> or admin

6300# configure terminal [Config mode]

6300(config)# spanning-tree mode mstp [Set mode to mstp]

6300(config)# sh spanning-tree [check if STP is enabled]

Spanning-tree is disabled

6300(config)#spanning-tree [Enable Spanning-tree Protocol]

6300(config)# sh spanning-tree

Spanning tree status : Enabled Protocol: MSTP

##### MST0

Root ID Priority : 32768

MAC-Address: d0:67:26:49:f1:38

This bridge is the root

Hello time(in seconds):2 Max Age(in seconds):20

Forward Delay(in seconds):15

Bridge ID Priority : 32768

MAC-Address: d0:67:26:49:f1:38

Hello time(in seconds):2 Max Age(in seconds):20

Forward Delay(in seconds):15

Port	Role	State	Cost	Priority	Type
------	------	-------	------	----------	------

-----

```

6300(config)# spanning-tree config-name x          [Sets the configuration name of the MST
region in which the switch resides.]
6300(config)# spanning-tree instance 1 vlan 2      [create a new instance with VLAN(s)
mapped or map VLAN(s) to an existing one.]
6300(config)# interface 1/1/1                    [configure interface]
6300(config-if)# no shutdown
6300(config-if)# no routing
6300(config-if)# exit
6300(config)# sh int mgmt                        [To display the configuration information for a
management interface]

Address Mode          : dhcp
Admin State           : up
Mac Address           : d0:67:26:49:f1:39
IPv4 address/subnet-mask : 10.101.60.199/24
Default gateway IPv4   : 10.101.60.1
IPv6 address/prefix    :
IPv6 link local address/prefix: fe80::d267:26ff:fe49:f139/64
Default gateway IPv6   :
Primary Nameserver     :
Secondary Nameserver   :

```

## **LLDP**

```

switch# configure terminal
switch(config)# lldp

switch(config)# lldp management-ipv4-address 16.93.49.9

switch# configure terminal
switch(config)# lldp select-tlv management-address
switch(config)# lldp select-tlv port-description
switch(config)# lldp select-tlv port-vlan-id
switch(config)# lldp select-tlv system-capabilities
switch(config)# lldp select-tlv system-description

```

```
switch(config)# lldp select-tlv system-name
```

```
switch# show lldp configuration
```

#### LLDP Global Configuration

```
=====
```

```
LLDP Enabled           : No
LLDP Transmit Interval : 30
LLDP Hold Time Multiplier : 4
LLDP Transmit Delay Interval : 2
LLDP Reinit Timer Interval : 2
```

#### TLVs Advertised

```
=====
```

Management Address

Port Description

Port VLAN-ID

System Capabilities

System Description

System Name

#### LLDP Port Configuration

```
=====
```

PORT	TX-ENABLED	RX-ENABLED
------	------------	------------

```
-----
```

1/1/1	Yes	Yes
1/1/2	Yes	Yes
1/1/3	Yes	Yes
1/1/4	Yes	Yes
1/1/5	Yes	Yes
1/1/6	Yes	Yes



.....

.....

mgmt        Yes        Yes

^^^

switch# show lldp tlv

TLVs Advertised

=====

Management Address

Port Description

Port VLAN-ID

System Capabilities

System Description

System Name

MED PoE

Dot3 PoE

^^^

## II.     STP NI

Alert template :

SWITCH\_RPVST\_TOPOLOGY\_CHANGE:

```
{'name': 'Switch RPVST Topology Change',
 'name_desc': gettext(u'Switch RPVST Topology Change'),
 'type': 'device',
 'category': category_dict['SWITCH'],
 'subject': 'Switch Topology Change alert',
 'description': 'NI_POC: First TCN generated for previous batch of '
                'tcn_events on switch: {least_src} at {least_src_time}',
 "rule_meta": {
    "duration": {
```

```

        "max": 120,
        "min": 5
    },
    'group': [],
    'label': [],
    'site': [],
    'device_id': [],
    'device_type': DeviceFamily.SWITCH,
    'delivery_options': [1, 2, 3, 4],
    "allow_multiple_rules": False,
    'allow_multiple_severity_in_rule': False,
    'condition': {}
},
"default_rules": [{
    "conditions": [{"severity": 2}],
    "duration": 10,
    "group": [],
    "rule_number": 0
}
]
},

```

### III. 2.5.3 Monitoring Feature

#### REST API request & response

- `/monitor/v2/switch/info?::switch_id_key::=::switch_id_value::&from_timestamp=::from_timestamp::&to_timestamp=::to_timestamp::`
- `/monitor/v2/switch/vlan?::switch_id_key::=::switch_id_value::&from_timestamp=::from_timestamp::&to_timestamp=::to_timestamp::&metadata=true`
- `/monitor/v2/switch/faceplate?::switch_id_key::=::switch_id_value::&from_timestamp=::from_timestamp::&to_timestamp=::to_timestamp::`

- `/monitor/v2/switch/device_health?::switch_id_key::=:switch_id_value::&from_time stamp=:from_timestamp::&to_timestamp=:to_timestamp::`

**The functions which accepts data from CX switch are present in switch.py file in MONITORING Repo:**

```
@app.route('/vlan')
@login_required
@validate_access(current_user)
def get_switch_stack_vlan():
    return json_response(switch_api.get_switch_stack_vlan(current_user.cid, utils_v2.get_arg
s()))
```

>>Returns the json response by calling get\_switch\_stack\_vlan from api/switch\_api

**List of all the UI-APIs available for Vlan page are:**

- `/monitor/v2/switch/info?::switch_id_key::=:switch_id_value::&from_timestamp=:from_timestamp::&to_timestamp=:to_timestamp::`
- `/monitor/v2/switch/vlan?::switch_id_key::=:switch_id_value::&from_timestamp=:from_timestamp::&to_timestamp=:to_timestamp::&metadata=true`
- `/monitor/v2/switch/faceplate?::switch_id_key::=:switch_id_value::&from_timestamp=:from_timestamp::&to_timestamp=:to_timestamp::`
- `/monitor/v2/switch/device_health?::switch_id_key::=:switch_id_value::&from_time stamp=:from_timestamp::&to_timestamp=:to_timestamp::`

**List of all the NB APIs available for Vlan page are:**

- `/monitoring/v1/switches/{serial}/vlan`
- `/monitoring/v1/cx_switches/{serial}/vlan`
- `/monitoring/v1/switch_stacks/{stack_id}/vlan`
- `/monitoring/v1/cx_switch_stacks/{stack_id}/vlan`

**The Vlan information is present in Elastic Search :**

So fetching data from ES in cl bash

**curl -XGET [http://10.164.2.187:9200/\\_cat/indices](http://10.164.2.187:9200/_cat/indices)**

the above curl command would return the name of documents

**curl -XGET "[http://10.164.2.187:9200/central\\_monitor\\_vlan/\\_search](http://10.164.2.187:9200/central_monitor_vlan/_search)" | jq**

the above command will open central\_monitor\_vlan which is the document for vlan

# Chapter6: Input/output Screens/ Model's Photograph

## I. Rest API verification and testing



Fig. 6.1 Topology for REST API testing

## II. STP NI

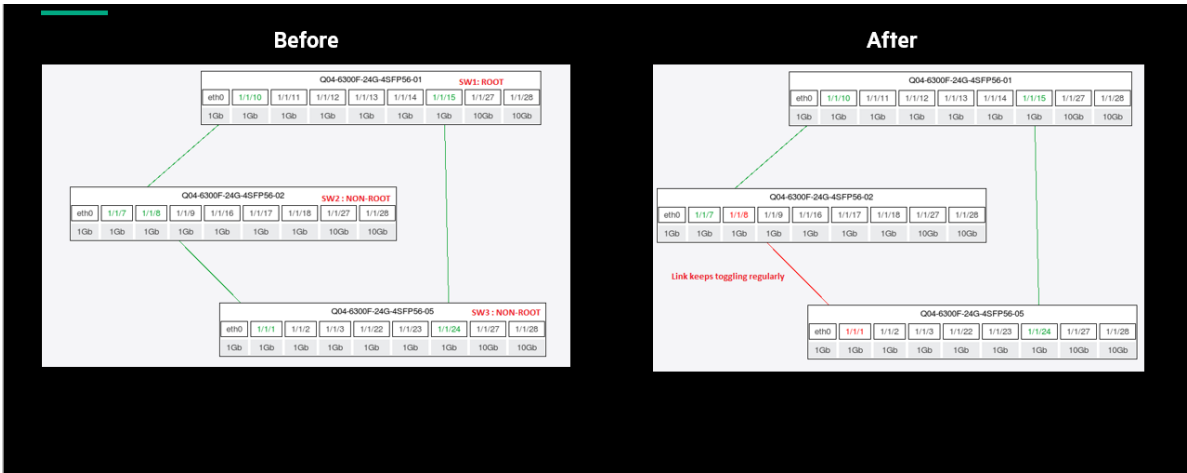


Fig. 6.2 Topologies before and after flapping of ports

## SW2 and SW3: NON-ROOT(sends tcn bpdv regularly in short-intervals)

```

2020-05-29T09:51:53.196989+00:00 6300 lldp[2973]: Event[110]LOG_INFO[MSTR]1|Configured LLDP reinit-delay to 2
2020-05-29T09:51:53.218147+00:00 6300 intf[715]: Event[403]LOG_INFO[MSTR]1|Link status for interface 1/1/8 is up
2020-05-29T09:51:53.239824+00:00 6300 hpe-pvstd[3089]: Event[5007]LOG_WARN[MSTR]1|Topology change generated on port 1/1/8 on VLAN 1.
2020-05-29T09:51:54.720991+00:00 6300 hpe-pvstd[3089]: Event[5006]LOG_WARN[MSTR]1|Topology change received on port 1/1/8 from source: 38:21:c7:5c:d9:a7 on VLAN 1.
2020-05-29T09:51:56.233652+00:00 6300 hpe-pvstd[3089]: Event[5006]LOG_WARN[MSTR]1|Topology change received on port 1/1/8 from source: 38:21:c7:5c:d9:a7 on VLAN 1.
2020-05-29T09:51:57.401220+00:00 6300 lldp[2973]: Event[104]LOG_INFO[MSTR]1|LLDP neighbor 38:21:c7:5c:d9:a0 added on 1/1/8
2020-05-29T09:52:12.425916+00:00 6300 intf[715]: Event[404]LOG_INFO[MSTR]1|Link status for interface 1/1/8 is down
2020-05-29T09:52:12.774265+00:00 6300 hpe-pvstd[3089]: Event[5006]LOG_WARN[MSTR]1|Topology change received on port 1/1/7 from source: 38:21:c7:5a:d1:a0 on VLAN 1.
2020-05-29T09:52:14.580228+00:00 6300 hpe-pvstd[3089]: Event[5006]LOG_WARN[MSTR]1|Topology change received on port 1/1/7 from source: 38:21:c7:5a:d1:a0 on VLAN 1.
2020-05-29T09:52:48.764763+00:00 6300 lldp[2973]: Event[110]LOG_INFO[MSTR]1|Configured LLDP reinit-delay to 2
2020-05-29T09:52:48.780623+00:00 6300 intf[715]: Event[403]LOG_INFO[MSTR]1|Link status for interface 1/1/8 is up
2020-05-29T09:52:48.815169+00:00 6300 hpe-pvstd[3089]: Event[5007]LOG_WARN[MSTR]1|Topology change generated on port 1/1/8 on VLAN 1.
2020-05-29T09:52:42.746659+00:00 6300 hpe-pvstd[3089]: Event[5006]LOG_WARN[MSTR]1|Topology change received on port 1/1/8 from source: 38:21:c7:5c:d9:a7 on VLAN 1.
2020-05-29T09:52:44.210845+00:00 6300 hpe-pvstd[3089]: Event[5006]LOG_WARN[MSTR]1|Topology change received on port 1/1/8 from source: 38:21:c7:5c:d9:a7 on VLAN 1.
2020-05-29T09:52:46.225442+00:00 6300 intf[715]: Event[404]LOG_INFO[MSTR]1|Link status for interface 1/1/8 is down
2020-05-29T09:52:46.513340+00:00 6300 hpe-pvstd[3089]: Event[5006]LOG_WARN[MSTR]1|Topology change received on port 1/1/7 from source: 38:21:c7:5a:d1:a0 on VLAN 1.
2020-05-29T09:52:47.574505+00:00 6300 hpe-pvstd[3089]: Event[5006]LOG_WARN[MSTR]1|Topology change received on port 1/1/7 from source: 38:21:c7:5a:d1:a0 on VLAN 1.
2020-05-29T09:53:04.349828+00:00 6300 lldp[2973]: Event[110]LOG_INFO[MSTR]1|Configured LLDP reinit-delay to 2
2020-05-29T09:53:04.375204+00:00 6300 intf[715]: Event[403]LOG_INFO[MSTR]1|Link status for interface 1/1/8 is up
2020-05-29T09:53:04.401262+00:00 6300 hpe-pvstd[3089]: Event[5007]LOG_WARN[MSTR]1|Topology change generated on port 1/1/8 on VLAN 1.
2020-05-29T09:53:07.740427+00:00 6300 hpe-pvstd[3089]: Event[5006]LOG_WARN[MSTR]1|Topology change received on port 1/1/8 from source: 38:21:c7:5c:d9:a7 on VLAN 1.
2020-05-29T09:53:09.252090+00:00 6300 hpe-pvstd[3089]: Event[5006]LOG_WARN[MSTR]1|Topology change received on port 1/1/8 from source: 38:21:c7:5c:d9:a7 on VLAN 1.

```

Fig. 6.3 Syslog events

## EVENTS REFLECTED ON CENTRAL

### SW2 : NON-ROOT

EVENTS (21)			
OCURRED ON		EVENT TYPE	DESCRIPTION
May 29, 2020, 15:47		RPVST	Event[5006]LOG_WARN[MSTR]1 Topology change received on port 1/1/8 from source: 38:21:c7:5c:d9:a7 on VLAN 1.
May 29, 2020, 15:47		RPVST	Event[5006]LOG_WARN[MSTR]1 Topology change received on port 1/1/8 from source: 38:21:c7:5c:d9:a7 on VLAN 1.
May 29, 2020, 15:47		RPVST	Event[5007]LOG_WARN[MSTR]1 Topology change generated on port 1/1/8 on VLAN 1.
May 29, 2020, 15:46		RPVST	Event[5006]LOG_WARN[MSTR]1 Topology change received on port 1/1/7 from source: 38:21:c7:5a:d1:a0 on VLAN 1.
May 29, 2020, 15:46		RPVST	Event[5006]LOG_WARN[MSTR]1 Topology change received on port 1/1/7 from source: 38:21:c7:5a:d1:a0 on VLAN 1.
May 29, 2020, 15:46		RPVST	Event[5006]LOG_WARN[MSTR]1 Topology change received on port 1/1/8 from source: 38:21:c7:5c:d9:a7 on VLAN 1.
May 29, 2020, 15:46		RPVST	Event[5006]LOG_WARN[MSTR]1 Topology change received on port 1/1/8 from source: 38:21:c7:5c:d9:a7 on VLAN 1.
May 29, 2020, 15:46		RPVST	Event[5007]LOG_WARN[MSTR]1 Topology change generated on port 1/1/8 on VLAN 1.
May 29, 2020, 15:46		RPVST	Event[5006]LOG_WARN[MSTR]1 Topology change received on port 1/1/7 from source: 38:21:c7:5a:d1:a0 on VLAN 1.
May 29, 2020, 15:46		RPVST	Event[5006]LOG_WARN[MSTR]1 Topology change received on port 1/1/7 from source: 38:21:c7:5a:d1:a0 on VLAN 1.

Fig. 6.4 Events reflected on central

## ALERT FOR A SWITCH GENERATING MULTIPLE TCN IN SHORT TIME

OPEN ALERTS (6)				ACKNOWLEDGE ALL
OCURRED ON		CATEGORY	SEVERITY	DESCRIPTION
May 29, 2020, 17:28		Switch RPVST Multipl...	warning	NIL_POC: Multiple TCN generated for previous batch of tcen_events on switch: SG9ZKN700Z

Fig. 6.5 Alert for a switch generating multiple tcen in short time

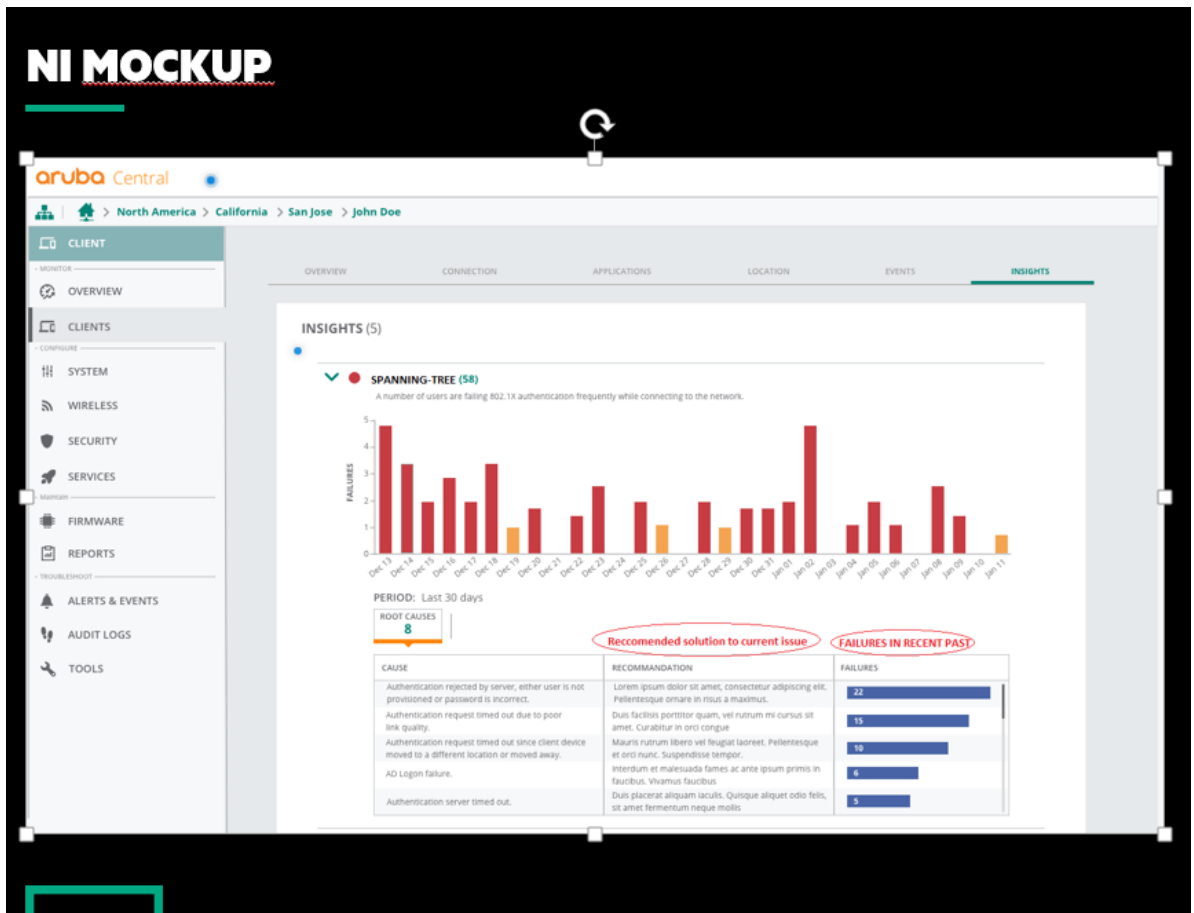


Fig. 6.6 NI Mock up

### III.2.5.3 Monitoring Feature

```
GET /monitor/v2/switch/device_health?stack_id=01008030-e090af80&view=compact&from_timestamp=1538980135273&to_timestamp=1538990935273 304 1.472 ms - -
GET /base_ui/styles/images/icon-health-offline.svg 404 3.108 ms - 1138
GET /monitor/v2/switch/vlan?stack_id=01008030-e090af80&from_timestamp=1538980135273&to_timestamp=1538990935273&metadata=true&limit=50&offset=0 304 1.030 ms - -
GET /monitor/v2/switch/device_health?stack_id=01008030-e090af80&view=compact&from_timestamp=1538980135273&to_timestamp=1538990935273 304 0.936 ms - -
GET /monitor/v2/switch/info?stack_id=01008030-e090af80&from_timestamp=1538980135273&to_timestamp=1538990935273 304 2001.339 ms - -
GET /monitor/v2/switch/faceplate?stack_id=01008030-e090af80&from_timestamp=1538980135273&to_timestamp=1538990935273 304 2.774 ms - -
GET /base_ui/styles/images/icon-health-good.svg 404 2.729 ms - 1138
GET /base_ui/styles/images/icons-tagged.svg 404 1.934 ms - 1138
GET /static/common/img/aruba_central.png 404 2.577 ms - 1138
GET /monitor/v2/switch/info?stack_id=01008030-e090af80&from_timestamp=1538980135273&to_timestamp=1538990935273 304 2002.532 ms - -
GET /monitor/v2/switch/info?stack_id=01008030-e090af80&from_timestamp=1538980135273&to_timestamp=1538990935273 304 2001.379 ms - -
```

Fig. 6.7 GitBash screenshot for UI API

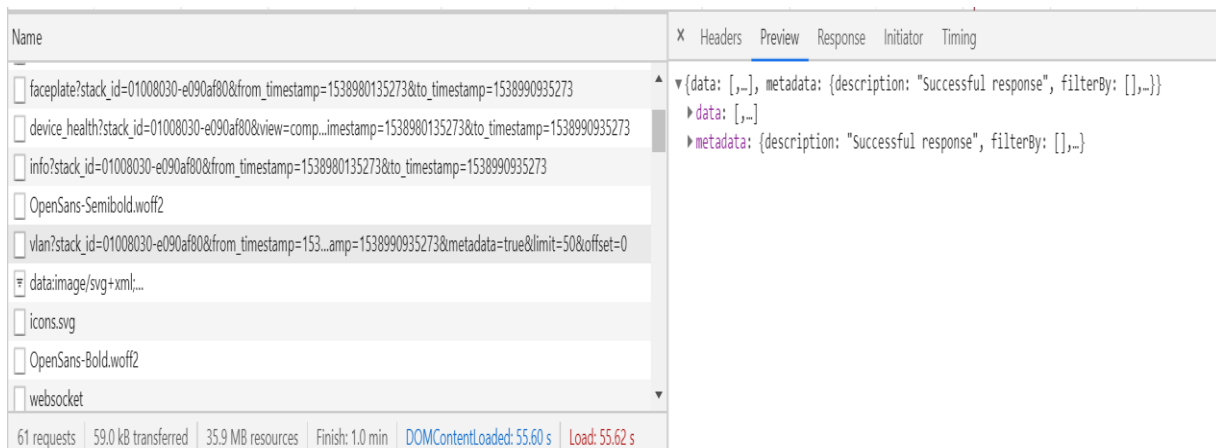


Fig. 6.8 Mock UI for running UI API

```
ubuntu@divyaasw-cl-15821-0:~$ curl -XGET "http://10.164.2.66:9200/_cat/indices"
yellow open central_monitor_vlan                jgEUTcfjQaaFGoxfL6PNlA 3 1 1 0 6.5kb 6.5kb
yellow open central_monitor_notifications-2020_24_171 tV99kn3dSEytEkthS0d7Pg 5 1 1 0 11kb 11kb
yellow open central_tunnel_interface             niqLDXVGQSWdHWPesDV-qA 3 1 0 0 573b 573b
yellow open central_probe_status                 bNfxIZjuSniZnD8vpm9IAA 3 1 0 0 573b 573b
yellow open config_template_variables            3xrVPhpcSNKW541u3EpIQw 3 1 6 0 20.5kb 20.5kb
yellow open central_interface                    VfawY6o8ReWJp045_dWsGg 3 1 64 0 170.3kb 170.3kb
yellow open branch_health-central-lite           k1fwHc6dQMyxqVjm-Dp0zg 3 1 1 0 3.8kb 3.8kb
yellow open notification_definition               okeZyuoKQP-QtaAmLPG1uA 3 1 10 0 55.6kb 55.6kb
yellow open bgp_neighbors                       fmNisaHDR3-w-0Z8qIJPTQ 3 1 0 0 573b 573b
yellow open monitoring_switch_routes             u3m1uLHeQ6Gne68bPJj9_Q 3 1 0 0 573b 573b
yellow open central_monitor_notifications-2020_24_168 6q6_0u-zTqmNMA7ZBwIkHg 5 1 3 0 31.4kb 31.4kb
yellow open central_switch_device_neighbours     D2FFgRrVT4qDwAAIOGqaIA 3 1 0 0 573b 573b
yellow open central_vsx                          fX8qrMULQUuAxXHawGtqsQ 3 1 0 0 573b 573b
yellow open central_monitor_events_switch-2020_24 jzhnFmrsvSsSURSyhjMfkSg 3 1 28 0 66kb 66kb
yellow open apigw-stats-1592179200000           0N7CsbYurQ0u0qpQ5_sx51A 5 1 0 0 955b 955b
yellow open central_tunnel_interface_ap          G5oB5udBTKC21VFI2ooh8g 3 1 0 0 573b 573b
yellow open bgp_rib                             aXjvi_4TT9q0RrUd72Rc-g 3 1 0 0 573b 573b
yellow open central_uplink_interface             M89Lzi61Qfe0ZQGYICyUvg 3 1 0 0 573b 573b
yellow open audit_trail_2020_6                   t5J5jAbWTLm3FdpuXN0Ecw 3 1 93 0 207.3kb 207.3kb
yellow open notification_template                YKFM2hHjTNeo3zh2ftmGSA 3 1 145 0 332kb 332kb
yellow open caas_event_log-2020_24              fR0YqCRaTEehby0YNrMfwQ 3 1 24 0 85.3kb 85.3kb
ubuntu@divyaasw-cl-15821-0:~$ curl -XGET "http://10.164.2.187:9200/central_monitor_vlan/_search" | jq
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 636 100 636 0 0 124k 0 --:--:-- --:--:-- --:--:-- 124k
{
```

Fig. 6.9 The elastic search for VLAN page



## Chapter7: System Testing

---

### Testing is mostly done using NBAPI

List of all the UI-APIs available for Vlan page are:

- /monitor/v2/switch/info?::switch\_id\_key::=:switch\_id\_value::&from\_timestamp=:from\_timestamp::&to\_timestamp=:to\_timestamp::
- /monitor/v2/switch/vlan?::switch\_id\_key::=:switch\_id\_value::&from\_timestamp=:from\_timestamp::&to\_timestamp=:to\_timestamp::&metadata=true
- /monitor/v2/switch/faceplate?::switch\_id\_key::=:switch\_id\_value::&from\_timestamp=:from\_timestamp::&to\_timestamp=:to\_timestamp::
- /monitor/v2/switch/device\_health?::switch\_id\_key::=:switch\_id\_value::&from\_timestamp=:from\_timestamp::&to\_timestamp=:to\_timestamp::

List of all the NBAPIs available for Vlan page are:

- /monitoring/v1/switches/{serial}/vlan
- /monitoring/v1/cx\_switches/{serial}/vlan
- /monitoring/v1/switch\_stacks/{stack\_id}/vlan
- /monitoring/v1/cx\_switch\_stacks/{stack\_id}/vlan

Note: {serial} refers here the serial number of the device that is onboarded on the cl-instance.

```

yellow open caas_event_log-2020_24                                fR0YqCRaTEehby0YNrM
ubuntu@divyaasw-cl-15821-0:~$ curl -XGET "http://10.164.2.187:9200/centra
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  C
          Dload  Upload   Total      Spent    Left   S
100    636    100    636      0      0   124k        0 --:--:-- --:--:-- --:--:--
{
  "took": 2,
  "timed_out": false,
  "_shards": {
    "total": 3,
    "successful": 3,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 1,
    "max_score": 1,
    "hits": [
      {
        "_index": "central_monitor_vlan",
        "_type": "vlan_info",
        "_id": "d5d9e939a64d44f2a24ff7fd9ebd3bb7-SG9ZKN7071",
        "_score": 1,
        "_source": {
          "cid": "d5d9e939a64d44f2a24ff7fd9ebd3bb7",
          "device_id": "SG9ZKN7071",
          "vlan": [
            {
              "id": 1,
              "name": "DEFAULT_VLAN_1",
              "tagged_ports": [],
              "untagged_ports": [],
              "ipaddress": [],
              "access_ports": [],
              "type": 4,
              "is_management_vlan": false,
              "is_voice_enabled": false,
              "is_jumbo_enabled": false,
              "is_igmp_enabled": false,
              "status": 2,
              "oper_state_reason": "NO MEMBER FORWARDING"
            }
          ]
        }
      }
    ]
  }
}

```

Fig. 7 NBAPI for Vlan Info

## Chapter8: Individual Contribution

---

In Rest api testing , setting up of the topology and onboarding the device also looking for attributes in various available tables was my part. After looking for attributes in various tables ,testing them on CLI commands and retrieving them on CURL commands and also modifying the values from CURL and retrieving them on CLI commands was a part of my task and hence , finally documenting them in FG guide so that it can go under review and be verified and merged.

In STP NI , the main task was to first set up the topology locally on the PMG set up and check by flapping the ports that if the syslog events are generating the TCN notifications and with how much time difference. After completing this set up locally, the main task was set up the topology on central and check if the same events are generated on alerts and events page. If so, then the changing the alert settings on Alert/Event Page, by adding the new template and configuring it. Also stating the NI mockup, like how the alert would be generated if any port flaps and also the suggestion to rectify that.

In 2.5.2 MON Feature, there were many modules and those were divided amongst the team. My module was VLANs Page, for this page the overall workflow was to be understood. The UI workflow in frontend to backend, also the databases all were the part of the understanding the module. After that we were allotted with IFDs the internal found defects as JIRA tickets which we needed to resolve, 2 were allotted and resolved by me.

## Chapter9: Conclusion

---

### 9.1. Limitation and Future Scope

1. The Rest Api tested were MSTP and LLDP only for now, the others will also be documented with CURL commands.
2. In LLDP, post commands were still left, due to some UI issues, those will be resolved and further testing on post command will be done.
3. In STP NI, the alert was generated but still the suggestion part is to be done where the user will be suggested what to do if such alert comes.
4. In 2.5.2 MON feature, soon 2.5.3 would be released and all the IFDs and CFDs would be assigned to our team.

## Chapter10: Bibliography







---

- Aruba, a. H. (2018, january). Retrieved from Aruba REST API Guide for ArubaOS-Switch 16.05Part Number: 5200-4219aPublished: January 2018Edition: 2:  
[https://support.hpe.com/hpesc/public/docDisplay?docId=a00040422en\\_us](https://support.hpe.com/hpesc/public/docDisplay?docId=a00040422en_us)
- LP., H. P. (2018, April). Retrieved from ArubaOS 8.3.0.x:  
[https://www.arubanetworks.com/techdocs/ArubaOS\\_83x\\_Web\\_Help/Content/PDFs/ArubaOS%208.3.0.x%20API%20Guide.pdf](https://www.arubanetworks.com/techdocs/ArubaOS_83x_Web_Help/Content/PDFs/ArubaOS%208.3.0.x%20API%20Guide.pdf)
- MATEI, A. (2014, Decemeber 3). *How to test a REST api from command line with curl*. Retrieved from CodepediaOrg: <https://www.codepedia.org/ama/how-to-test-a-rest-api-from-command-line-with-curl/>

## Document Information

<b>Analyzed document</b>	Divya_Aswani_160328_Major_Report_ - divya aswani.pdf (D76011375)
<b>Submitted</b>	7/4/2020 7:08:00 PM
<b>Submitted by</b>	hitesh Jangir
<b>Submitter email</b>	hiteshjangir.cet@modyuniversity.ac.in
<b>Similarity</b>	13%
<b>Analysis address</b>	hiteshjangir.cet.modyun@analysis.arkund.com

## Sources included in the report

<b>W</b>	URL: <a href="https://netbergtw.com/wp-content/uploads/Files/OPS_user_guide.pdf">https://netbergtw.com/wp-content/uploads/Files/OPS_user_guide.pdf</a> Fetched: 7/4/2020 7:10:00 PM		<b>6</b>
<b>W</b>	URL: <a href="https://www.arubanetworks.com/techdocs/ArubaOS_83x_Web_Help/Content/PDFs/ArubaOS%20...">https://www.arubanetworks.com/techdocs/ArubaOS_83x_Web_Help/Content/PDFs/ArubaOS%20...</a> Fetched: 7/4/2020 7:10:00 PM		<b>4</b>
<b>SA</b>	URL: Meenal_Major_160409 Report - meenal jain.docx Fetched: 7/4/2020 7:08:00 PM		<b>1</b>
<b>W</b>	URL: <a href="https://support.hpe.com/hpsc/doc/public/display?docId=emr_na-a00042834en_us&amp;docLoc...">https://support.hpe.com/hpsc/doc/public/display?docId=emr_na-a00042834en_us&amp;docLoc...</a> Fetched: 11/7/2019 7:51:16 PM		<b>8</b>
<b>W</b>	URL: <a href="https://techhub.hpe.com/eginfolib/networking/docs/switches/RA/15-18/5998-8155_ra-2...">https://techhub.hpe.com/eginfolib/networking/docs/switches/RA/15-18/5998-8155_ra-2 ...</a> Fetched: 7/4/2020 7:10:00 PM		<b>1</b>
<b>W</b>	URL: <a href="https://www.codepedia.org/ama/how-to-test-a-rest-api-from-command-line-with-curl/">https://www.codepedia.org/ama/how-to-test-a-rest-api-from-command-line-with-curl/</a> Fetched: 7/4/2020 7:10:00 PM		<b>1</b>