Key Performance Indicators (KPI's) for Network Automation

Introduction

A Crosswork Health Insights KPI (simply known as KPI hereafter) is a structure that can capture key device and network health metrics. The application provides out-of-box KPIs that users can start using with minimum configuration. The power of the application comes from the ability to create custom KPIs, define custom alerts and optionally remediate triggered anomalies automatically.
This lab will help you get an overview of the out-of-box Crosswork Health Insights KPI's using Application programming Interface (known as API's here after) requests.

Objective

The objective of the lab is to provide you a sense of familiarity with the usage of API's requests for Cisco Crosswork Applications of Health Insights and Change Automation.
This exercise helps you to get an idea of what are the parameters which define the key performance indicators (KPI's). This section is an informational section for the developers who want to develop their own KPI's. We would be using a collection of API requests to help us provide detailed insights about the KPI's.

Lab Environment

This lab is based on dCloud environment. Each student has their own unique pod access information and will be shared [if any questions, please reach out to the administration].

Exercise 1

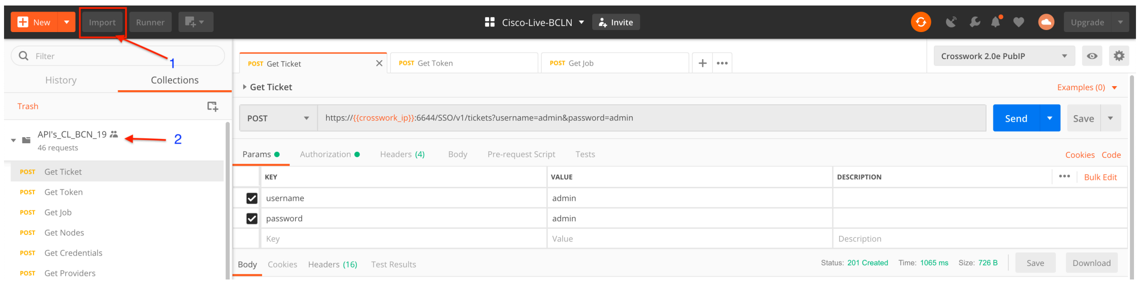
1. If you don't have Postman already installed, you can download it from here. Once you install it, follow the steps below to import the collections and environment:
2. Click on Import, browse to the location where you cloned this repo and add the two files:
    a. *"pub-ip-env.json"*
    b. *"API's_CL_BCN_19.postman_collection.json"*
3. Make sure you select the "**Crosswork 2.0e PubIP"** environment
4. Expand the Imported collection
5. Start making API requests

Exercise 2

Information about KPI's

This section helps you to understand the details of KPI's.

1. From the desktop, launch **Postman** application.
2. Click on Import on the left-hand corner and import the collection **"API's_CL_BCN_19"**



API's_CL_BCN_19
46 requests

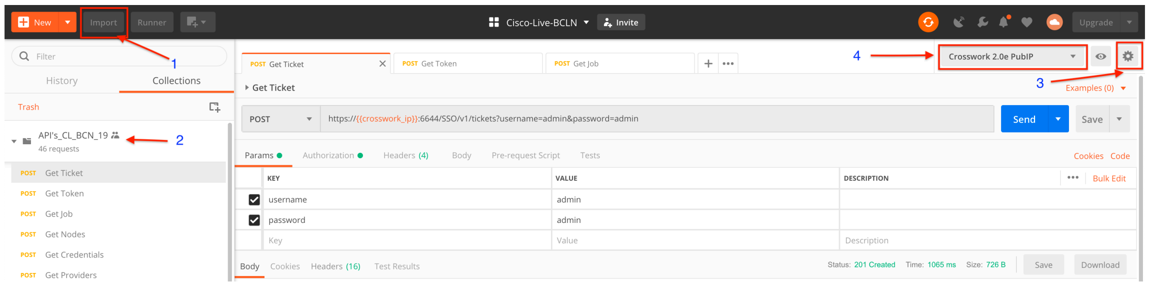
- POST Get Ticket
- POST Get Token
- POST Get Job
- POST Get Nodes
- POST Get Credentials
- POST Get Providers

POST https://{{crosswork_ip}}:6644/SSO/v1/tickets?username=admin&password=admin

KEY	VALUE	DESCRIPTION
✓ username	admin	
✓ password	admin	
Key	Value	Description

Status: 201 Created Time: 1065 ms Size: 726 B

3. Click on setting symbol on the right-hand corner and import the environment **“Crosswork-NSO 2.0e PubIP”**
   a. On the right extreme corner of the application, from the drop-down please select “**Crosswork 2.0e PubIP”** environment



API's_CL_BCN_19
46 requests

- POST Get Ticket
- POST Get Token
- POST Get Job
- POST Get Nodes
- POST Get Credentials
- POST Get Providers

POST https://{{crosswork_ip}}:6644/SSO/v1/tickets?username=admin&password=admin

KEY	VALUE	DESCRIPTION
✓ username	admin	
✓ password	admin	
Key	Value	Description

Status: 201 Created Time: 1065 ms Size: 726 B

4. Update the Postman **Environment** with the new ticket value. Click the **eye** icon in the top right-hand corner of the window. Please update “**any_user and any_password**” under the environment.
5. Expand the collection “**API’s_CL_BCN_19”**
   a. Select the **Get Ticket** request, and then click **Send**. After you get a reply back, copy the reply to the clipboard by clicking the copy icon.
   b. Update the Postman **Environment** with the new ticket value. Click the **eye** icon in the top right-hand corner of the window. Then double-click next to the edit Pen icon in the **Current Value** field for **ticket**
   c. Place the cursor in the **ticket > Current Value** field and press **CTRL + V** to paste the new ticket value into the ticket field. Press **Enter** and then click **outside** the pop-up to close the dialog box.
   d. With our ticket, we can now request a token. Select the **Get Token** command and click **Send**. Click the **copy** icon to copy the response to the clipboard.
   e. Click the **Eye** icon in the upper right-hand corner of the screen. Place your cursor in the **token > CURRENT VALUE** field and click **CTRL + V** to paste the token value into the field. Press **Enter** and click outside the pop-up to close the dialog box.
6. Next, we want to check the job queue. Select **Get Job** and click **Send**. Looking at the reply, we see that there is one job, and that it is in state **5** (completed).
7. Select **Get Credentials** and click **Send**. Reply provides us the information about the profiles created on the Crosswork environment

8. Select **Get Nodes** and click **Send**. Reply provides us the information about the nodes which have been on-boarded on the Crosswork environment.
9. Select **Get Providers** and click **Send**. Reply provides us the information about the provider which has been on-boarded on the Crosswork environment.
10. Select ***<KPI of your choice from the collection list>*** and click **Send**. Reply provides us the information about the parameters of the respective KPI.
11. Repeat the **STEP 10** to run various API requests to understand the different KPI's which are available as a part of crosswork.

<*Look at the below example to understand the output of KPI in a detailed manger*>

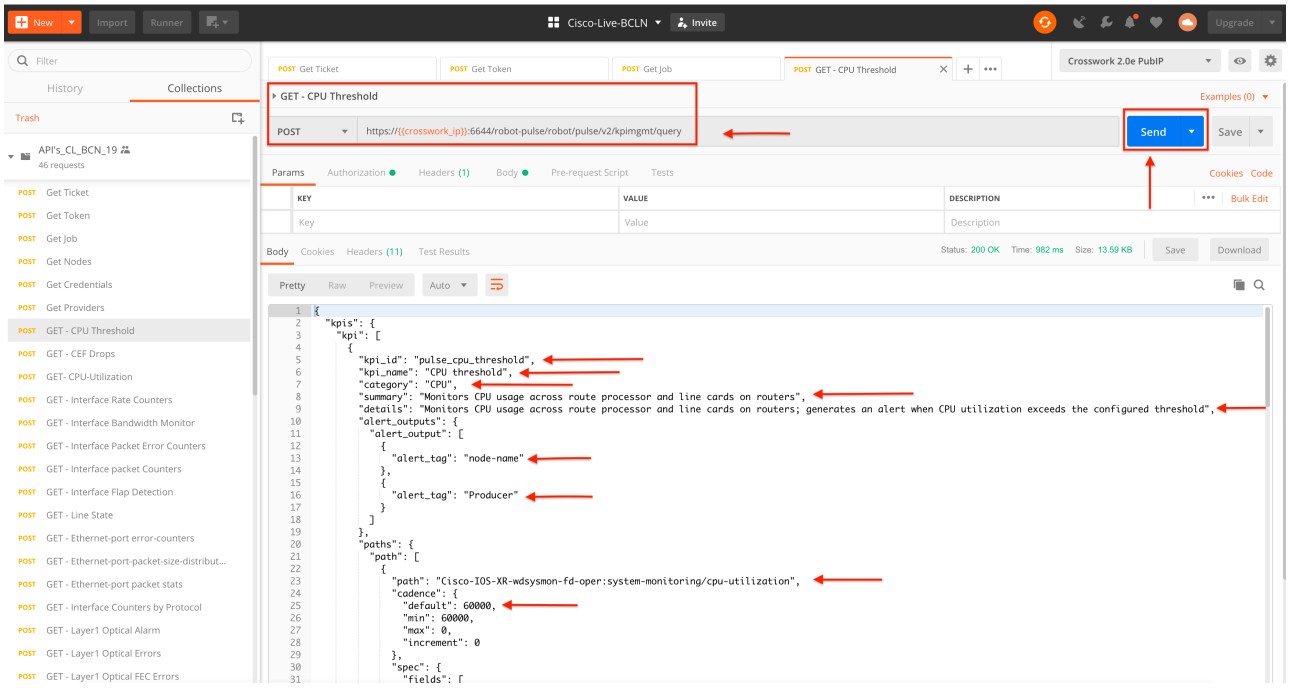
Example to Understand the output
Select the request **"GET CPU Threshold"** and click "**SEND**"
When this request is executed, the output provides us a detailed information about the CPU Threshold KPI that has been enabled on the respective device.

This KPI is defined to report data and generate an alert whenever the threshold value of the CPU exceeds a pre-set value and is cleared whenever the threshold value is below the predefined value for defined time interval under the parameter section of the KPI. Below is an explanation of the values which are defined under the hood of the KPI.

Name	**Current Value**	**Explanation**	**Modification**
KPI_ID	pulse_cpu_threshold	Internal Call	No
KPI_Name	CPU threshold	User Defined	Yes
Category	CPU	Used to segregate KPI's	Yes
Summary	Monitors CPU usage across route processor and line cards on routers	one-line summary of what the user wants the KPI to track	Yes
Details	Monitors CPU usage across route processor and line cards on routers; generates an alert when CPU utilization exceeds the configured threshold	Detailed Description about the KPI and its functions	Yes
Alert	node-name	Defined under Yang file	No, system generated
	Producer	Name of the device where the alert is generated	No, system generated
Path	Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization	YANG path that represents the device or network health	No

		metric to be tracked and it's a **container**. Contents of this container are definitions of data collection points	
Cadence	60000	Data is collected and reported every 60s	Yes
Script	Auto-Generated	Auto-Generated	No
Parameters	Name 1. Threshold → Alert is generated when it crosses this value 2. Clear Time → Alert is Cleared when the value stays below the threshold value for the defined time interval		Yes
	Type 1. Float 2. String		
	Value 1. **String** → Major/Minor/Critical/Warning 2. **Float** → Numeric Value (<100)		
	Display_name	This is for the user to identify the alert and to be displayed on the dashboard.	
	Possible Values 1. **String** → Major/Minor/Critical/Warning 2. **Float** → Numeric Value (<100)	This is a drop down depending upon the selection of the value field.	
Sensor Type	YANG_MDT	Telemetry is being used to track the data	Yes

Screen-shot of the respective output on Postman



GET - CPU Threshold

POST https://{{crosswork_ip}}:6644/robot-pulse/robot/pulse/v2/kpimgmt/query

Status: 200 OK Time: 982 ms Size: 13.59 KB

```
{
  "kpis": {
    "kpi": [
      {
        "kpi_id": "pulse_cpu_threshold",
        "kpi_name": "CPU threshold",
        "category": "CPU",
        "summary": "Monitors CPU usage across route processor and line cards on routers",
        "details": "Monitors CPU usage across route processor and line cards on routers; generates an alert when CPU utilization exceeds the configured threshold",
        "alert_outputs": {
          "alert_output": [
            {
              "alert_tag": "node-name"
            },
            {
              "alert_tag": "Producer"
            }
          ]
        },
        "paths": {
          "path": [
            {
              "path": "Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization",
              "cadence": {
                "default": 60000,
                "min": 60000,
                "max": 0,
                "increment": 0
              },
              "spec": {
                "fields": [
```