# LTRSPG-3918

# A Cisco Solution for Multivendor Telemetry Collection

Krishnan Thirukonda - Technical Marketing Engineer
Nicola Martino - Technical Marketing Engineer

# Index

## Learning Objectives

Upon completion of this lab, you will be able to:

- Familiarize with Crosswork Data Gateway GUI
- Understand supported Crosswork Data Gateway collection protocols for Cisco and third-party vendors
- Being able to use Crosswork to create collection jobs via its GUI and visualize collected KPIs on Grafana
- Being able to use Postman to create collection jobs leveraging Crosswork API
- Understand how to generate JSON payloads to create collection jobs
- Understand Crosswork Data Gateway message definition and be able to distinguish, within Data Gateway messages, the actual device payload from the message header

## Scenario

In this lab, you will practice with Telemetry collection capabilities of Crosswork Data Gateway. Crosswork Data Gateway is an essential component of a larger solution called Crosswork Network Controller, where it plays a centralized collector function for real time data collection for multi-vendor network devices. Data collection can use different methods like MDT, gNMI, SNMP, CLI and so on. Data collection also extends to TRAP, SYSLOG and EDT network generated events. Collected data is then securely delivered and consumed by the Crosswork application running on the Crosswork Network Controller solution. Collected data can also be delivered to external customer managed messaging buses, either Kafka or gRPC servers where, then, customer target applications can connect and retrieve data from.

This lab will be focusing on this latest case where Crosswork Data Gateway collects telemetry data from the network and send it to an external Kafka server. To that end, students will use Crosswork API to create collection jobs on Crosswork Data Gateway.  Lab will cover same use case while leveraging Crosswork UI rather than its API, to achieve same target (this will be limited to CLI and SNMP collection protocols). At the end of this lab, students will also be familiar with Crosswork Data Gateway protobuf message structure and will be able to read it throughout all its fields.

## About Crosswork Network Controller Solution

Crosswork Data Gateway is part of a larger Cisco solution called Crosswork Network Controller. The solution allows customers to deploy services faster and enables intent-based network optimization mitigating network congestion issues. Cisco Crosswork Network Controller has been shaped by the experience of helping multiple tier-one service provider and large enterprise customers automate everything from simple device turn-up to sophisticated full lifecycle service management. Cisco's Network Controller combines intent-based network automation to deliver critical capabilities for service orchestration and fulfilment, network optimization, service path computation, device deployment and management, and fault remediation.

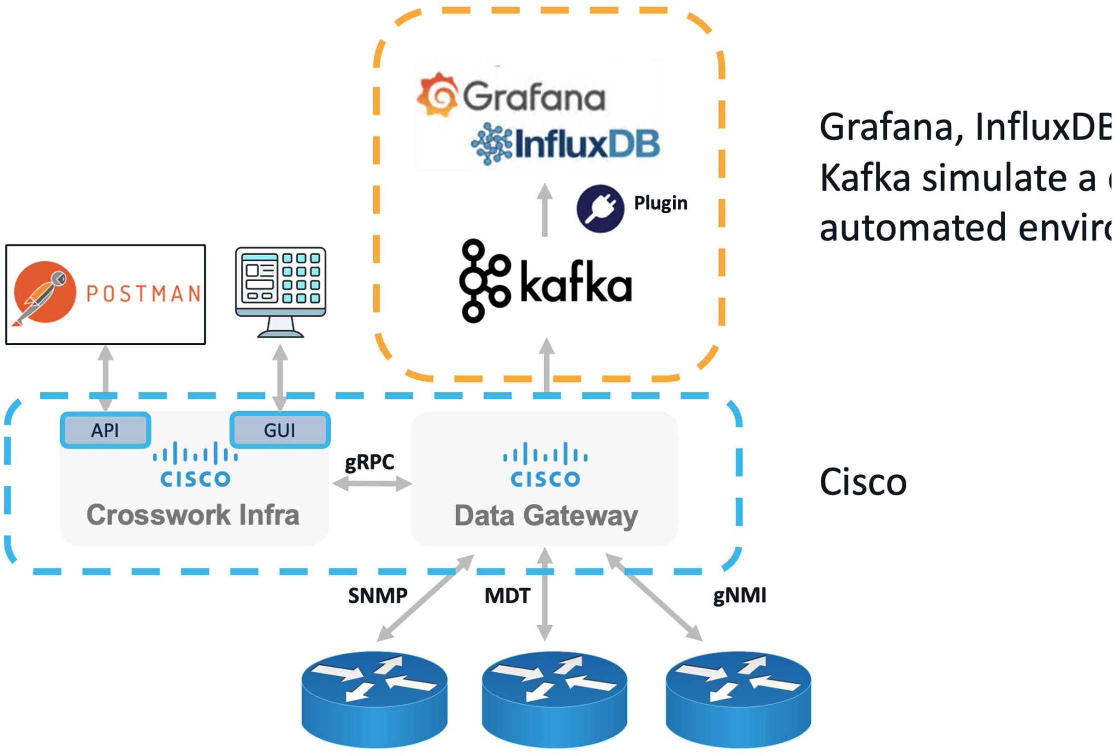
Cisco Crosswork Network Controller offers service providers a turnkey network automation solution that delivers increased service agility, cost efficiency, and optimization for faster time to customer value, improved end-user service experience and lower operating cost.

Cisco Crosswork Network Controller is a microservices-based platform. The data collection functionality has been separated out into its own VM and into its own software package called Cisco Crosswork Data Gateway. Cisco Crosswork Data Gateway gathers all the information from the monitored devices and forwards it to Crosswork for analysis and processing. Crosswork Data Gateway can also be used to send collected data to external customer-managed messaging buses such as Kafka or gRPC servers.

Cisco Crosswork Data Gateway offloads voluminous data collection closer to the devices while allowing target applications to scale independently. It enables service providers to quickly program and deploy collectors and offers central visibility into services collecting data and the type of data being collected.

The solution also uses Cisco Network Service Orchestrator (NSO) to deploy configuration changes or create new

## Lab Components



## Component Details

- Crosswork Platform Infrastructure 4.1
- Crosswork Data Gateway 3.0
- Grafana 7.5.5
- InfluxDB 1.8.5
- NSO 5.5.2.9
- VIRL 1.6
- Cisco XRv 9000 rel. 7.3.2
- Cisco CRS16S rel. 6.6.3
- Juniper vMX 21.1R1.11
- TraGenSer 5.1
- Demo Portal 2.0

## Disclaimer

This lab is intended to familiarize with Crosswork Network Controller and Data Gateway. Although part of this lab, the external Kafka, Grafana, InfluxDB and other software components, are not delivered by Cisco as part of the product. Only target for those components is to simulate a customer automated environment where Grafana represents a potential target application consuming data collected by Crosswork Data Gateway.

Other Crosswork Applications leverages their internal Kafka and directly consume collected data (with a different set of use cases), but this is not covered in this lab

## Equipment/Application Details

| Application | | | | |
|---|---|---|---|---|
| dCloud Automation UI | Pod for troubleshooting | http://198.18.133.1:8000/ | N/A | N/A |
| Crosswork UI | Crosswork Portal | https://198.18.134.219:30603 | admin | C!sco12345 |
| Crosswork SSH | Crosswork VM | 198.18.134.219 | cw-admin | cRo55work! |
| CDG SSH | CDG VM | 198.18.134.225 | dg-admin | cRo55work! |
| Grafana UI | Grafana Portal | http://198.18.134.26:3001 | admin | admin |
| Demo Portal | Demo Portal | http://198.18.134.26:7979/index.html | Not required | |
| App VM | VM for python consumers | ssh cisco@198.18.134.26 -p 20022 | cisco | cisco |
| Cisco nodes TELNET | Node-1 to Node-8 | 198.19.1.1-8 | cisco | cisco |
| vMX node TELNET | vmx99 | 198.19.1.99 | cisco | cisco |
| NSO UI | Web UI for NSO | http://198.18.134.28:8080 | admin | admin |
| NSO CLI | Command line interface for NSO | ssh admin@198.18.134.28 -p 2024 | admin | Admin |
| NSO Host | NSO Host container | ssh cisco@198.18.134.28 | cisco | C1sco12345 |

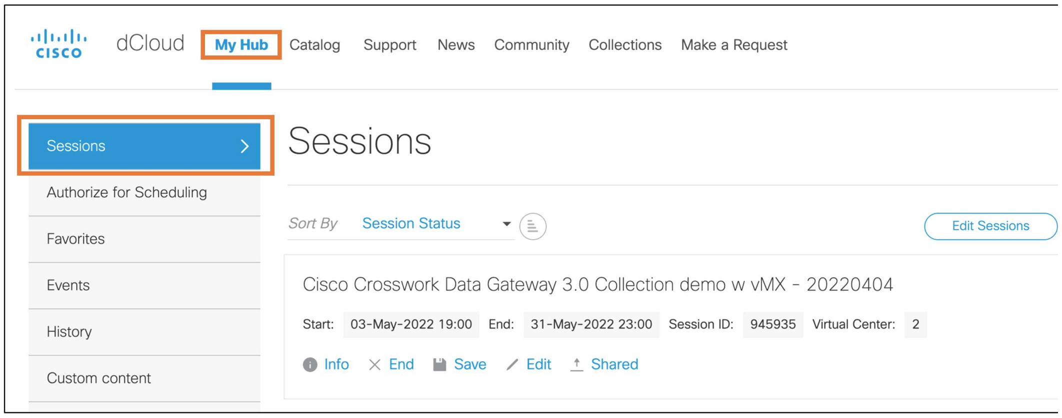
## Get Started

Labs run on Cisco dCloud infrastructure on Cisco Data Centers distributed around the globe. Every user will have a session assigned.

There are two options to access the lab:

- (Preferred) Connect from the dCloud portal on https://dcloud.cisco.com using your CCO account. You might need to switch to a different data center depending on which one has been assigned to you.

  Under **My Hub** -> **Sessions**, select **View**

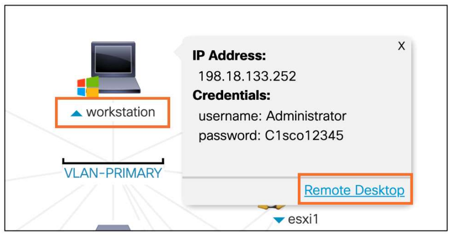


Sessions

Cisco Crosswork Data Gateway 3.0 Collection demo w vMX - 20220404

Start: 03-May-2022 19:00 End: 31-May-2022 23:00 Session ID: 945935 Virtual Center: 2

  Then select **Workstation** -> **Remote Desktop**



**IP Address:**
198.18.133.252
**Credentials:**
username: Administrator
password: C1sco12345

- Directly VPN to the dCloud lab session via Cisco AnyConnect, then use your Windows Remote Desktop client (RDP) using following IP and credentials

- IP: **198.18.133.252**, username: **administrator**, password: **C1sco12345**

In both cases you will reach the remote desktop where everything you need to execute the lab has been preconfigured

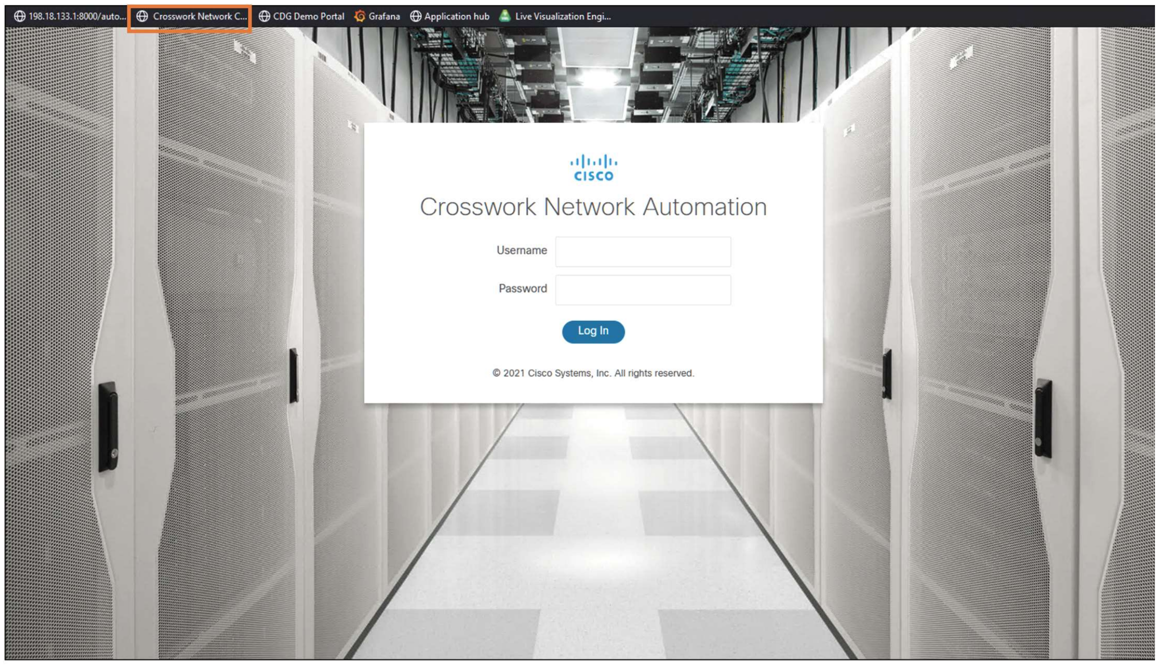
## Task 1: Verify Setup

In this task, user will review and verify the Crosswork and Grafana setup are correct and prepare the lab environment to execute next scenarios properly. At the same time, user will familiarize with Crosswork GUI and its components.

As mentioned before, Data Gateway instances, devices, credentials, and provider profiles have been added by demo-automation. This configuration must be in the correct state before we continue with subsequent tasks.

### Step 1

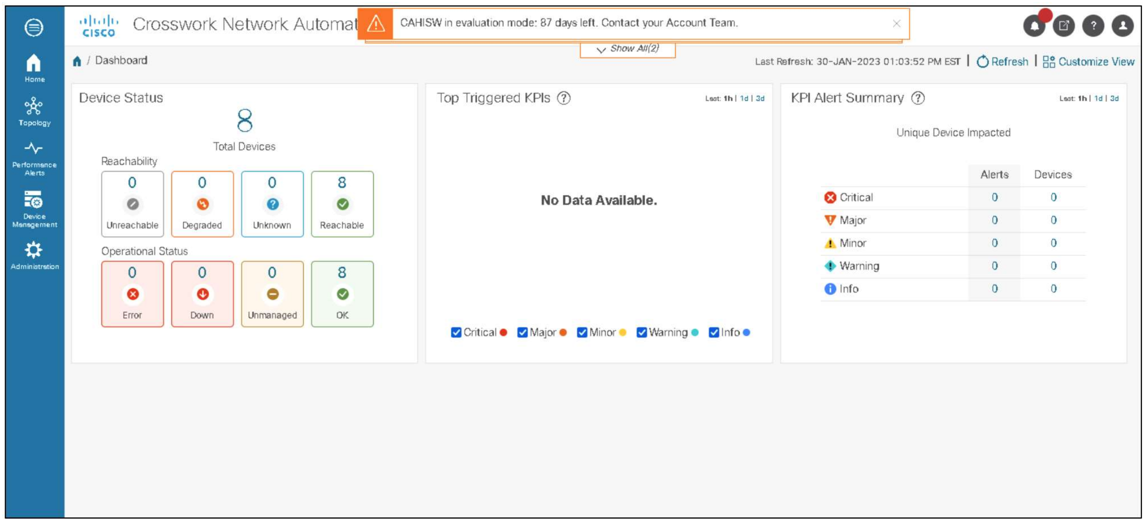
In your workstation browser, select the **Crosswork** shortcut in the browser bookmarks bar, or browse to https://crosswork.demo.dcloud.cisco.com:30603. Enter username **admin**, password **C!sco12345** and click **Log In** (username and password might have been already populated for you).


198.18.133.1:8000/auto...
Crosswork Network C...
CDG Demo Portal
Grafana
Application hub
Live Visualization Engi...
cisco
Crosswork Network Automation
Username
Password
Log In
© 2021 Cisco Systems, Inc. All rights reserved.

## Step 2

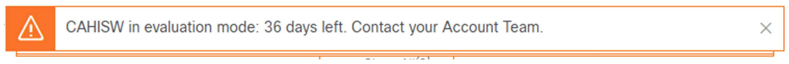
You land on the main Crosswork Network Controller dashboard. Here we have a consolidated view on the status of what the Crosswork solution is managing.

**NOTE**: What you see here in terms of dashboard content and navigation menu, depends on the solution components being installed. What you will see in your lab might be different as different components might have been installed. The one required to execute this lab are included.



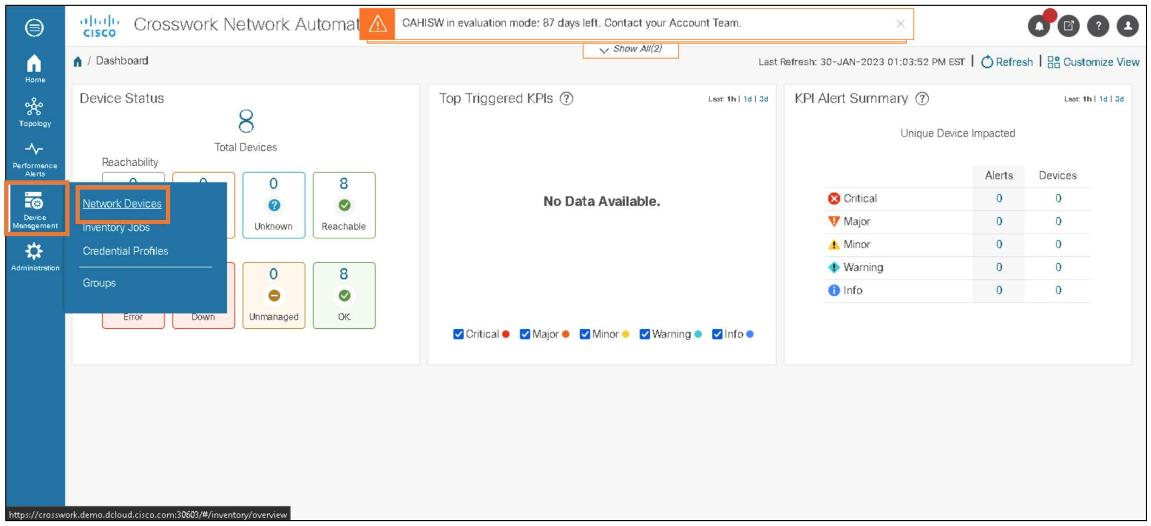
| | Alerts | Devices |
|---|---|---|
| Critical | 0 | 0 |
| Major | 0 | 0 |
| Minor | 0 | 0 |
| Warning | 0 | 0 |
| Info | 0 | 0 |

The Crosswork Network Controller uses Cisco Smart licensing, and a reminder is displayed at top of the page. The user may choose to remove it, clicking the X, but it will return for the next screen. This is no impact on functionality or our demo. There is a 90 day evaluation period with full functionality.


CAHISW in evaluation mode: 36 days left. Contact your Account Team.

## Step 3

In the left-hand pane, navigate to **Device Management > Network Devices**.

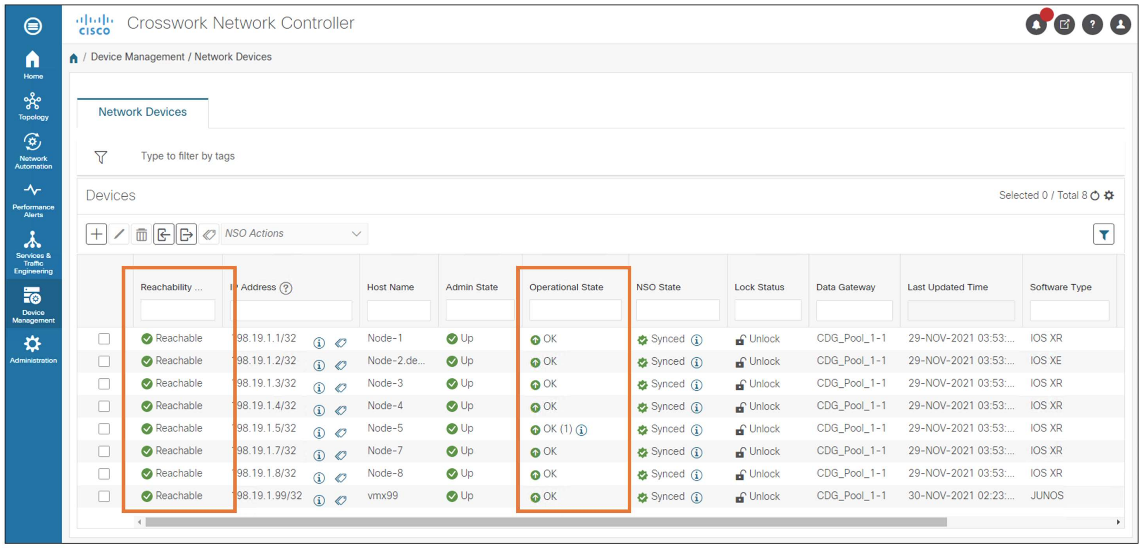


CAHISW in evaluation mode: 87 days left. Contact your Account Team.

| | Alerts | Devices |
|---|---|---|
| Critical | 0 | 0 |
| Major | 0 | 0 |
| Minor | 0 | 0 |
| Warning | 0 | 0 |
| Info | 0 | 0 |

## Step 4

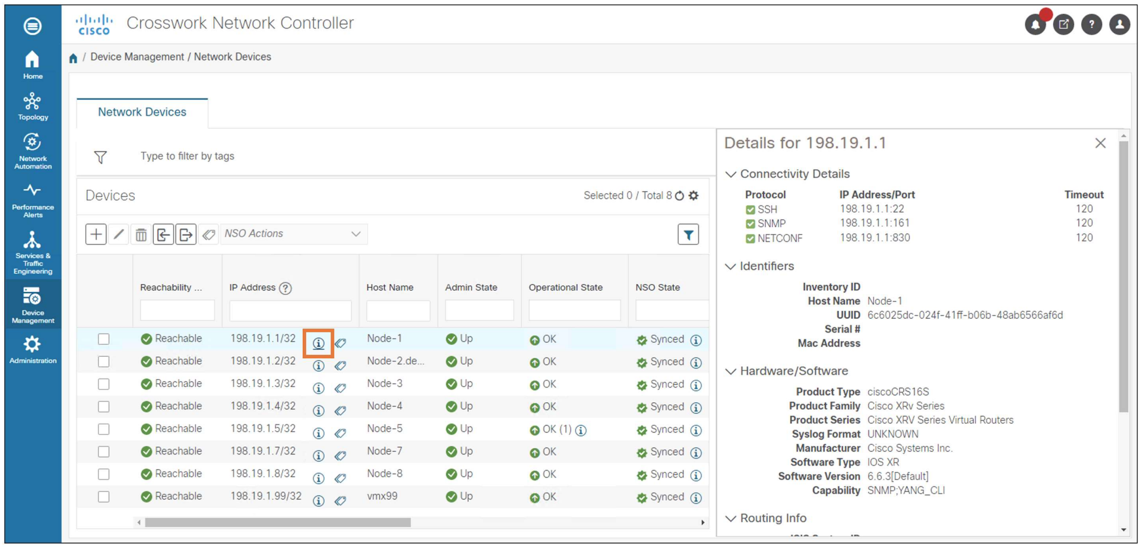
All devices in the table must have a **Reachability State** and an **Operational State** of green.



| Reachability ... | IP Address | Host Name | Admin State | Operational State | NSO State | Lock Status | Data Gateway | Last Updated Time | Software Type |
|---|---|---|---|---|---|---|---|---|---|
| Reachable | 98.19.1.1/32 | Node-1 | Up | OK | Synced | Unlock | CDG_Pool_1-1 | 29-NOV-2021 03:53:... | IOS XR |
| Reachable | 98.19.1.2/32 | Node-2.de... | Up | OK | Synced | Unlock | CDG_Pool_1-1 | 29-NOV-2021 03:53:... | IOS XE |
| Reachable | 98.19.1.3/32 | Node-3 | Up | OK | Synced | Unlock | CDG_Pool_1-1 | 29-NOV-2021 03:53:... | IOS XR |
| Reachable | 98.19.1.4/32 | Node-4 | Up | OK | Synced | Unlock | CDG_Pool_1-1 | 29-NOV-2021 03:53:... | IOS XR |
| Reachable | 98.19.1.5/32 | Node-5 | Up | OK (1) | Synced | Unlock | CDG_Pool_1-1 | 29-NOV-2021 03:53:... | IOS XR |
| Reachable | 98.19.1.7/32 | Node-7 | Up | OK | Synced | Unlock | CDG_Pool_1-1 | 29-NOV-2021 03:53:... | IOS XR |
| Reachable | 98.19.1.8/32 | Node-8 | Up | OK | Synced | Unlock | CDG_Pool_1-1 | 29-NOV-2021 03:53:... | IOS XR |
| Reachable | 98.19.1.99/32 | vmx99 | Up | OK | Synced | Unlock | CDG_Pool_1-1 | 30-NOV-2021 02:23:... | JUNOS |

## Step 5

You can click the “**i**” icon close to the device IP to get more information about each device.



Devices

Selected 0 / Total 8

| | Reachability ... | IP Address | | Host Name | Admin State | Operational State | NSO State |
|---|---|---|---|---|---|---|---|
| ☐ | Reachable | 198.19.1.1/32 | | Node-1 | Up | OK | Synced |
| ☐ | Reachable | 198.19.1.2/32 | | Node-2.de... | Up | OK | Synced |
| ☐ | Reachable | 198.19.1.3/32 | | Node-3 | Up | OK | Synced |
| ☐ | Reachable | 198.19.1.4/32 | | Node-4 | Up | OK | Synced |
| ☐ | Reachable | 198.19.1.5/32 | | Node-5 | Up | OK (1) | Synced |
| ☐ | Reachable | 198.19.1.7/32 | | Node-7 | Up | OK | Synced |
| ☐ | Reachable | 198.19.1.8/32 | | Node-8 | Up | OK | Synced |
| ☐ | Reachable | 198.19.1.99/32 | | vmx99 | Up | OK | Synced |

Details for 198.19.1.1

Connectivity Details

| Protocol | IP Address/Port | Timeout |
|---|---|---|
| ✓ SSH | 198.19.1.1:22 | 120 |
| ✓ SNMP | 198.19.1.1:161 | 120 |
| ✓ NETCONF | 198.19.1.1:830 | 120 |

Identifiers

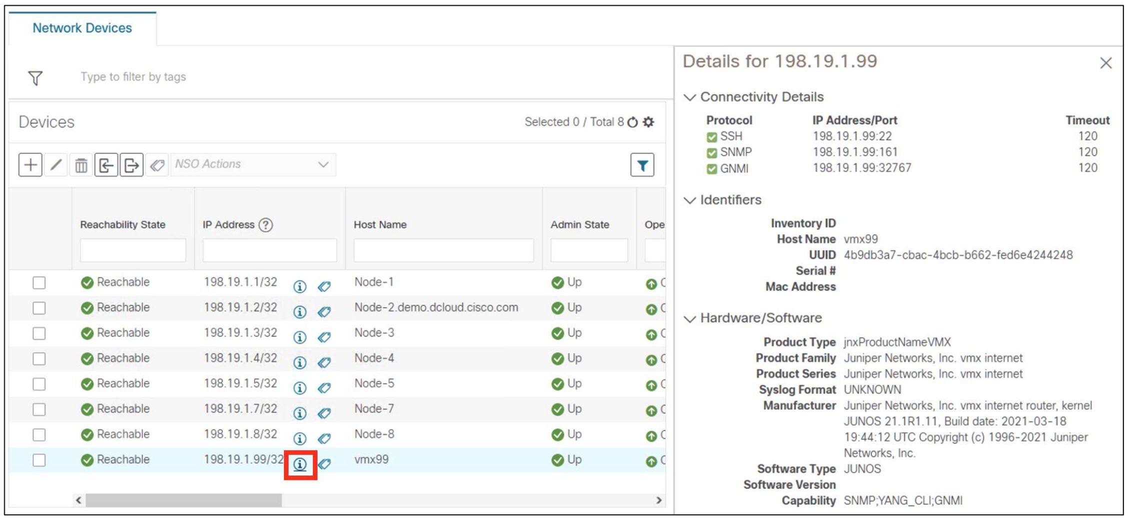
| | |
|---|---|
| **Inventory ID** | |
| **Host Name** | Node-1 |
| **UUID** | 6c6025dc-024f-41ff-b06b-48ab6566af6d |
| **Serial #** | |
| **Mac Address** | |

Hardware/Software

| | |
|---|---|
| **Product Type** | ciscoCRS16S |
| **Product Family** | Cisco XRv Series |
| **Product Series** | Cisco XRV Series Virtual Routers |
| **Syslog Format** | UNKNOWN |
| **Manufacturer** | Cisco Systems Inc. |
| **Software Type** | IOS XR |
| **Software Version** | 6.6.3[Default] |
| **Capability** | SNMP;YANG_CLI |

Routing Info

(Optional) You can do same for the Juniper vmx99 and see modelling is similar to the Cisco nodes



Network Devices

Type to filter by tags

Devices — Selected 0 / Total 8

NSO Actions

| | Reachability State | IP Address | Host Name | Admin State |
|---|---|---|---|---|
| ☐ | Reachable | 198.19.1.1/32 | Node-1 | Up |
| ☐ | Reachable | 198.19.1.2/32 | Node-2.demo.dcloud.cisco.com | Up |
| ☐ | Reachable | 198.19.1.3/32 | Node-3 | Up |
| ☐ | Reachable | 198.19.1.4/32 | Node-4 | Up |
| ☐ | Reachable | 198.19.1.5/32 | Node-5 | Up |
| ☐ | Reachable | 198.19.1.7/32 | Node-7 | Up |
| ☐ | Reachable | 198.19.1.8/32 | Node-8 | Up |
| ☐ | Reachable | 198.19.1.99/32 | vmx99 | Up |

Details for 198.19.1.99

Connectivity Details

| Protocol | IP Address/Port | Timeout |
|---|---|---|
| ✓ SSH | 198.19.1.99:22 | 120 |
| ✓ SNMP | 198.19.1.99:161 | 120 |
| ✓ GNMI | 198.19.1.99:32767 | 120 |

Identifiers

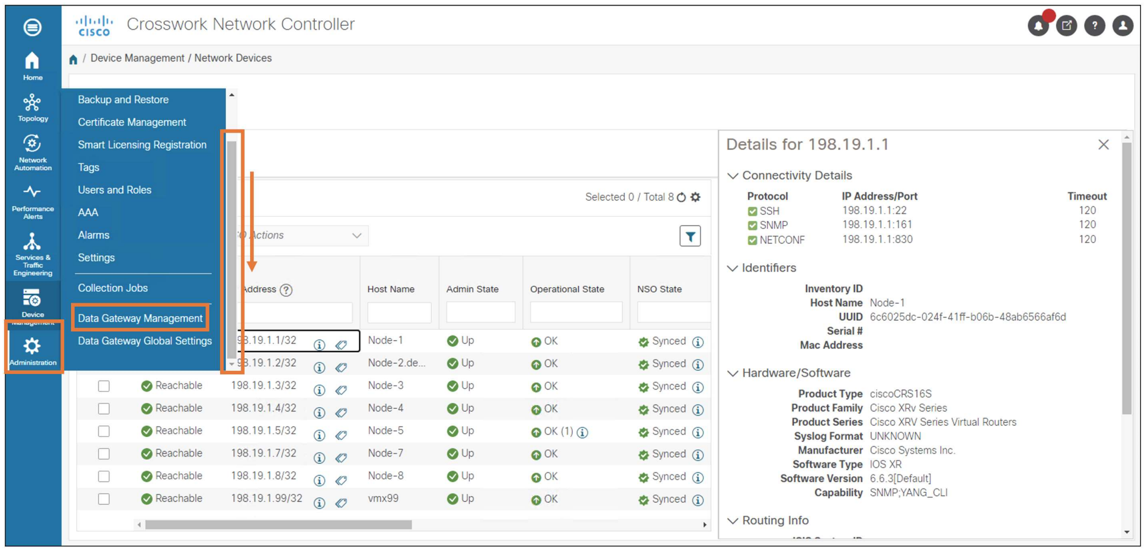
| | |
|---|---|
| Inventory ID | |
| Host Name | vmx99 |
| UUID | 4b9db3a7-cbac-4bcb-b662-fed6e4244248 |
| Serial # | |
| Mac Address | |

Hardware/Software

| | |
|---|---|
| Product Type | jnxProductNameVMX |
| Product Family | Juniper Networks, Inc. vmx internet |
| Product Series | Juniper Networks, Inc. vmx internet |
| Syslog Format | UNKNOWN |
| Manufacturer | Juniper Networks, Inc. vmx internet router, kernel JUNOS 21.1R1.11, Build date: 2021-03-18 19:44:12 UTC Copyright (c) 1996-2021 Juniper Networks, Inc. |
| Software Type | JUNOS |
| Software Version | |
| Capability | SNMP;YANG_CLI;GNMI |

## Step 6

In the left-hand pane, navigate to **Administration > Data Gateway Management**. You might need to slide the right bar down to see the whole navigation menu list.



Device Management / Network Devices

Selected 0 / Total 8

| IP Address | Host Name | Admin State | Operational State | NSO State |
|---|---|---|---|---|
| 198.19.1.1/32 | Node-1 | Up | OK | Synced |
| 198.19.1.2/32 | Node-2.de... | Up | OK | Synced |
| 198.19.1.3/32 | Node-3 | Up | OK | Synced |
| 198.19.1.4/32 | Node-4 | Up | OK | Synced |
| 198.19.1.5/32 | Node-5 | Up | OK (1) | Synced |
| 198.19.1.7/32 | Node-7 | Up | OK | Synced |
| 198.19.1.8/32 | Node-8 | Up | OK | Synced |
| 198.19.1.99/32 | vmx99 | Up | OK | Synced |

Details for 198.19.1.1

Connectivity Details

| Protocol | IP Address/Port | Timeout |
|---|---|---|
| ✓ SSH | 198.19.1.1:22 | 120 |
| ✓ SNMP | 198.19.1.1:161 | 120 |
| ✓ NETCONF | 198.19.1.1:830 | 120 |

Identifiers

| | |
|---|---|
| Inventory ID | |
| Host Name | Node-1 |
| UUID | 6c6025dc-024f-41ff-b06b-48ab6566af6d |
| Serial # | |
| Mac Address | |

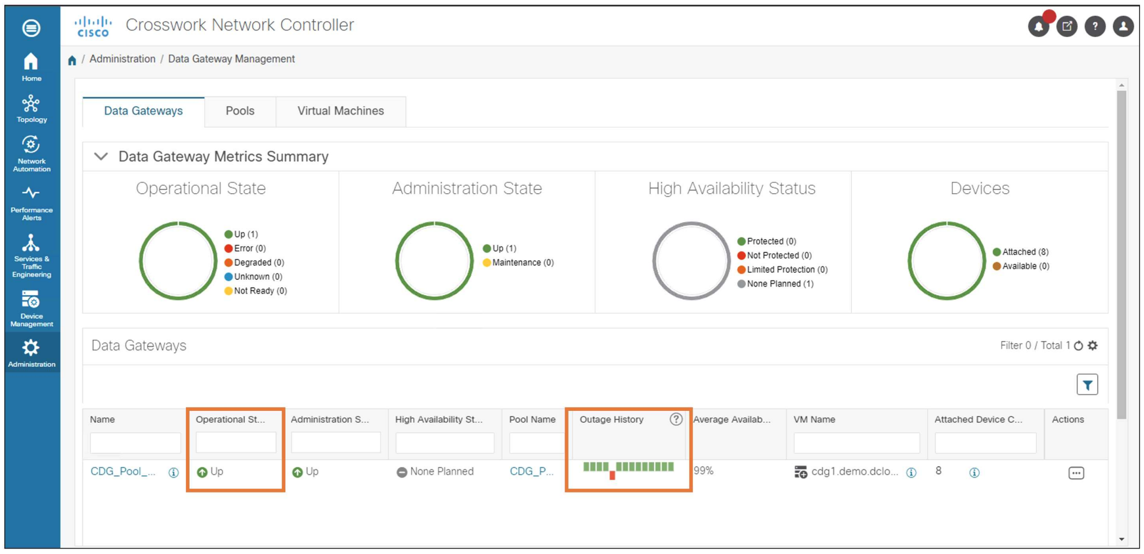
Hardware/Software

| | |
|---|---|
| Product Type | ciscoCRS16S |
| Product Family | Cisco XRv Series |
| Product Series | Cisco XRV Series Virtual Routers |
| Syslog Format | UNKNOWN |
| Manufacturer | Cisco Systems Inc. |
| Software Type | IOS XR |
| Software Version | 6.6.3[Default] |
| Capability | SNMP;YANG_CLI |

Routing Info

## Step 7

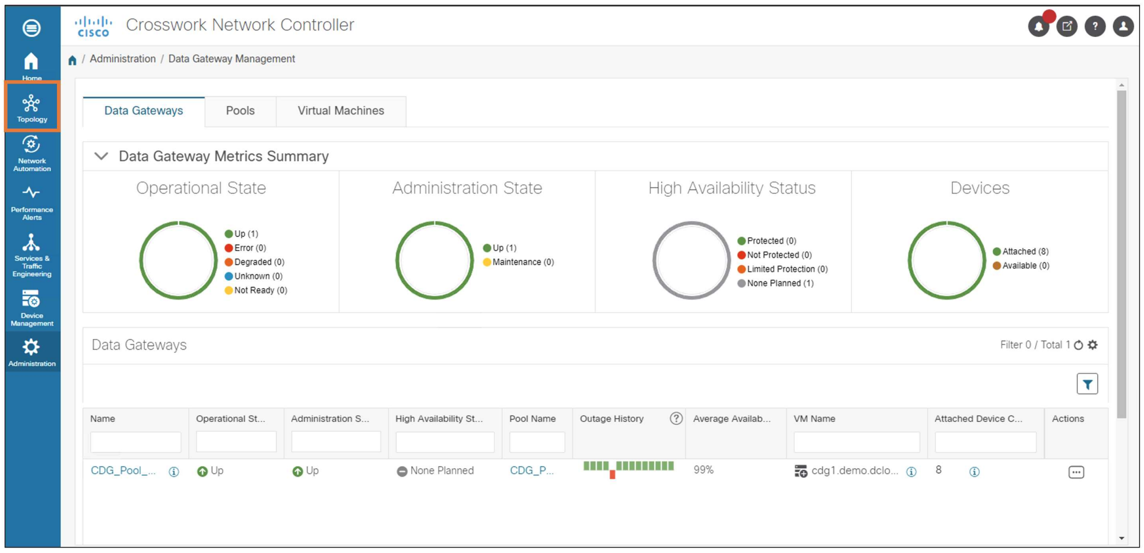
The Data Gateway **Operational State** must be **Up**. You may see some degradation occurred during last 14 days in the Outage History column. That refers to past status only. The only important status is the current one.



| Name | Operational St... | Administration S... | High Availability St... | Pool Name | Outage History | Average Availab... | VM Name | Attached Device C... | Actions |
|---|---|---|---|---|---|---|---|---|---|
| CDG_Pool_... | Up | Up | None Planned | CDG_P... | | 99% | cdg1.demo.dclo... | 8 | |

## Step 8

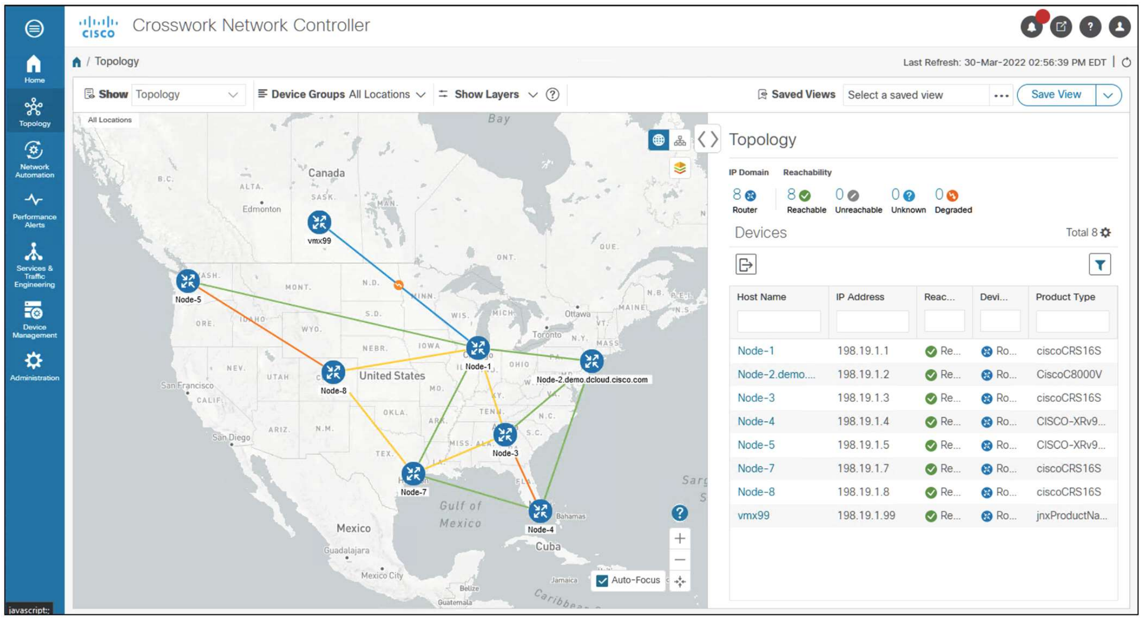
In the left-hand pane, select **Topology**



| Name | Operational St... | Administration S... | High Availability St... | Pool Name | Outage History | Average Availab... | VM Name | Attached Device C... | Actions |
|---|---|---|---|---|---|---|---|---|---|
| CDG_Pool_... | Up | Up | None Planned | CDG_P... | | 99% | cdg1.demo.dclo... | 8 | |

## Step 9

This shows the current network Topology view, with Inventory of devices and device types on the right side



Last Refresh: 30-Mar-2022 02:56:39 PM EDT

Topology

IP Domain: 8 Router

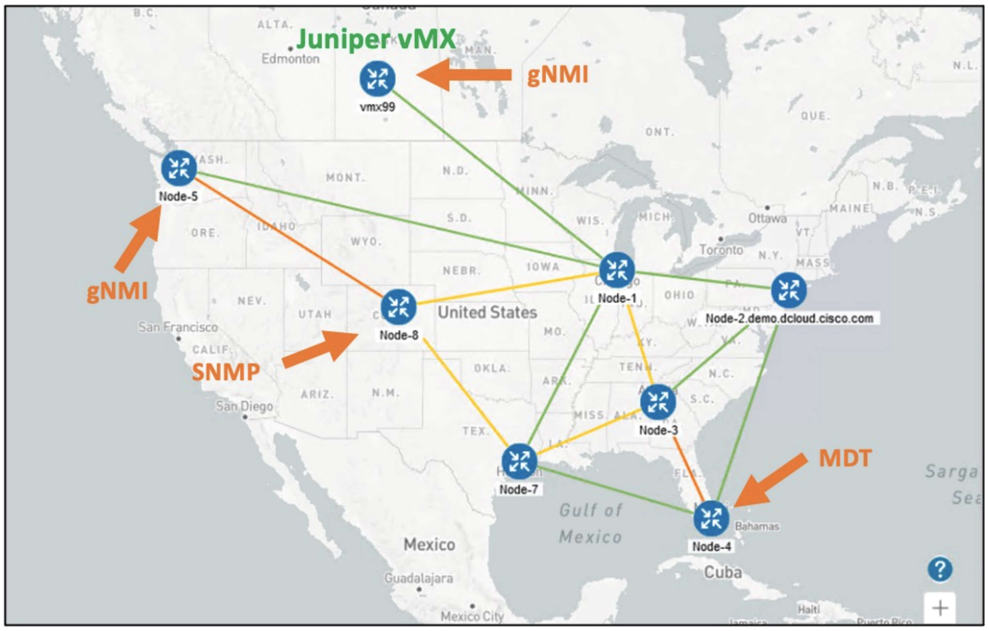
Reachability: 8 Reachable, 0 Unreachable, 0 Unknown, 0 Degraded

Devices — Total 8

| Host Name | IP Address | Reac... | Devi... | Product Type |
|---|---|---|---|---|
| Node-1 | 198.19.1.1 | ✓ Re... | Ro... | ciscoCRS16S |
| Node-2.demo.... | 198.19.1.2 | ✓ Re... | Ro... | CiscoC8000V |
| Node-3 | 198.19.1.3 | ✓ Re... | Ro... | ciscoCRS16S |
| Node-4 | 198.19.1.4 | ✓ Re... | Ro... | CISCO-XRv9... |
| Node-5 | 198.19.1.5 | ✓ Re... | Ro... | CISCO-XRv9... |
| Node-7 | 198.19.1.7 | ✓ Re... | Ro... | ciscoCRS16S |
| Node-8 | 198.19.1.8 | ✓ Re... | Ro... | ciscoCRS16S |
| vmx99 | 198.19.1.99 | ✓ Re... | Ro... | jnxProductNa... |

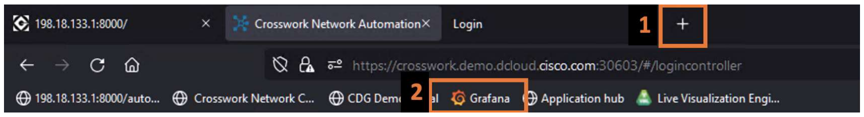
## Step 10

During this lab, we will collect interface statistics from different devices using different collection protocols as shown in following picture:


Juniper vMX
gNMI
vmx99
Node-5
gNMI
Node-8
SNMP
Node-1
Node-2.demo.dcloud.cisco.com
Node-3
Node-7
Node-4
MDT

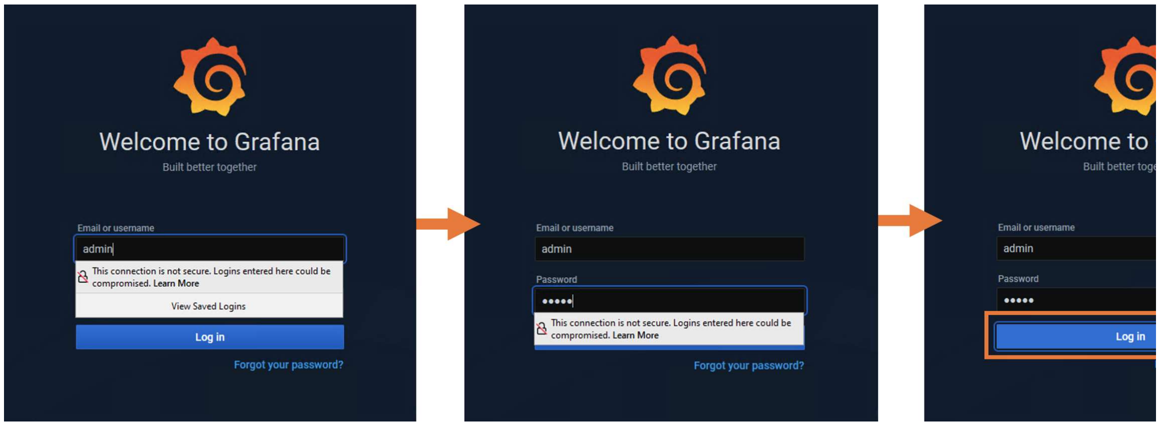
## Step 11

In your workstation browser, open a new tab, then select the **Grafana** shortcut in the browser bookmarks bar, or browse to http://198.18.134.26:3001.

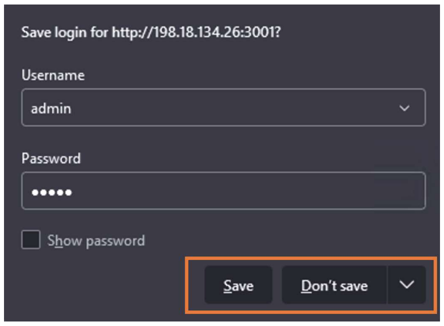

198.18.133.1:8000/
Crosswork Network Automation
Login
1
2
https://crosswork.demo.dcloud.cisco.com:30603/#/logincontroller
198.18.133.1:8000/auto...
Crosswork Network C...
CDG Demo
Grafana
Application hub
Live Visualization Engi...

Login with username **admin**, password **admin**.

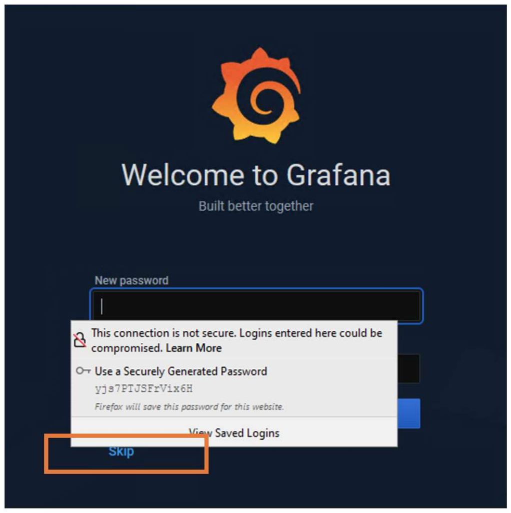
**TIP**: Use **TAB** on the keyboard to move from **username** to **password** to **Login** button



Click on one of the 2 save options (it doesn't really matter)



Then select **Skip** when prompted to change the password.


Welcome to Grafana

Built better together

New password

This connection is not secure. Logins entered here could be compromised. Learn More

Use a Securely Generated Password

yjs7PTJSFrVix6H

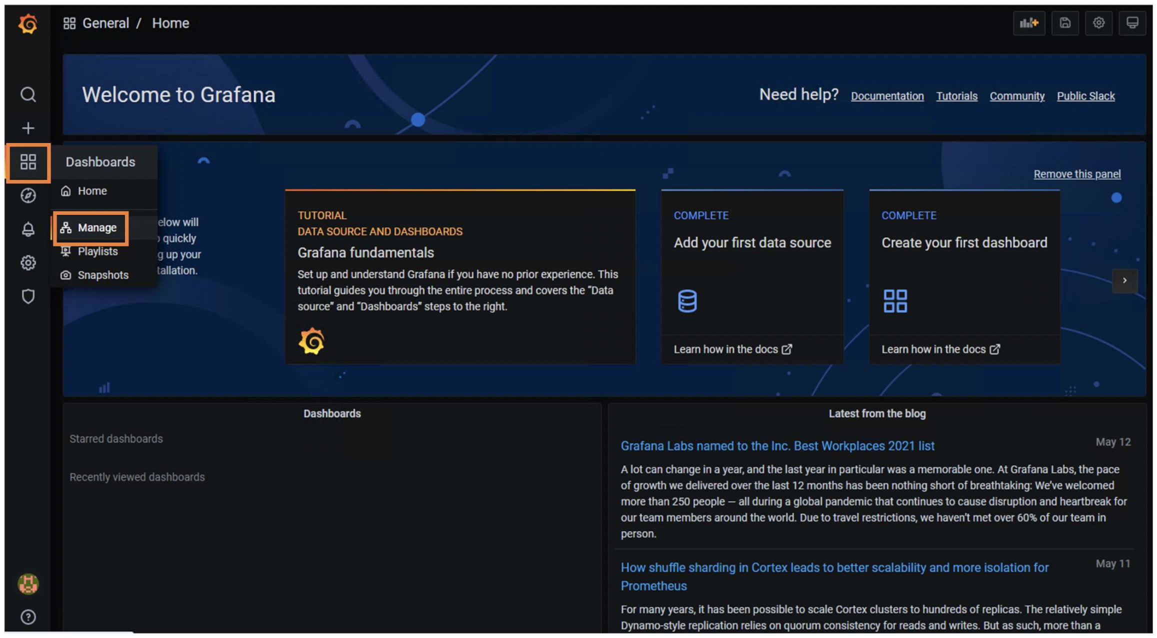
Firefox will save this password for this website.

View Saved Logins

Skip

## Step 12

On the left-hand pane, navigate to **Dashboards -> Manage**


General / Home

Welcome to Grafana

Need help? Documentation Tutorials Community Public Slack

Dashboards

Home

Manage

Playlists

Snapshots

elow will
quickly
g up your
tallation.

Remove this panel

TUTORIAL
DATA SOURCE AND DASHBOARDS

Grafana fundamentals

Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.

COMPLETE

Add your first data source

Learn how in the docs

COMPLETE

Create your first dashboard

Learn how in the docs

Dashboards

Starred dashboards

Recently viewed dashboards

Latest from the blog

Grafana Labs named to the Inc. Best Workplaces 2021 list May 12

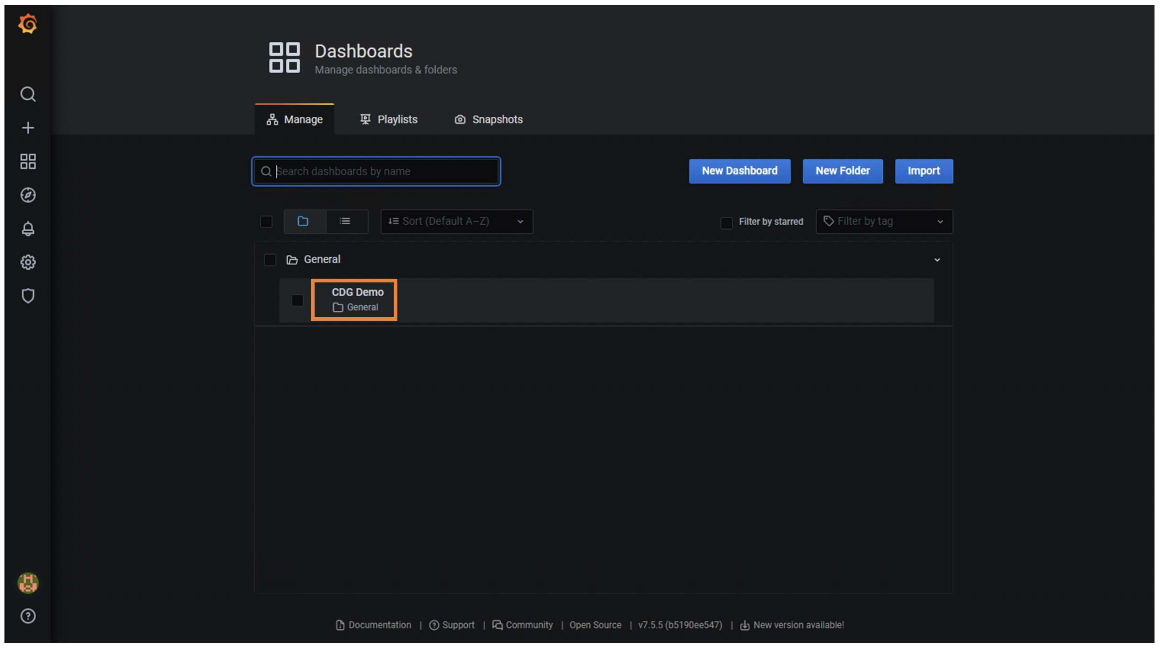
A lot can change in a year, and the last year in particular was a memorable one. At Grafana Labs, the pace of growth we delivered over the last 12 months has been nothing short of breathtaking: We've welcomed more than 250 people — all during a global pandemic that continues to cause disruption and heartbreak for our team members around the world. Due to travel restrictions, we haven't met over 60% of our team in person.

How shuffle sharding in Cortex leads to better scalability and more isolation for Prometheus May 11

For many years, it has been possible to scale Cortex clusters to hundreds of replicas. The relatively simple Dynamo-style replication relies on quorum consistency for reads and writes. But as such, more than a

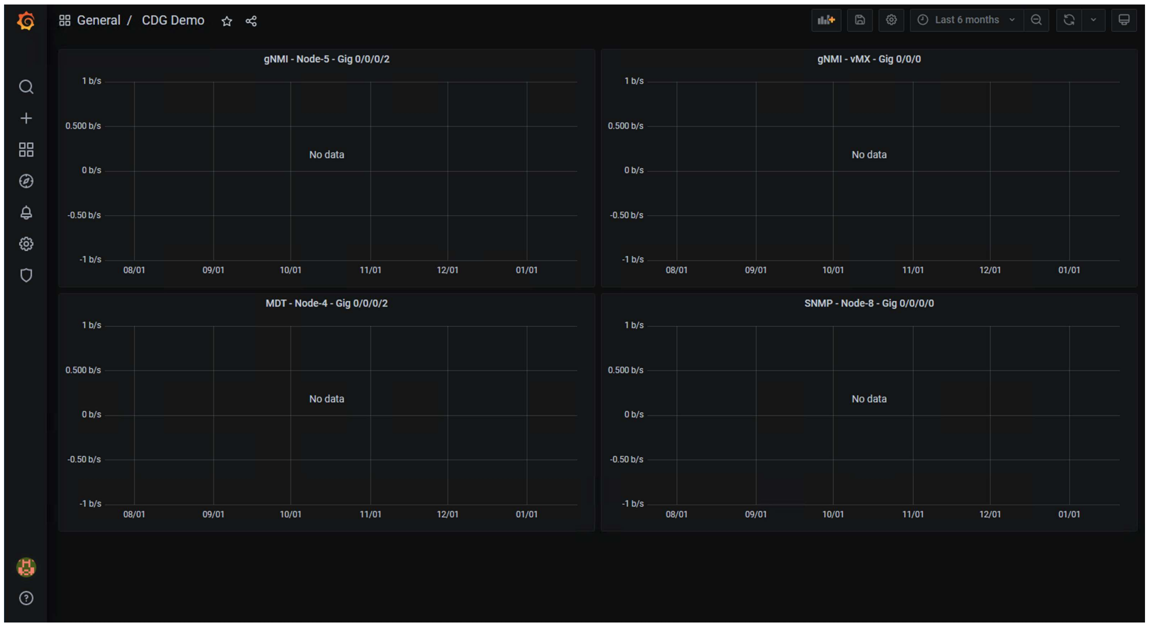
## Step 13

Then select **CDG Demo**


Dashboards
Manage dashboards & folders
Manage
Playlists
Snapshots
Search dashboards by name
New Dashboard
New Folder
Import
Sort (Default A–Z)
Filter by starred
Filter by tag
General
CDG Demo
General
Documentation | Support | Community | Open Source | v7.5.5 (b5190ee547) | New version available!

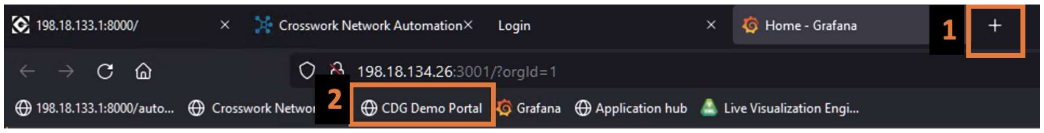
## Step 14

The CDG Demo Dashboard has 4 graphs. All graphs should have no data points at this time.


General / CDG Demo
Last 6 months
gNMI - Node-5 - Gig 0/0/0/2
gNMI - vMX - Gig 0/0/0
MDT - Node-4 - Gig 0/0/0/2
SNMP - Node-8 - Gig 0/0/0/0
No data

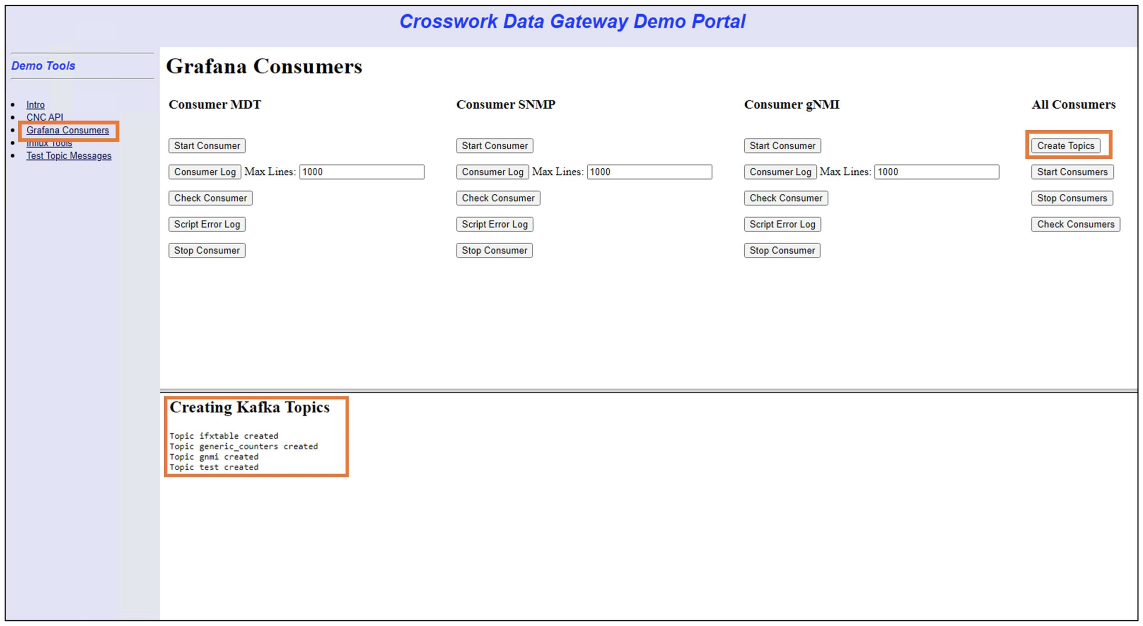
## Step 15

In your workstation browser, open a new tab and select the **CDG Demo Portal** shortcut in the browser bookmarks bar, or browse to http://198.18.134.26:7979. No password is required.


198.18.133.1:8000/
Crosswork Network Automation
Login
Home - Grafana
1
198.18.134.26:3001/?orgId=1
198.18.133.1:8000/auto...
Crosswork Networ
2
CDG Demo Portal
Grafana
Application hub
Live Visualization Engi...

## Step 16

On the left-hand pane, select **Grafana Consumers**, then select **Create Topics** on the right side under **All Consumers**. On the bottom side of the page, verify all Kafka topics get created.

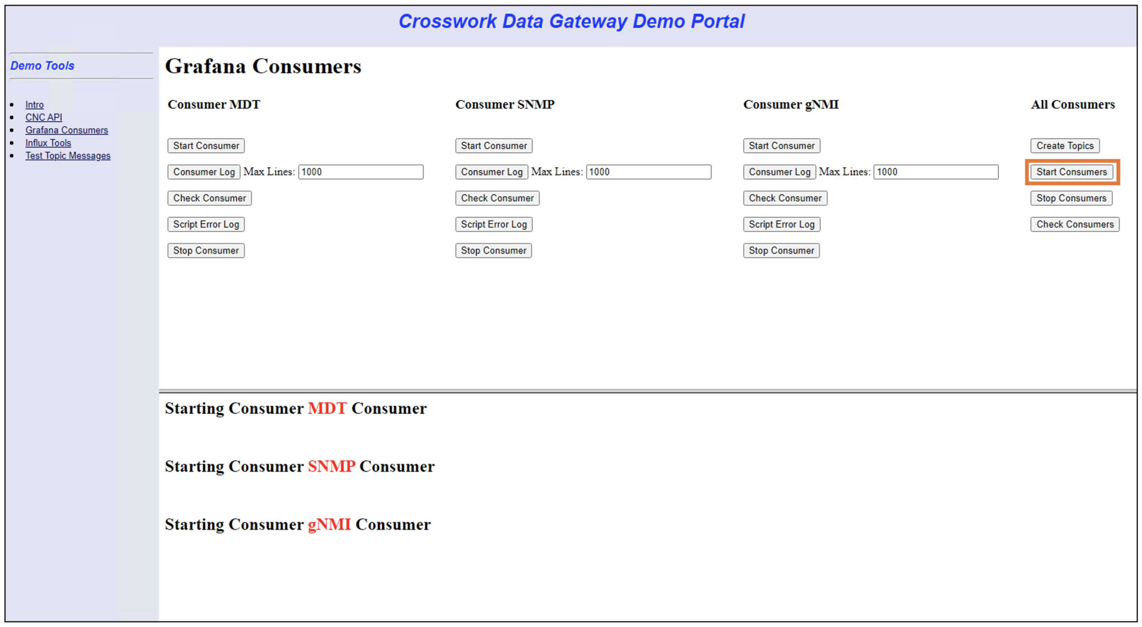


Creating Kafka Topics

```
Topic ifxtable created
Topic generic_counters created
Topic gnmi created
Topic test created
```

## Step 17

Still under **All Consumers**, now select **Start Consumers**



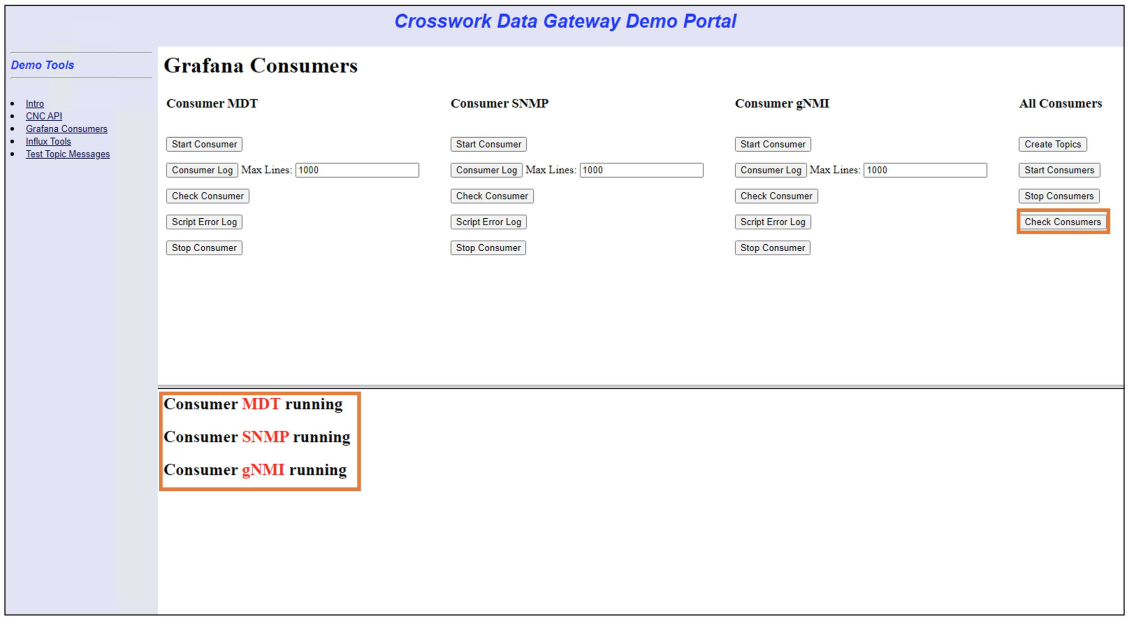
Starting Consumer MDT Consumer

Starting Consumer SNMP Consumer

Starting Consumer gNMI Consumer

## Step 18

Wait 5 seconds, then click **Check Consumers**. All consumers should be reported in the **running** state at the bottom of the page.



Consumer MDT running

Consumer SNMP running

Consumer gNMI running

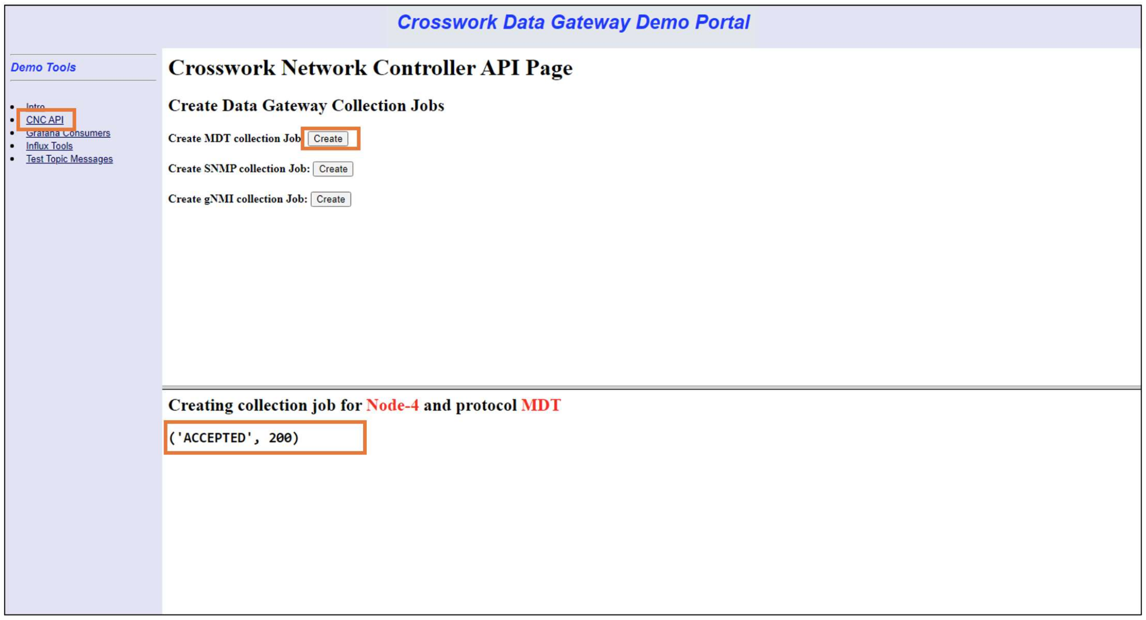
**This concludes task 1.**

## Task 2: Create Collection Jobs using an automation tool

This task focuses on creating collection jobs leveraging Crosswork API in an automated way using an home-made tool with a graphical front end that simulates a potential customer automation environment. Collection jobs will collect interface traffic counters and graphically present them on Grafana, simulating a potential customer target application. Metrics will be same for each node but collected using a different protocol: MDT (via a native YANG data model), SNMP and gMNI (via an OpenConfig YANG data model). Portal will interface (transparently to the user) with Crosswork API to create collection jobs. During next task, user will have more visibility to the underling API call and its payload.

### Step 1

On the left-hand pane in the **CDG Demo Portal**, select **CNC API**. Then click **Create** on the **Create MDT Collection Job**. Verify the UI returns (**ACCEPTED, 200**) in the bottom pane.

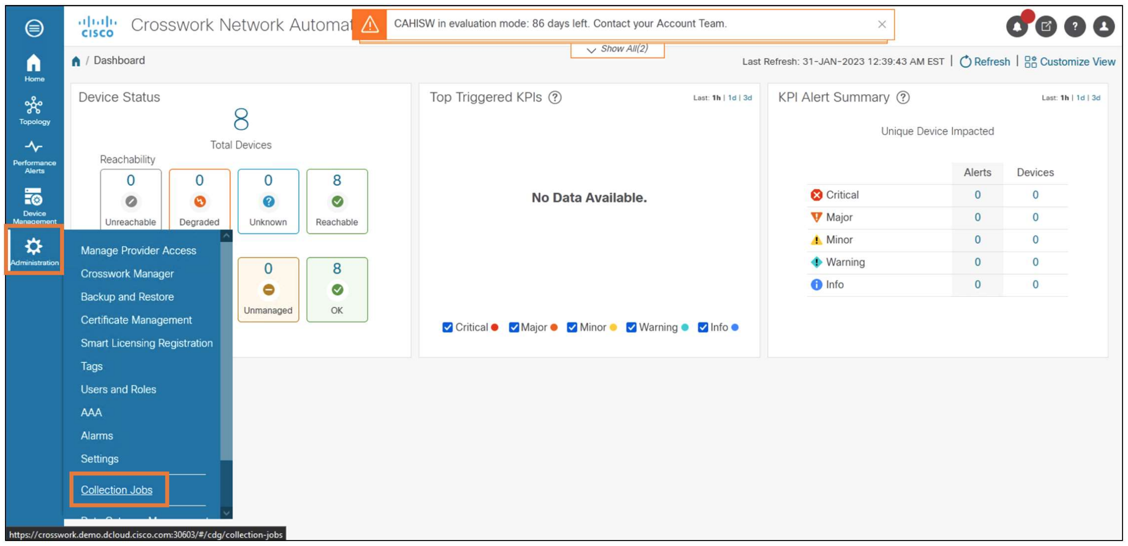

Crosswork Data Gateway Demo Portal
Demo Tools
Intro
CNC API
Grafana Consumers
Influx Tools
Test Topic Messages
Crosswork Network Controller API Page
Create Data Gateway Collection Jobs
Create MDT collection Job: Create
Create SNMP collection Job: Create
Create gNMI collection Job: Create

Creating collection job for Node-4 and protocol MDT

```
('ACCEPTED', 200)
```

## Step 2

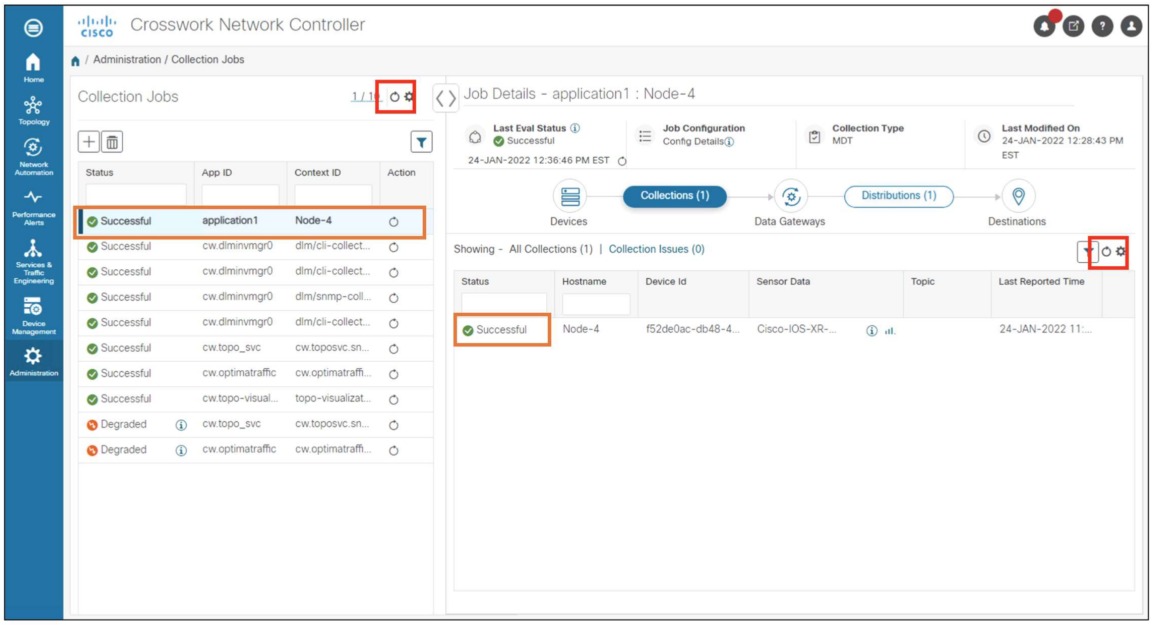
Navigate back to Crosswork (if necessary, log in again with username **admin** and password **C!sco12345**) and navigate to **Administration > Collection Jobs** to check the collection job we just created



|  | Alerts | Devices |
|---|---|---|
| Critical | 0 | 0 |
| Major | 0 | 0 |
| Minor | 0 | 0 |
| Warning | 0 | 0 |
| Info | 0 | 0 |

## Step 3

The table on the left side of the window lists all system defined collection jobs and the user defined collection job just created. Our collection job will have **application1** in the **App ID** column and **Node-4** in the **Context ID** column. Select that entry fin the Collection Job list (You may have to click on the refresh button in the left pane to see the new collection job in the list).



| Status | App ID | Context ID | Action |
|---|---|---|---|
| Successful | application1 | Node-4 | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.dlminvmgr0 | dlm/snmp-coll... | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.topo_svc | cw.toposvc.sn... | |
| Successful | cw.optimatraffic | cw.optimatraffi... | |
| Successful | cw.topo-visual... | topo-visualizat... | |
| Degraded | cw.topo_svc | cw.toposvc.sn... | |
| Degraded | cw.optimatraffic | cw.optimatraffi... | |

Job Details - application1 : Node-4

Last Eval Status: Successful
24-JAN-2022 12:36:46 PM EST
Job Configuration: Config Details
Collection Type: MDT
Last Modified On: 24-JAN-2022 12:28:43 PM EST

Showing - All Collections (1) | Collection Issues (0)

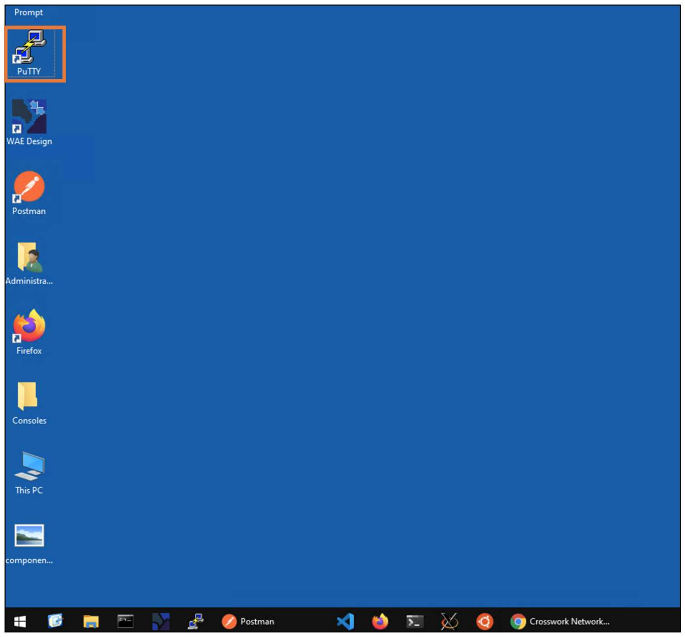
| Status | Hostname | Device Id | Sensor Data | Topic | Last Reported Time |
|---|---|---|---|---|---|
| Successful | Node-4 | f52de0ac-db48-4... | Cisco-IOS-XR-... | | 24-JAN-2022 11:... |

The Job detail on the right side of the window will show the status for each device involved in the collection job. In this case **Node-4** only (if you check other collection jobs, you will see more devices). The collection **Status** should be reported as **Successful**.

Note that being an MDT collection job, it might take some time to move to a Successful state as it will require NSO to create required MDT configuration on **Node-4**. In this case, reload the job list using the refresh button highlighted in red untill state changes to **Successful**. Note that the Collection Jobs list on the left and the Job Details on the right won't refresh automatically and have different refresh buttons.

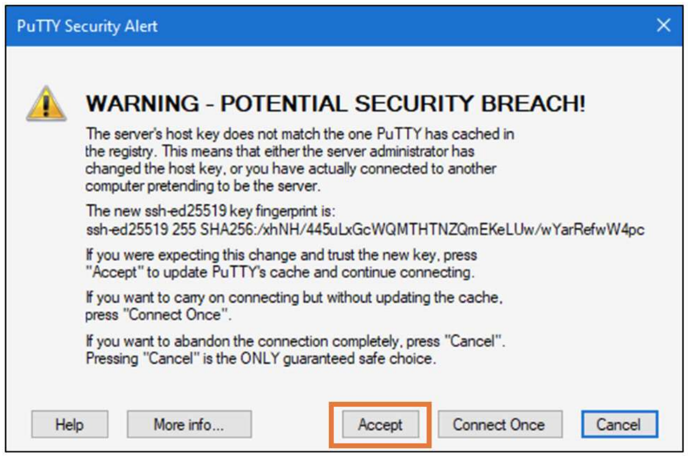
## Step 4

On your workstation desktop, launch the **PuTTY** shortcut, and login using **IP 198.19.1.4**, username **cisco**, password **cisco**



## Step 5

Click on **Accept** if warned


PuTTY Security Alert

WARNING - POTENTIAL SECURITY BREACH!

The server's host key does not match the one PuTTY has cached in the registry. This means that either the server administrator has changed the host key, or you have actually connected to another computer pretending to be the server.

The new ssh-ed25519 key fingerprint is:
ssh-ed25519 255 SHA256:/xhNH/445uLxGcWQMTHTNZQmEKeLUw/wYarRefwW4pc

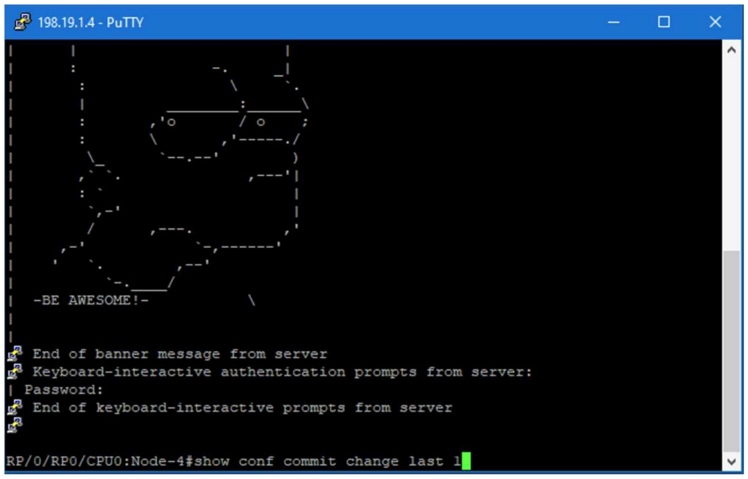
If you were expecting this change and trust the new key, press "Accept" to update PuTTY's cache and continue connecting.

If you want to carry on connecting but without updating the cache, press "Connect Once".

If you want to abandon the connection completely, press "Cancel". Pressing "Cancel" is the ONLY guaranteed safe choice.

## Step 6

Issue the command **show configuration commit change last 1**.



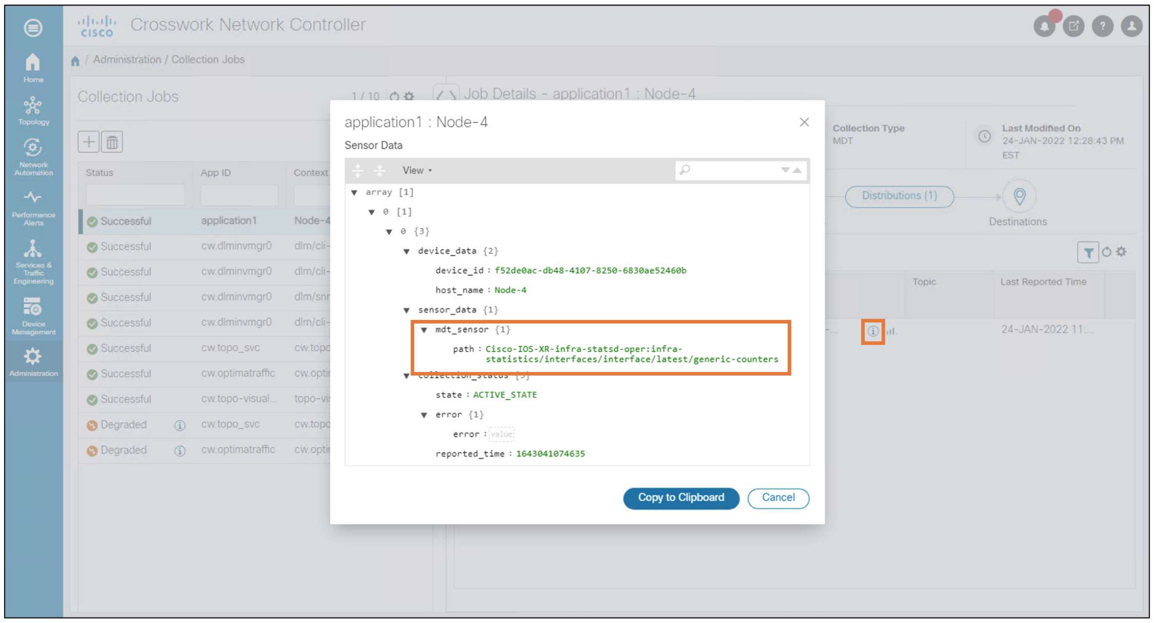
```
 -BE AWESOME!-

 End of banner message from server
 Keyboard-interactive authentication prompts from server:
| Password:
 End of keyboard-interactive prompts from server

RP/0/RP0/CPU0:Node-4#show conf commit change last 1
```

The command output should show the MDT sensor path, subscription and destination group, as shown below. This MDT configuration was automatically pushed down by Cisco NSO triggered by the API call.

```
telemetry model-driven
 destination-group CW_1b4ac245d863cf3e787d42bae97f1d18dd300d5e
  address-family ipv4 198.18.1.220 port 9010
   encoding self-describing-gpb
   protocol tcp
  !
 !
 sensor-group CW_1b4ac245d863cf3e787d42bae97f1d18dd300d5e
  sensor-path Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/latest/generic-counters
 !
 subscription CW_1b4ac245d863cf3e787d42bae97f1d18dd300d5e
  sensor-group-id CW_1b4ac245d863cf3e787d42bae97f1d18dd300d5e sample-interval 60000
  destination-id CW_1b4ac245d863cf3e787d42bae97f1d18dd300d5e
 !
!
end
```

## Step 7

Go back to Crosswork UI and click on the “**i**” icon close to the **Sensor Data** column entry. The Sensor data details will show the same MDT Sensor Path shown in the above CLI output. Click **Cancel** to exist the window.

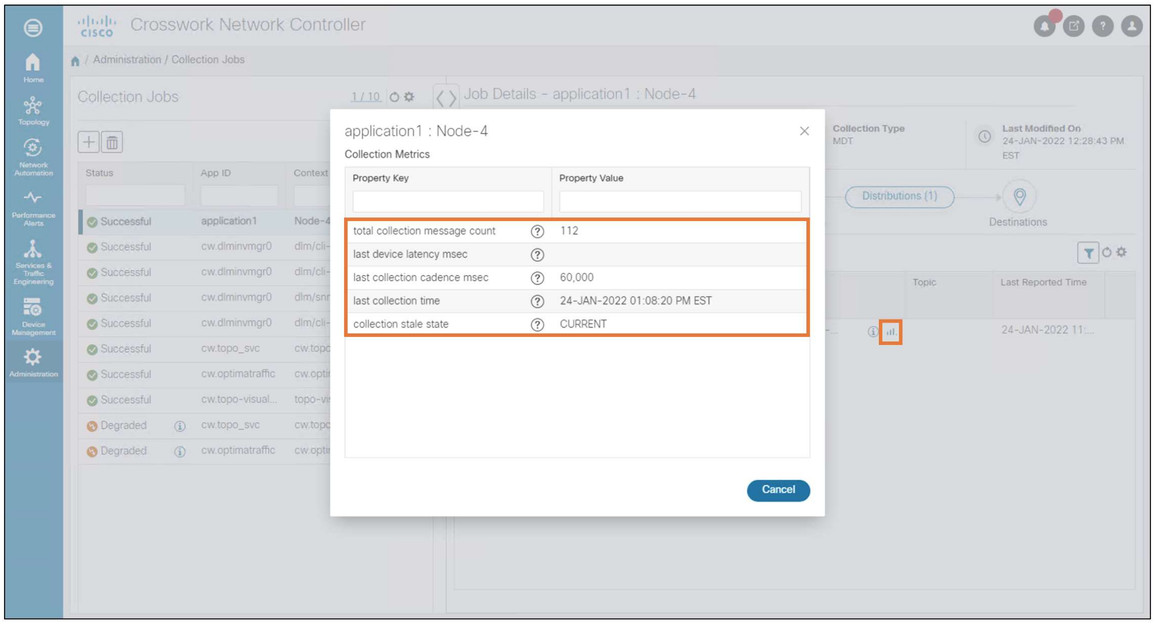


Crosswork Network Controller
Administration / Collection Jobs
Collection Jobs
Job Details - application1 : Node-4
application1 : Node-4
Sensor Data
View
Copy to Clipboard
Cancel

```
▼ array [1]
  ▼ 0 [1]
    ▼ 0 {3}
      ▼ device_data {2}
          device_id : f52de0ac-db48-4107-8250-6830ae52460b
          host_name : Node-4
      ▼ sensor_data {1}
        ▼ mdt_sensor {1}
            path : Cisco-IOS-XR-infra-statsd-oper:infra-
                   statistics/interfaces/interface/latest/generic-counters
      ▼ collection_status {3}
          state : ACTIVE_STATE
        ▼ error {1}
            error : value
          reported_time : 1643041074635
```

## Step 8

Now click on the signal bars icon close to the **Sensor Data** column entry. Note that it can take up 15 – 20 seconds for the collection job metrics window to appear. This window provides information related to the total number of received messages, latency, and collection time. Click **Cancel** to exit the collection job metrics window.



| Status | App ID | Context |
|---|---|---|
| Successful | application1 | Node-4 |
| Successful | cw.dlminvmgr0 | dlm/cli- |
| Successful | cw.dlminvmgr0 | dlm/cli- |
| Successful | cw.dlminvmgr0 | dlm/snr |
| Successful | cw.dlminvmgr0 | dlm/cli- |
| Successful | cw.topo_svc | cw.topo |
| Successful | cw.optimatraffic | cw.opti |
| Successful | cw.topo-visual... | topo-vi |
| Degraded | cw.topo_svc | cw.topo |
| Degraded | cw.optimatraffic | cw.opti |

Collection Type
MDT

Last Modified On
24-JAN-2022 12:28:43 PM EST

| Topic | Last Reported Time |
|---|---|
|  | 24-JAN-2022 11:... |

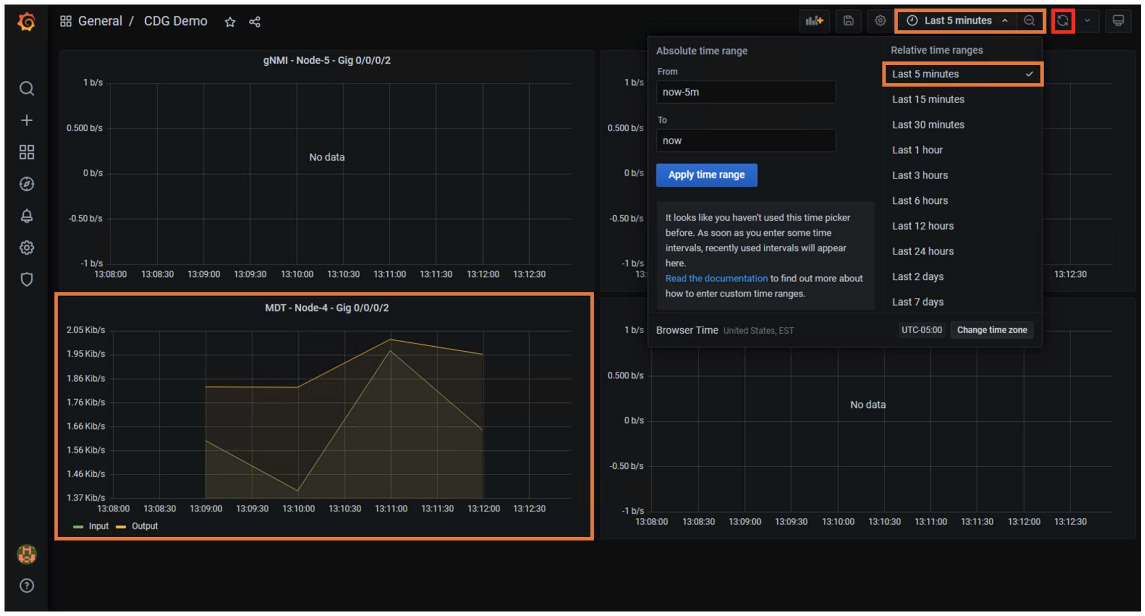
application1 : Node-4

Collection Metrics

| Property Key | Property Value |
|---|---|
| total collection message count | 112 |
| last device latency msec |  |
| last collection cadence msec | 60,000 |
| last collection time | 24-JAN-2022 01:08:20 PM EST |
| collection stale state | CURRENT |

## Step 9

Navigate back to the **Grafana** UI. Default time filter is set to **Last 6 hours** so your graph might appear compressed. On the right side click the time filter picker and select **Last 5 minutes**. Wait a few minutes and click on the refresh-button highlighted in red below and verify that your **MDT** graph is being populated. You may need to click on refresh button a few times to see the change.

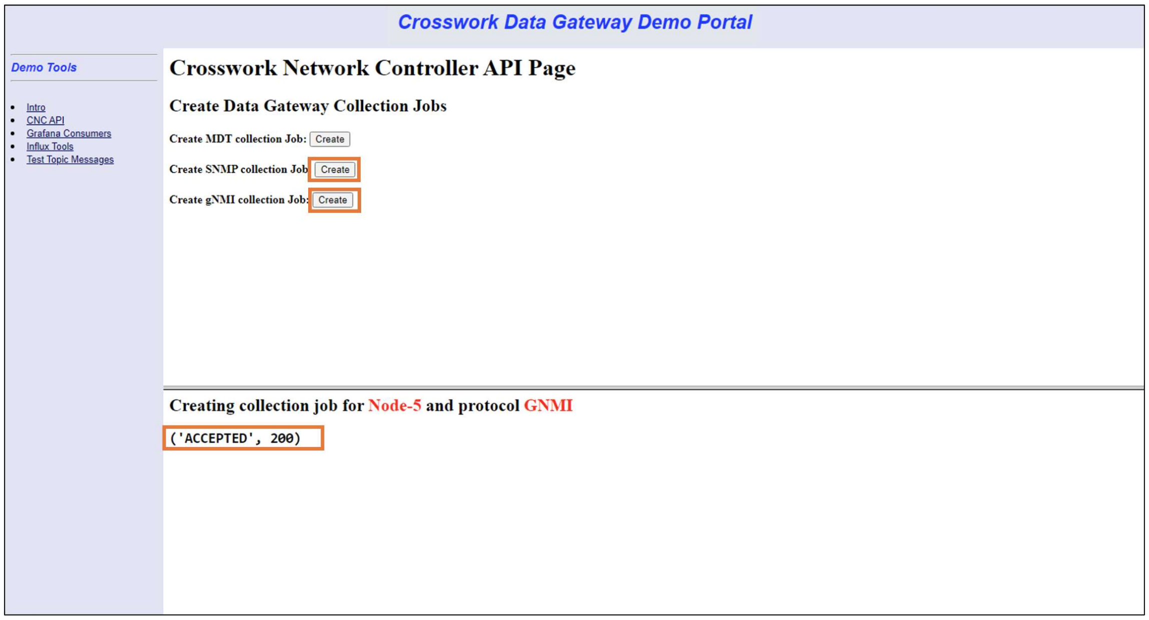


It looks like you haven't used this time picker before. As soon as you enter some time intervals, recently used intervals will appear here. Read the documentation to find out more about how to enter custom time ranges.

## Step 10

Navigate back to the CDG Demo Portal then click **Create** for **Create SNMP Collection Job** and **Create gNMI Collection Job**. Verify that you see a status message of (**ACCEPTED, 200**) for both operations in the bottom pane.

**NOTE**: Wait for the first Collection Job to prints the expected output on the on the bottom pane, before proceeding with the second one



Crosswork Network Controller API Page

Create Data Gateway Collection Jobs

Create MDT collection Job: Create

Create SNMP collection Job: Create

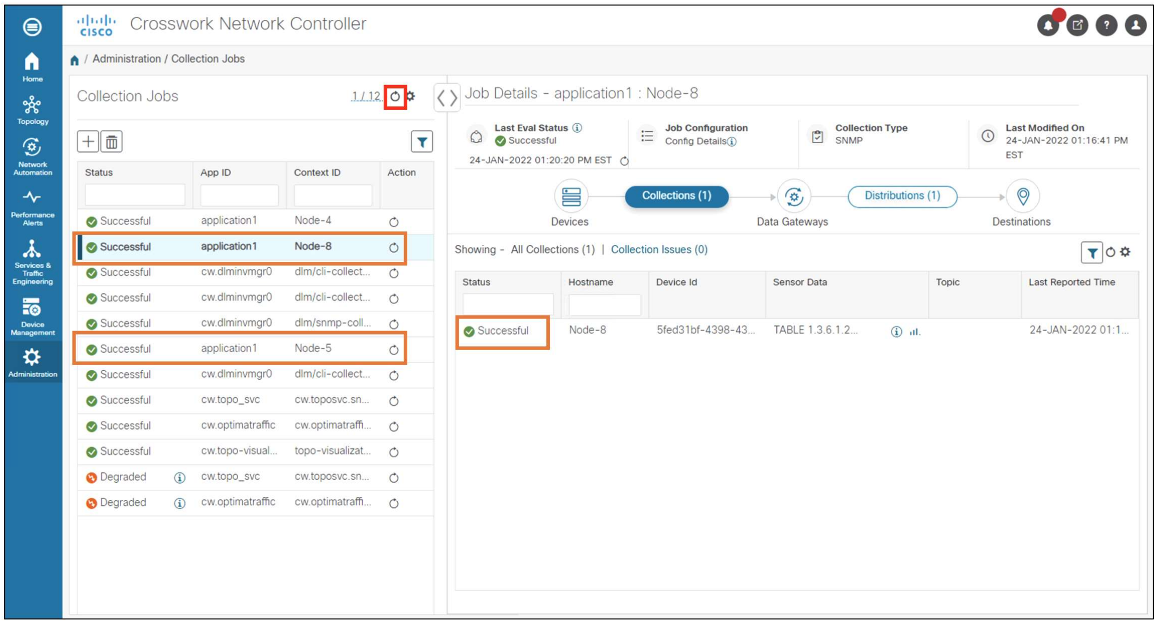
Create gNMI collection Job: Create

Creating collection job for Node-5 and protocol GNMI

```
('ACCEPTED', 200)
```

## Step 11

Navigate back to the Crosswork and check collection job status. You may have to click the Refresh icon to see the new jobs. They will both have **application1** in the **App ID** column and **Node-**5 and **Node-8** in the **Context ID** column. Verify both are showing green on the left pane.



Administration / Collection Jobs

Collection Jobs 1 / 12

| Status | App ID | Context ID | Action |
|---|---|---|---|
| Successful | application1 | Node-4 | |
| Successful | application1 | Node-8 | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.dlminvmgr0 | dlm/snmp-coll... | |
| Successful | application1 | Node-5 | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.topo_svc | cw.toposvc.sn... | |
| Successful | cw.optimatraffic | cw.optimatraffi... | |
| Successful | cw.topo-visual... | topo-visualizat... | |
| Degraded | cw.topo_svc | cw.toposvc.sn... | |
| Degraded | cw.optimatraffic | cw.optimatraffi... | |

Job Details - application1 : Node-8

Last Eval Status: Successful, 24-JAN-2022 01:20:20 PM EST
Job Configuration: Config Details
Collection Type: SNMP
Last Modified On: 24-JAN-2022 01:16:41 PM EST

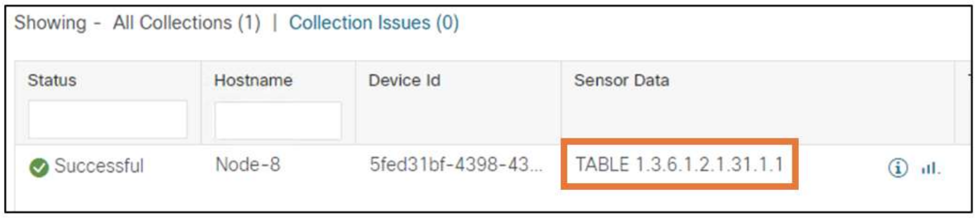
Devices — Collections (1) — Data Gateways — Distributions (1) — Destinations

Showing - All Collections (1) | Collection Issues (0)

| Status | Hostname | Device Id | Sensor Data | Topic | Last Reported Time |
|---|---|---|---|---|---|
| Successful | Node-8 | 5fed31bf-4398-43... | TABLE 1.3.6.1.2... | | 24-JAN-2022 01:1... |

## Step 12

In the left-hand pane, select the **Node-8** collection job and the **Sensor Data** column contains an SNMP OID.

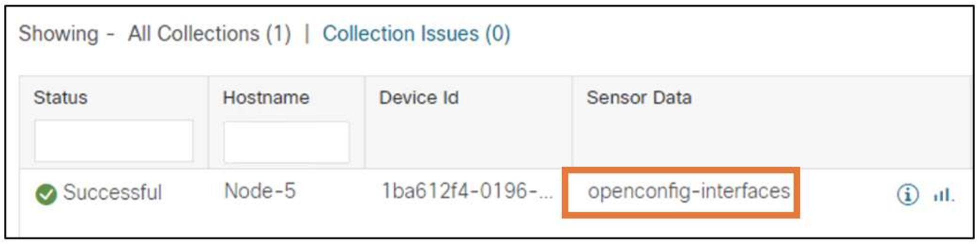


Showing - All Collections (1) | Collection Issues (0)

| Status | Hostname | Device Id | Sensor Data |
|---|---|---|---|
| Successful | Node-8 | 5fed31bf-4398-43... | TABLE 1.3.6.1.2.1.31.1.1 |

## Step 13

In the left-pane, again select the **Node-5** job. Check that the **Sensor Data** column contains **openconfig-interefaces**. This indicates a YANG data model

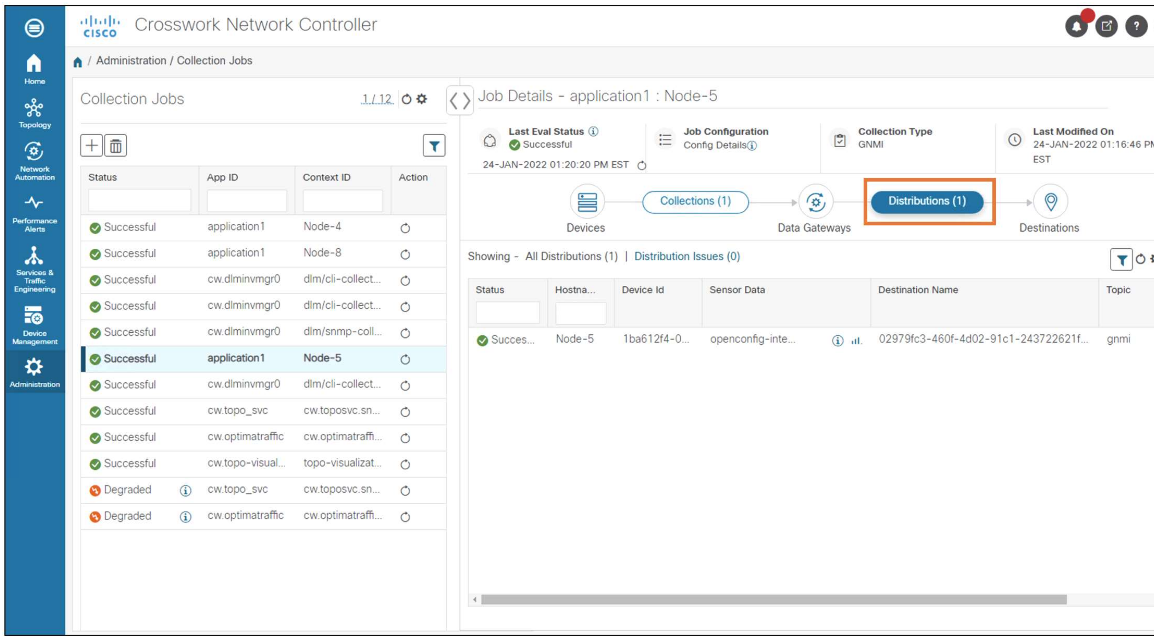


Showing - All Collections (1) | Collection Issues (0)

| Status | Hostname | Device Id | Sensor Data |
|---|---|---|---|
| ✓ Successful | Node-5 | 1ba612f4-0196-... | openconfig-interfaces |

## Step 14

Click on **Distribution**



/ Administration / Collection Jobs

Collection Jobs 1 / 12

| Status | App ID | Context ID | Action |
|---|---|---|---|
| Successful | application1 | Node-4 | ○ |
| Successful | application1 | Node-8 | ○ |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | ○ |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | ○ |
| Successful | cw.dlminvmgr0 | dlm/snmp-coll... | ○ |
| Successful | application1 | Node-5 | ○ |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | ○ |
| Successful | cw.topo_svc | cw.toposvc.sn... | ○ |
| Successful | cw.optimatraffic | cw.optimatraffi... | ○ |
| Successful | cw.topo-visual... | topo-visualizat... | ○ |
| Degraded ⓘ | cw.topo_svc | cw.toposvc.sn... | ○ |
| Degraded ⓘ | cw.optimatraffic | cw.optimatraffi... | ○ |

Job Details - application1 : Node-5

Last Eval Status ⓘ Successful
24-JAN-2022 01:20:20 PM EST
Job Configuration Config Details ⓘ
Collection Type GNMI
Last Modified On 24-JAN-2022 01:16:46 PM EST

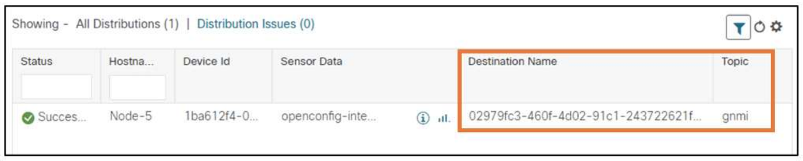
Devices → Collections (1) → Data Gateways → Distributions (1) → Destinations

Showing - All Distributions (1) | Distribution Issues (0)

| Status | Hostna... | Device Id | Sensor Data | | Destination Name | Topic |
|---|---|---|---|---|---|---|
| Succes... | Node-5 | 1ba612f4-0... | openconfig-inte... | ⓘ | 02979fc3-460f-4d02-91c1-243722621f... | gnmi |

## Step 15

In the left pane, select the **Node-5** job. In our lab we just have one external destination defined. One of the Data Gateway use cases is to act as single collection point for multiple applications. The Data Gateway can send the same collected data to multiple external destinations at the same time and those would be reported here. Note that your **Destination Name** may vary from what is shown below. The Kafka **topic** will also be shown.

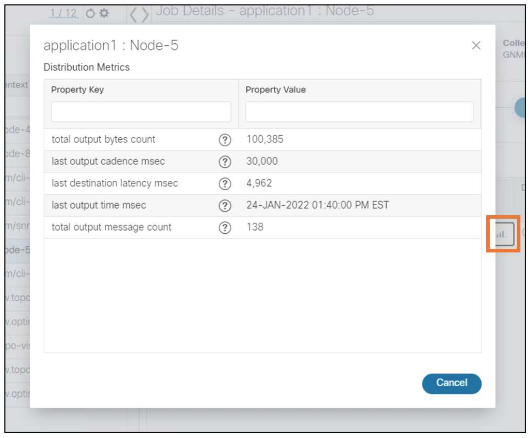


Showing - All Distributions (1) | Distribution Issues (0)

| Status | Hostna... | Device Id | Sensor Data | | Destination Name | Topic |
|---|---|---|---|---|---|---|
| Succes... | Node-5 | 1ba612f4-0... | openconfig-inte... | ⓘ | 02979fc3-460f-4d02-91c1-243722621f... | gnmi |

## Step 16

(Optional Step). Click the signal bars icon in the **Sensor Data** column (you might need to enlarge the column to see the icon). Note that it will take about 15 – 20 seconds for the collection metrics dialog to appear. This will show collection metrics on the distribution side as opposed to those seen before on the collection side. Click **Cancel** to exit.



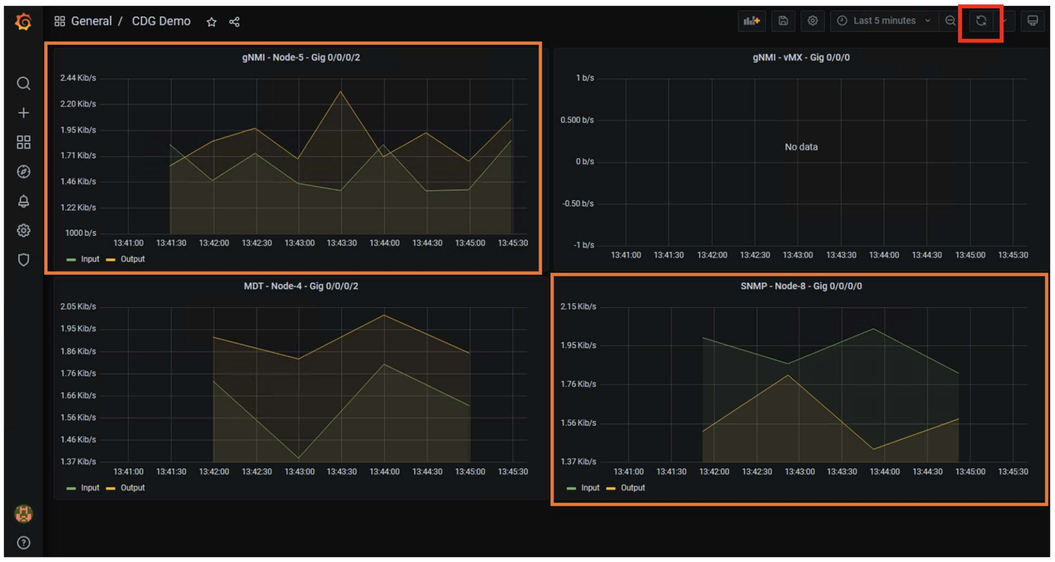
application1 : Node-5

Distribution Metrics

| Property Key | Property Value |
|---|---|
| total output bytes count | 100,385 |
| last output cadence msec | 30,000 |
| last destination latency msec | 4,962 |
| last output time msec | 24-JAN-2022 01:40:00 PM EST |
| total output message count | 138 |

## Step 17

Navigate back to the **Grafana**. Wait a few minutes and click on the refresh button on the upper right-hand side. Verify that the **gNMI** graph for **Node-5** and the SNMP graph for **Node-8** are now being populated. You may need to click the **Refresh-dashboard** icon in the upper right-hand corner a few times before you see the change.


General / CDG Demo
Last 5 minutes
gNMI - Node-5 - Gig 0/0/0/2
gNMI - vMX - Gig 0/0/0
No data
MDT - Node-4 - Gig 0/0/0/2
SNMP - Node-8 - Gig 0/0/0/0
Input
Output

**This concludes task 2.**

## Task 3: Create Collection Jobs using Postman

In this task, user will interact directly with Crosswork API. Target will same as previous task: collect interface traffic counters and graphically present them on Grafana. This time user will be using Postman rather that the CDG Portal as API client. Objective of this task is to practice with Crosswork API and their payload. User will review the required payload in details.

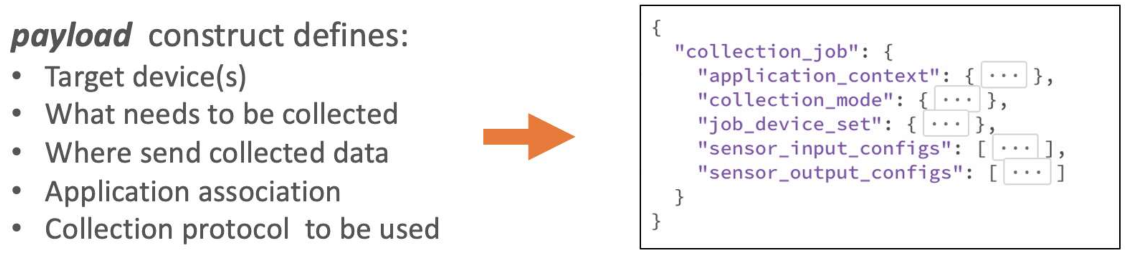
During this task, we will be using 3 API endpoints:

- /v1/nodes/query (GET method)
- /v1/destinations/query (GET method)
- /v1/collectionjob (PUT method)

First API returns all device details. One of those is the **device uuid** (unique internal identifier), required parameter for the create collection job API call

Second API returns all Crosswork destinations (either internal or external Kafka or gRPC servers) details. One of those is the **destination uuid**, also required for the create collection job API call

Third API is the one creating collection jobs. This API requires an articulated payload be used. This payload has multiple sections listed below:



***payload*** construct defines:

- Target device(s)
- What needs to be collected
- Where send collected data
- Application association
- Collection protocol to be used

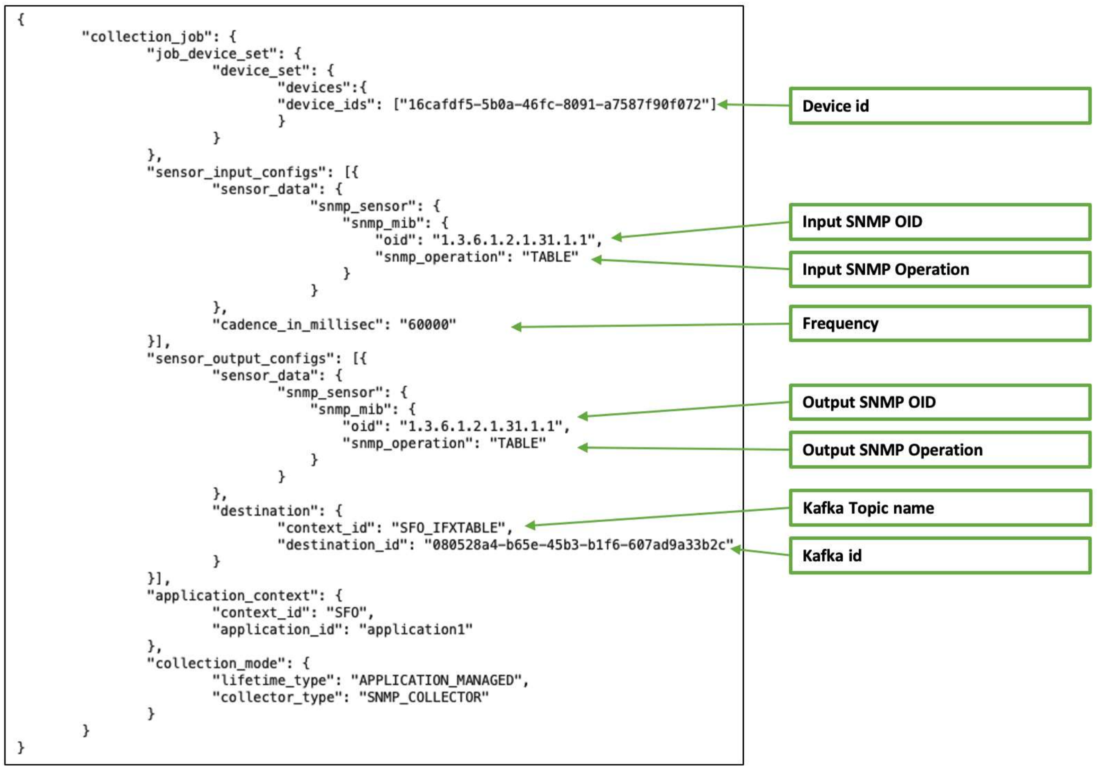
```
{
  "collection_job": {
    "application_context": {…},
    "collection_mode": {…},
    "job_device_set": {…},
    "sensor_input_configs": […],
    "sensor_output_configs": […]
  }
}
```

We will expand them in the coming example.

**NOTE**: dCloud labs come with all API payload already populated. Some payload parameters are static, other are dynamically generated by the system and are different for each session. As a result, we must update their values. To that end, first two API calls (GET nodes and GET destinations) will automatically set those parameters required on the third call (using Postman tests and variables).

## Payload Example:

The following graphic illustrates the **collection job** payload and highlights the most important parameters.



```
{
        "collection_job": {
                "job_device_set": {
                        "device_set": {
                                "devices":{
                                "device_ids": ["16cafdf5-5b0a-46fc-8091-a7587f90f072"]
                                }
                        }
                },
                "sensor_input_configs": [{
                        "sensor_data": {
                              "snmp_sensor": {
                                  "snmp_mib": {
                                      "oid": "1.3.6.1.2.1.31.1.1",
                                      "snmp_operation": "TABLE"
                                  }
                              }
                        },
                        "cadence_in_millisec": "60000"
                }],
                "sensor_output_configs": [{
                        "sensor_data": {
                                "snmp_sensor": {
                                    "snmp_mib": {
                                        "oid": "1.3.6.1.2.1.31.1.1",
                                        "snmp_operation": "TABLE"
                                    }
                                }
                        },
                        "destination": {
                                "context_id": "SFO_IFXTABLE",
                                "destination_id": "080528a4-b65e-45b3-b1f6-607ad9a33b2c"
                        }
                }],
                "application_context": {
                        "context_id": "SFO",
                        "application_id": "application1"
                },
                "collection_mode": {
                        "lifetime_type": "APPLICATION_MANAGED",
                        "collector_type": "SNMP_COLLECTOR"
                }
        }
}
```

- The **device_ids** and **destination_id** (made of long HEX value strings), uniquely identify nodes and external destinations (Kafka or gRPC servers) within the system. **device_ids** can also be a tag to have device scope changing dynamically rather be statically set.

- The **sensor_data** value for both **sensor_input_configs** and **sensor_output_configs** sections MUST match.

- The **destination** section is identified by the Kafka or gRPC server **destination_id** and a **context_id**.

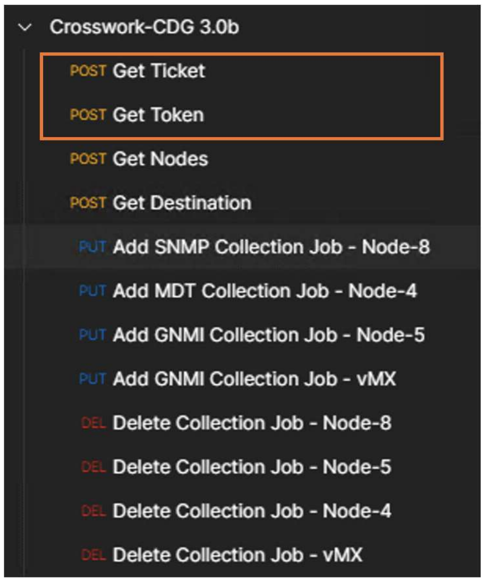
- The **application_context** section (composed of **application_id** and a **context_id**) uniquely identifies every single collection job within the system.

## Postman and provided CDG library

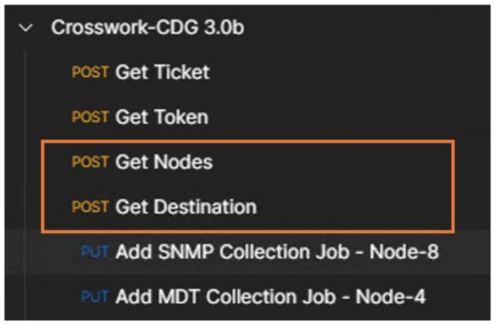
In this task, students will use Postman. Postman is a useful client for testing web services and makes it easy to test, develop, and automate API execution. Provided Postman library has all API commands to execute all lab tasks.

**Authentication**: Crosswork uses a two-step authentication process. In the first authentication step, the client connects to Crosswork with a username and password and Crosswork returns a ticket which, in turn, will be used by the client to connect back to Crosswork to retrieve a JWT (Java Web Token) time-limited token. This will be then used by all subsequent calls as part of the call headers.

Following illustrates those related API commands in the provided library


Crosswork-CDG 3.0b
POST Get Ticket
POST Get Token
POST Get Nodes
POST Get Destination
PUT Add SNMP Collection Job - Node-8
PUT Add MDT Collection Job - Node-4
PUT Add GNMI Collection Job - Node-5
PUT Add GNMI Collection Job - vMX
DEL Delete Collection Job - Node-8
DEL Delete Collection Job - Node-5
DEL Delete Collection Job - Node-4
DEL Delete Collection Job - vMX

Then we have two commands to execute the first two API described at the beginning of this task, to retrieve nodes and destination details.

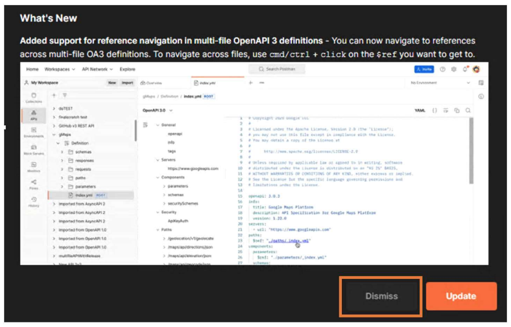


Finally, Postman library has four commands available to create collection jobs on different nodes using SNMP, MDT and gNMI as collection protocols.

## Step 1

On the remote workstation, launch **Postman**.

Clink on Dismiss if asked to upgrade

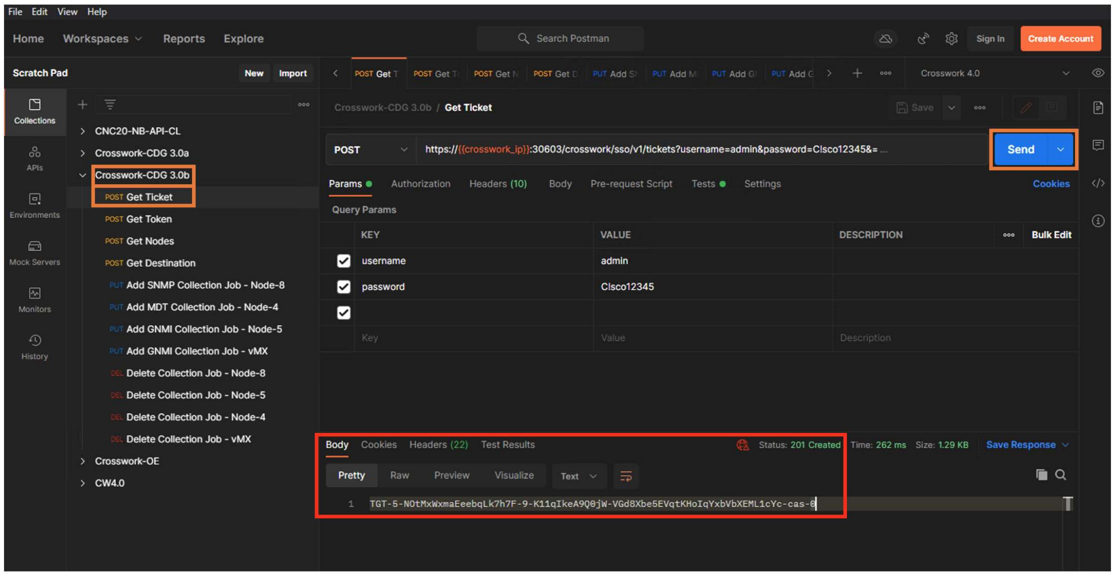


What's New

**Added support for reference navigation in multi-file OpenAPI 3 definitions** - You can now navigate to references across multi-file OA3 definitions. To navigate across files, use `cmd/ctrl + click` on the `$ref` you want to get to.

## Step 2

On the left side of the Postman UI, there's a list of API collections. Expand the one named **Crosswork-CDG3.0b** and click **Get Ticket**. Then click **Send** on the right side of the window. Make sure a ticket is generated and visible in the result section highlighted below.



Crosswork-CDG 3.0b / Get Ticket

```
POST https://{{crosswork_ip}}:30603/crosswork/sso/v1/tickets?username=admin&password=Clsco12345&= ...
```

Query Params

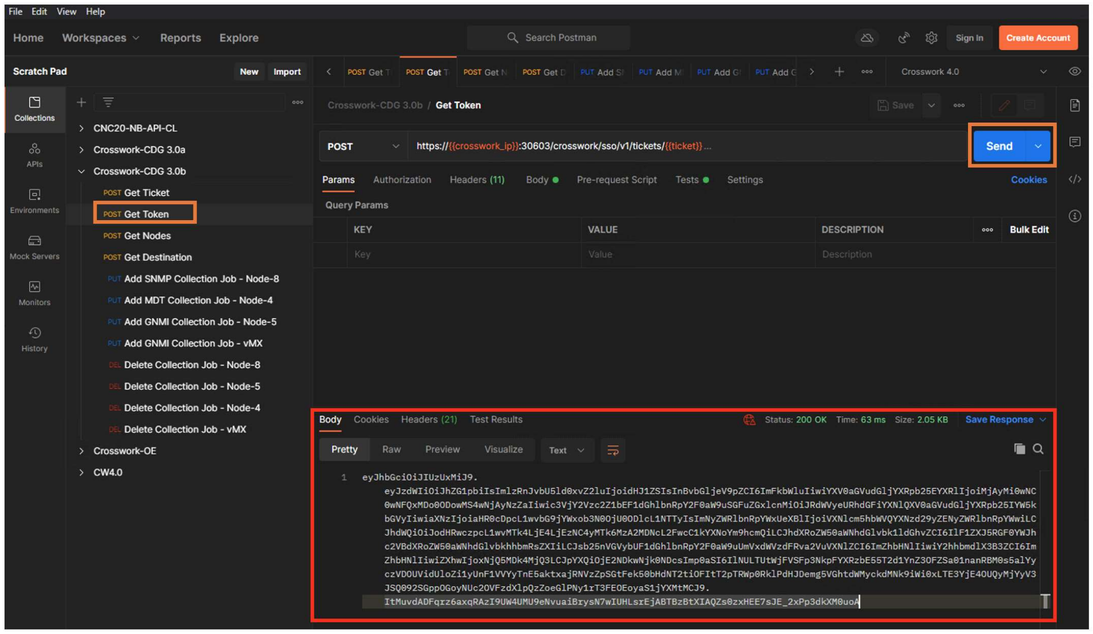
| KEY | VALUE | DESCRIPTION |
|---|---|---|
| username | admin | |
| password | Clsco12345 | |

Status: 201 Created Time: 262 ms Size: 1.29 KB

```
TGT-5-NOtMxWxmaEeebqLk7h7F-9-K11qIkeA9Q0jW-VGd8Xbe5EVqtKHoIqYxbVbXEML1cYc-cas-0
```

## Step 3

Click on the **Get Toke**n command. Then Click on **Send** on the right side of the window. Make sure a token is generated and visible in the result section here highlighted in red.


File Edit View Help
Home Workspaces Reports Explore
Search Postman
Sign In Create Account
Scratch Pad New Import
Collections
APIs
Environments
Mock Servers
Monitors
History
CNC20-NB-API-CL
Crosswork-CDG 3.0a
Crosswork-CDG 3.0b
POST Get Ticket
POST Get Token
POST Get Nodes
POST Get Destination
PUT Add SNMP Collection Job - Node-8
PUT Add MDT Collection Job - Node-4
PUT Add GNMI Collection Job - Node-5
PUT Add GNMI Collection Job - vMX
DEL Delete Collection Job - Node-8
DEL Delete Collection Job - Node-5
DEL Delete Collection Job - Node-4
DEL Delete Collection Job - vMX
Crosswork-OE
CW4.0
Crosswork 4.0
Crosswork-CDG 3.0b / Get Token
Save
POST
Send
Params Authorization Headers (11) Body Pre-request Script Tests Settings
Cookies
Query Params
KEY VALUE DESCRIPTION Bulk Edit
Key Value Description
Body Cookies Headers (21) Test Results
Save Response
Pretty Raw Preview Visualize Text

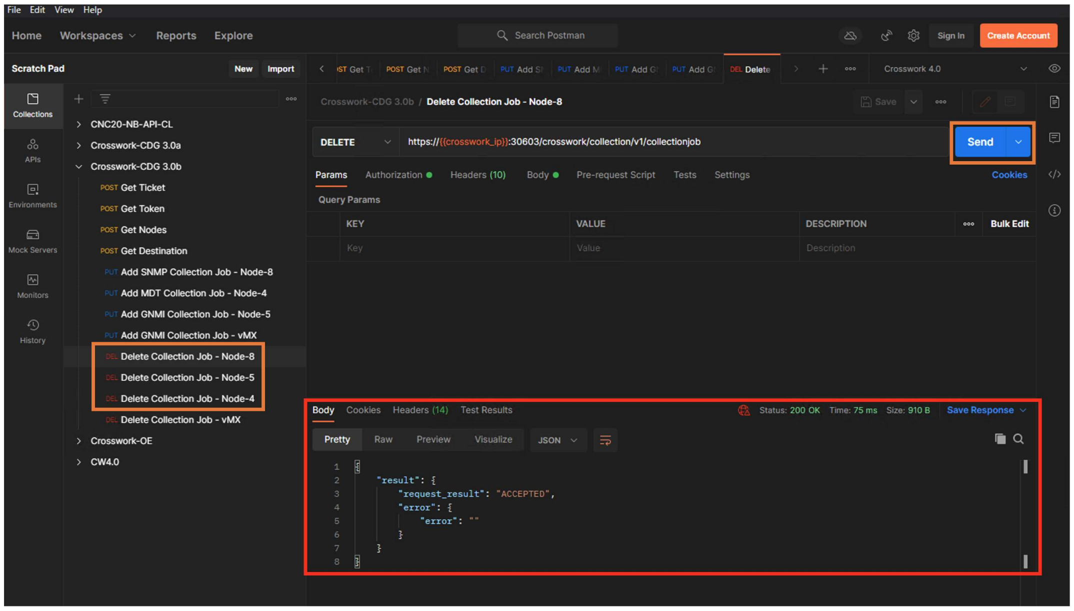
POST https://{{crosswork_ip}}:30603/crosswork/sso/v1/tickets/{{ticket}}...

Status: 200 OK Time: 63 ms Size: 2.05 KB

```
eyJhbGciOiJIUzUxMiJ9.
    eyJzdWIiOiJhZG1pbiIsImlzRnJvbU5ld0xvZ2luIjoidHJ1ZSIsInBvbGljeV9pZCI6ImFkbWluIiwiYXV0aGVudGljYXRpb25EYXRlIjoiMjAyMi0wNC
    0wNFQxMDo0ODowMS4wNjAyNzZaIiwic3VjY2Vzc2Z1bEF1dGhlbnRpY2F0aW9uSGFuZGxlcnMiOiJRdWVyeURhdGFiYXNlQXV0aGVudGljYXRpb25IYW5k
    bGVyIiwiaXNzIjoiaHR0cDpcL1wvbG9jYWxob3N0OjU0ODlcL1NTTyIsImNyZWRlbnRpYWxUeXBlIjoiVXNlcm5hbWVQYXNzd29yZENyZWRlbnRpYWwiLC
    JhdWQiOiJodHRwczpcL1wvMTk4LjE4LjEzNC4yMTk6MzA2MDNcL2FwcC1kYXNoYm9hcmQiLCJhdXRoZW50aWNhdGlvbk1ldGhvZCI6IlF1ZXJ5RGF0YWJh
    c2VBdXRoZW50aWNhdGlvbkhhbmRsZXIiLCJsb25nVGVybUF1dGhlbnRpY2F0aW9uUmVxdWVzdFRva2VuVXNlZCI6ImZhbHNlIiwiY2hhbmdlX3B3ZCI6Im
    ZhbHNlIiwiZXhwIjoxNjQ5MDk4MjQ3LCJpYXQiOjE2NDkwNjk0NDcsImp0aSI6IlNULTUtWjFVSFp3NkpFYXRzbE55T2d1YnZ3OFZSa01nanRBM0s5alYy
    czVDOUVidUloZi1yUnF1VVYyTnE5aktxajRNVzZpSGtFek50bHdNT2tiOFItT2pTRWp0RklPdHJDemg5VGhtdWMyckdMNk9iWi0xLTE3YjE4OUQyMjYyV3
    JSQ092SGppOGoyNUc2OVFzdXlpQzZoeGlPNy1rT3FEOEoyaS1jYXMtMCJ9.
    ItMuvdADFqrz6axqRAzI9UW4UMU9eNvuaiBrysN7wIUHLsrEjABTBzBtXIAQZs0zxHEE7sJE_2xPp3dkXM0uoA
```

## Step 4

If you completed task 2, you should have three user defined collections jobs running. Let's use Postman to delete them all. One by one, click the **Delete Collection Job - Node-8**, **Delete Collection Job - Node-5**, and **Delete Collection Job - Node-4** and click **Send** on all commands. Verify that “**requested_result**”: “**ACCEPTED**” is returned in the result section highlighted in red below.

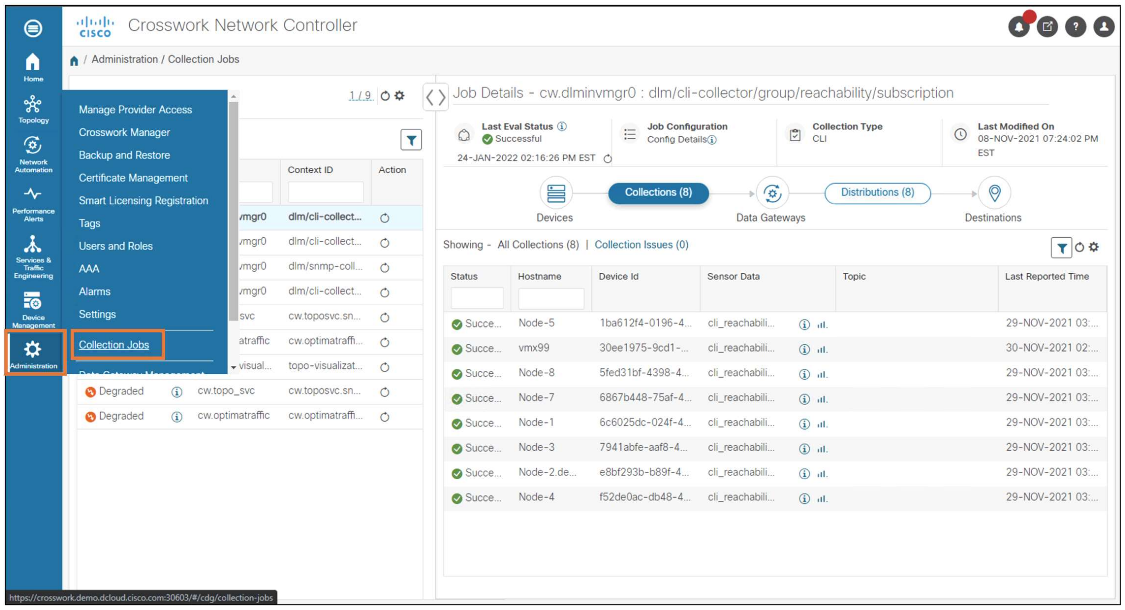

File Edit View Help
Home Workspaces Reports Explore
Search Postman
Sign In
Create Account
Scratch Pad
New
Import
Collections
APIs
Environments
Mock Servers
Monitors
History
CNC20-NB-API-CL
Crosswork-CDG 3.0a
Crosswork-CDG 3.0b
POST Get Ticket
POST Get Token
POST Get Nodes
POST Get Destination
PUT Add SNMP Collection Job - Node-8
PUT Add MDT Collection Job - Node-4
PUT Add GNMI Collection Job - Node-5
PUT Add GNMI Collection Job - vMX
DEL Delete Collection Job - Node-8
DEL Delete Collection Job - Node-5
DEL Delete Collection Job - Node-4
DEL Delete Collection Job - vMX
Crosswork-OE
CW4.0
Crosswork 4.0
Crosswork-CDG 3.0b / Delete Collection Job - Node-8
Save
Send
Params Authorization Headers (10) Body Pre-request Script Tests Settings
Cookies
Query Params
KEY VALUE DESCRIPTION
Bulk Edit
Key Value Description
Body Cookies Headers (14) Test Results
Status: 200 OK Time: 75 ms Size: 910 B
Save Response
Pretty Raw Preview Visualize JSON

```
DELETE https://{{crosswork_ip}}:30603/crosswork/collection/v1/collectionjob
```

```
{
    "result": {
        "request_result": "ACCEPTED",
        "error": {
            "error": ""
        }
    }
}
```

## Step 5

In Crosswork, navigate to **Administration -> Collection Jobs**.



| | | Context ID | Action |
|---|---|---|---|
| | vmgr0 | dlm/cli-collect... | |
| | vmgr0 | dlm/cli-collect... | |
| | vmgr0 | dlm/snmp-coll... | |
| | vmgr0 | dlm/cli-collect... | |
| | svc | cw.toposvc.sn... | |
| | atraffic | cw.optimatraffi... | |
| | visual... | topo-visualizat... | |
| Degraded | cw.topo_svc | cw.toposvc.sn... | |
| Degraded | cw.optimatraffic | cw.optimatraffi... | |

Job Details - cw.dlminvmgr0 : dlm/cli-collector/group/reachability/subscription

Last Eval Status: Successful
24-JAN-2022 02:16:26 PM EST

Job Configuration: Config Details

Collection Type: CLI

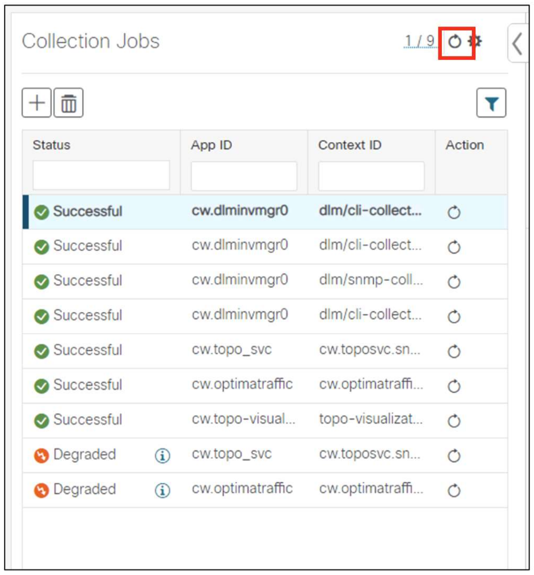
Last Modified On: 08-NOV-2021 07:24:02 PM EST

Showing - All Collections (8) | Collection Issues (0)

| Status | Hostname | Device Id | Sensor Data | Topic | Last Reported Time |
|---|---|---|---|---|---|
| Succe... | Node-5 | 1ba612f4-0196-4... | cli_reachabili... | | 29-NOV-2021 03:... |
| Succe... | vmx99 | 30ee1975-9cd1-... | cli_reachabili... | | 30-NOV-2021 02:... |
| Succe... | Node-8 | 5fed31bf-4398-4... | cli_reachabili... | | 29-NOV-2021 03:... |
| Succe... | Node-7 | 6867b448-75af-4... | cli_reachabili... | | 29-NOV-2021 03:... |
| Succe... | Node-1 | 6c6025dc-024f-4... | cli_reachabili... | | 29-NOV-2021 03:... |
| Succe... | Node-3 | 7941abfe-aaf8-4... | cli_reachabili... | | 29-NOV-2021 03:... |
| Succe... | Node-2.de... | e8bf293b-b89f-4... | cli_reachabili... | | 29-NOV-2021 03:... |
| Succe... | Node-4 | f52de0ac-db48-4... | cli_reachabili... | | 29-NOV-2021 03:... |

## Step 6

Verify that previous collection jobs have been removed from the job list. If the page was already open (as in our example), content wouldn't be automatically refreshed. You may have to click refresh (in red below).



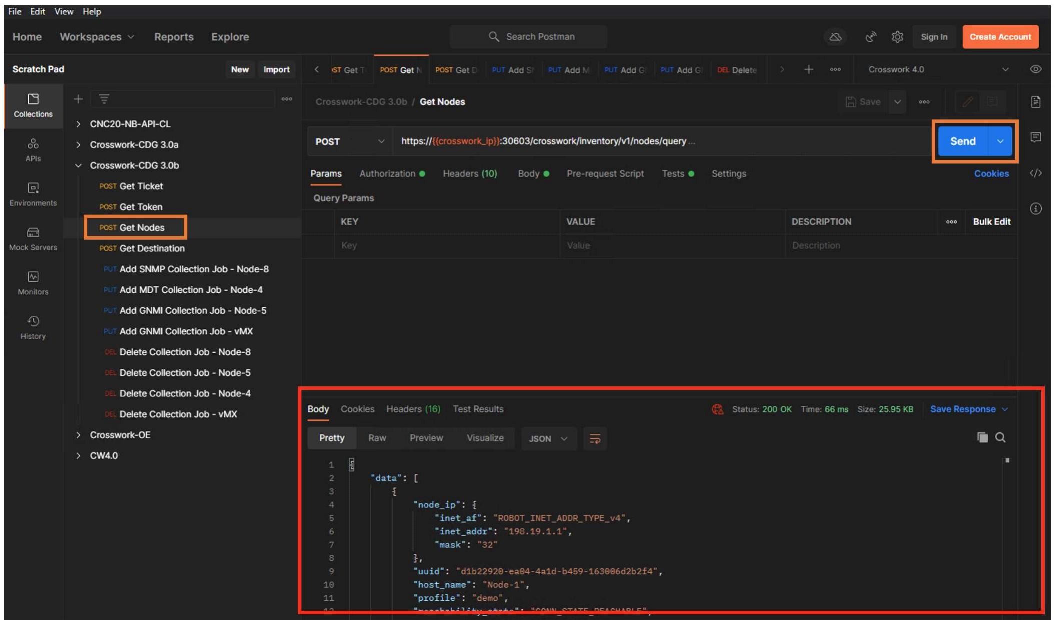
Collection Jobs

1 / 9

| Status | App ID | Context ID | Action |
|---|---|---|---|
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.dlminvmgr0 | dlm/snmp-coll... | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.topo_svc | cw.toposvc.sn... | |
| Successful | cw.optimatraffic | cw.optimatraffi... | |
| Successful | cw.topo-visual... | topo-visualizat... | |
| Degraded | cw.topo_svc | cw.toposvc.sn... | |
| Degraded | cw.optimatraffic | cw.optimatraffi... | |

## Step 7

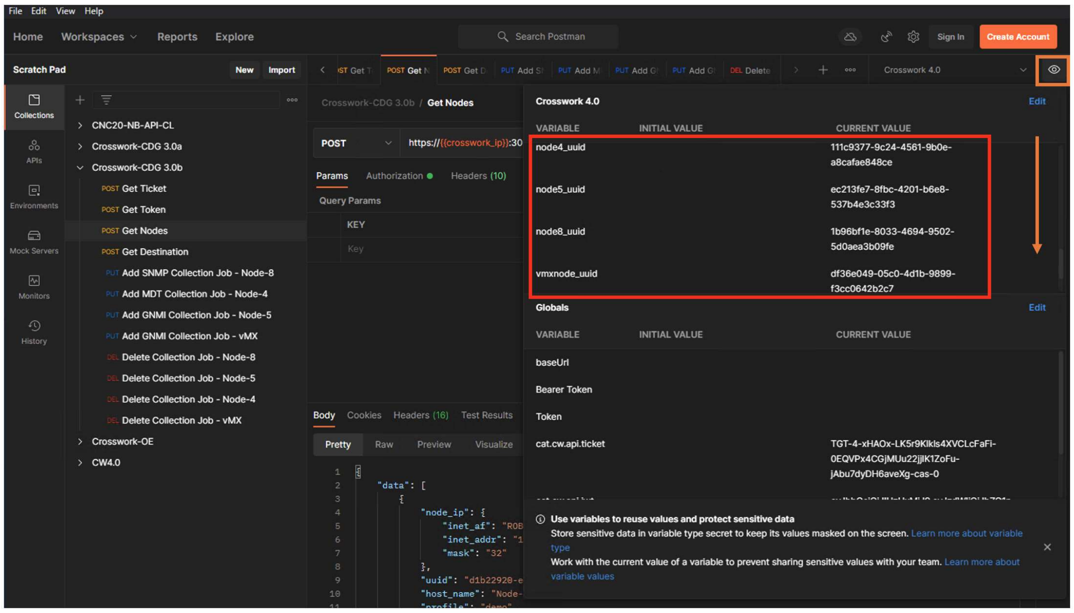
Go back to Postman UI. Click on the **Get Nodes** command. Then click on **Send** on the right side of the window. Verify output similar to the one below is generated, and no error is returned.


File Edit View Help
Home Workspaces Reports Explore
Search Postman
Sign In
Create Account
Scratch Pad
New Import
Crosswork 4.0
Collections
APIs
Environments
Mock Servers
Monitors
History
CNC20-NB-API-CL
Crosswork-CDG 3.0a
Crosswork-CDG 3.0b
POST Get Ticket
POST Get Token
POST Get Nodes
POST Get Destination
PUT Add SNMP Collection Job - Node-8
PUT Add MDT Collection Job - Node-4
PUT Add GNMI Collection Job - Node-5
PUT Add GNMI Collection Job - vMX
DEL Delete Collection Job - Node-8
DEL Delete Collection Job - Node-5
DEL Delete Collection Job - Node-4
DEL Delete Collection Job - vMX
Crosswork-OE
CW4.0
Crosswork-CDG 3.0b / Get Nodes
Save
POST
https://{{crosswork_ip}}:30603/crosswork/inventory/v1/nodes/query ...
Send
Params Authorization Headers (10) Body Pre-request Script Tests Settings
Cookies
Query Params
KEY VALUE DESCRIPTION
Bulk Edit
Key Value Description
Body Cookies Headers (16) Test Results
Status: 200 OK Time: 66 ms Size: 25.95 KB
Save Response
Pretty Raw Preview Visualize JSON

```
{
    "data": [
        {
            "node_ip": {
                "inet_af": "ROBOT_INET_ADDR_TYPE_v4",
                "inet_addr": "198.19.1.1",
                "mask": "32"
            },
            "uuid": "d1b22920-ea04-4a1d-b459-163006d2b2f4",
            "host_name": "Node-1",
            "profile": "demo",
```

## Step 8

Click on the "**eye**" icon on the top-right of the window, then scroll down the variable list till you see **node8_uuid** as sown in the picture. You should see current value being populated with device uuid for all nodes.



**Crosswork 4.0**

| VARIABLE | INITIAL VALUE | CURRENT VALUE |
|---|---|---|
| node4_uuid | | 111c9377-9c24-4561-9b0e-a8cafae848ce |
| node5_uuid | | ec213fe7-8fbc-4201-b6e8-537b4e3c33f3 |
| node8_uuid | | 1b96bf1e-8033-4694-9502-5d0aea3b09fe |
| vmxnode_uuid | | df36e049-05c0-4d1b-9899-f3cc0642b2c7 |

**Globals**

| VARIABLE | INITIAL VALUE | CURRENT VALUE |
|---|---|---|
| baseUrl | | |
| Bearer Token | | |
| Token | | |
| cat.cw.api.ticket | | TGT-4-xHAOx-LK5r9KIkls4XVCLcFaFi-0EQVPx4CGjMUu22jjIK1ZoFu-jAbu7dyDH6aveXg-cas-0 |

Use variables to reuse values and protect sensitive data
Store sensitive data in variable type secret to keep its values masked on the screen. Learn more about variable type
Work with the current value of a variable to prevent sharing sensitive values with your team. Learn more about variable values

Crosswork-CDG 3.0b / Get Nodes

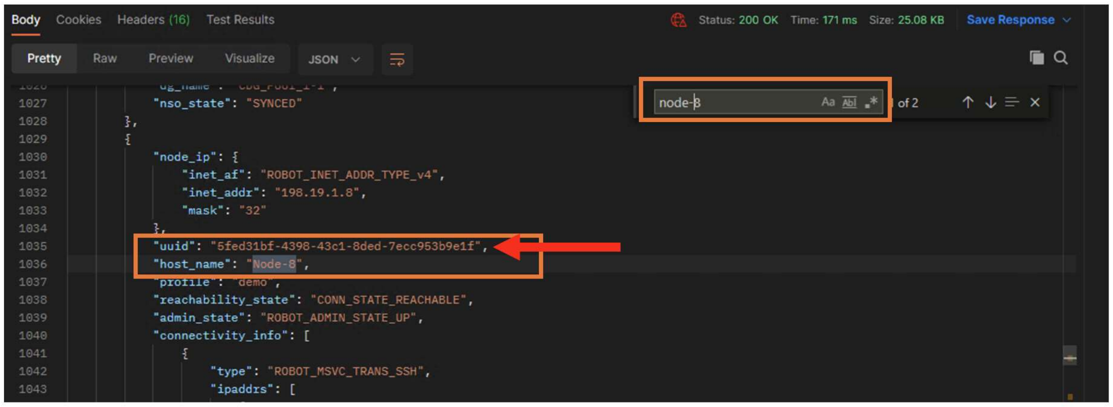
POST https://{{crosswork_ip}}:30

```
{
    "data": [
        {
            "node_ip": {
                "inet_af": "ROB
                "inet_addr": "1
                "mask": "32"
            },
            "uuid": "d1b22920-e
            "host_name": "Node-
```

## Step 9

Click anywhere in the result (bottom) frame and press **CTRL-F**. The search text-field appears. Type **Node-8** in the field. Postman will automatically go to the section in the result output where **Node-8** is returned. Look at the uuid shown there and compare its value with the one shown on the variable list seen before. They must match. No need to check all nodes, just Node-8 or any other node_uuid present in the variable list will be fine.

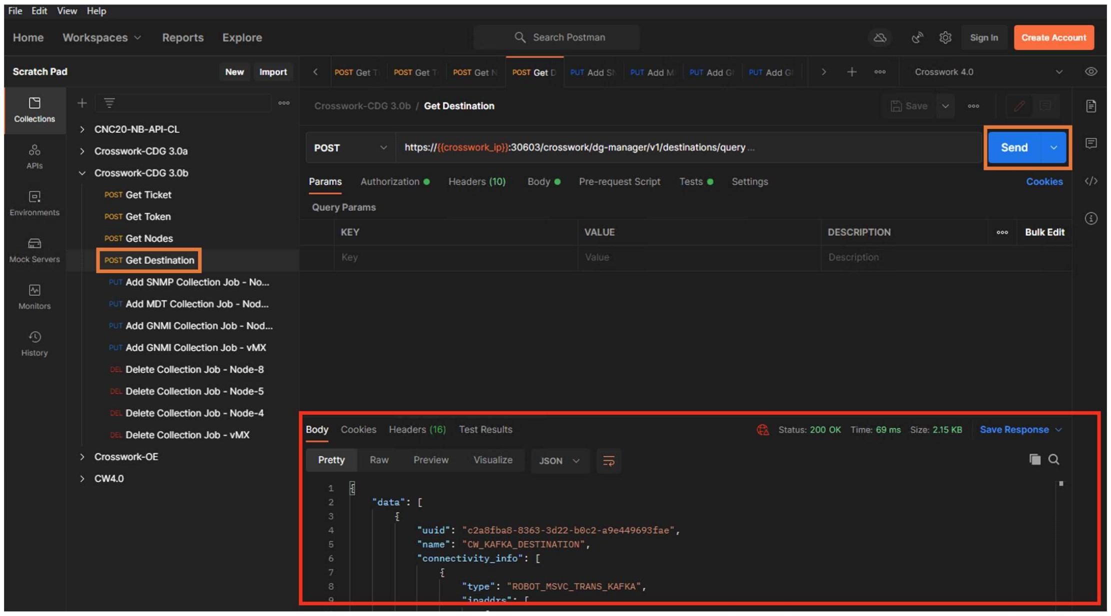
(Optional) User may want to review the whole command result



```
            "nso_state": "SYNCED"
        },
        {
            "node_ip": {
                "inet_af": "ROBOT_INET_ADDR_TYPE_v4",
                "inet_addr": "198.19.1.8",
                "mask": "32"
            },
            "uuid": "5fed31bf-4398-43c1-8ded-7ecc953b9e1f",
            "host_name": "Node-8",
            "profile": "demo",
            "reachability_state": "CONN_STATE_REACHABLE",
            "admin_state": "ROBOT_ADMIN_STATE_UP",
            "connectivity_info": [
                {
                    "type": "ROBOT_MSVC_TRANS_SSH",
                    "ipaddrs": [
```

## Step 10

Click **Get Destinations**. Then click **Send**. Verify that the output is similar to what is shown below, with no errors returned.



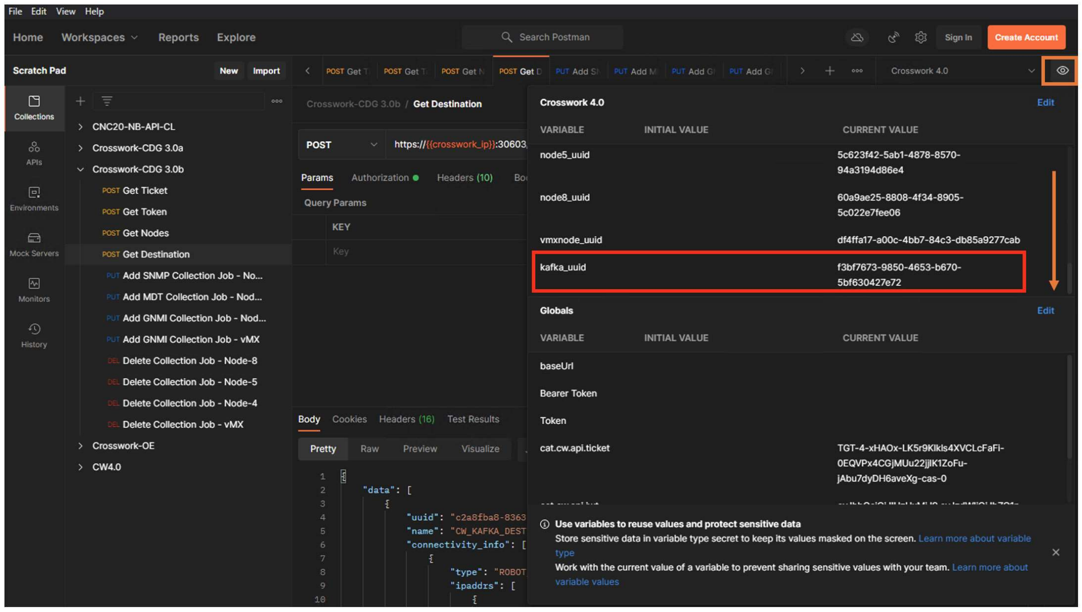
POST https://{{crosswork_ip}}:30603/crosswork/dg-manager/v1/destinations/query ...

Status: 200 OK Time: 69 ms Size: 2.15 KB

```
{
    "data": [
        {
            "uuid": "c2a8fba8-8363-3d22-b0c2-a9e449693fae",
            "name": "CW_KAFKA_DESTINATION",
            "connectivity_info": [
                {
                    "type": "ROBOT_MSVC_TRANS_KAFKA",
                    "inaddrs": [
```

## Step 11

Click on the “**eye**” icon on the top-right of the window, then scroll down the variable list till you see **kafka_uuid** as sown in the picture. You should see current value being populated with kafka uuid.



```
{
    "data": [
        {
            "uuid": "c2a8fba8-8363
            "name": "CW_KAFKA_DEST
            "connectivity_info": [
                {
                    "type": "ROBOT
                    "ipaddrs": [
                        {
```

**Crosswork 4.0** Edit

| VARIABLE | INITIAL VALUE | CURRENT VALUE |
|---|---|---|
| node5_uuid | | 5c623f42-5ab1-4878-8570-94a3194d86e4 |
| node8_uuid | | 60a9ae25-8808-4f34-8905-5c022e7fee06 |
| vmxnode_uuid | | df4ffa17-a00c-4bb7-84c3-db85a9277cab |
| kafka_uuid | | f3bf7673-9850-4653-b670-5bf630427e72 |

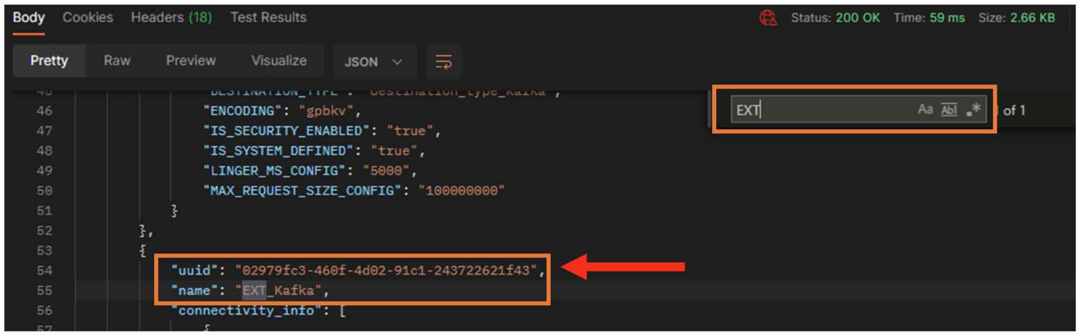
**Globals** Edit

| VARIABLE | INITIAL VALUE | CURRENT VALUE |
|---|---|---|
| baseUrl | | |
| Bearer Token | | |
| Token | | |
| cat.cw.api.ticket | | TGT-4-xHAOx-LK5r9Klkls4XVCLcFaFi-0EQVPx4CGjMUu22jjIK1ZoFu-jAbu7dyDH6aveXg-cas-0 |

**Use variables to reuse values and protect sensitive data**
Store sensitive data in variable type secret to keep its values masked on the screen. Learn more about variable type
Work with the current value of a variable to prevent sharing sensitive values with your team. Learn more about variable values

## Step 12

Click anywhere in the result (bottom) frame and press **CTRL-F**. The search text-field appears. Type **EXT** in the field. Postman will automatically go to the section in the result output where **EXT_Kafka** is returned. Look at the uuid shown there and compare its value with the one shown on the variable list seen before. They must match.

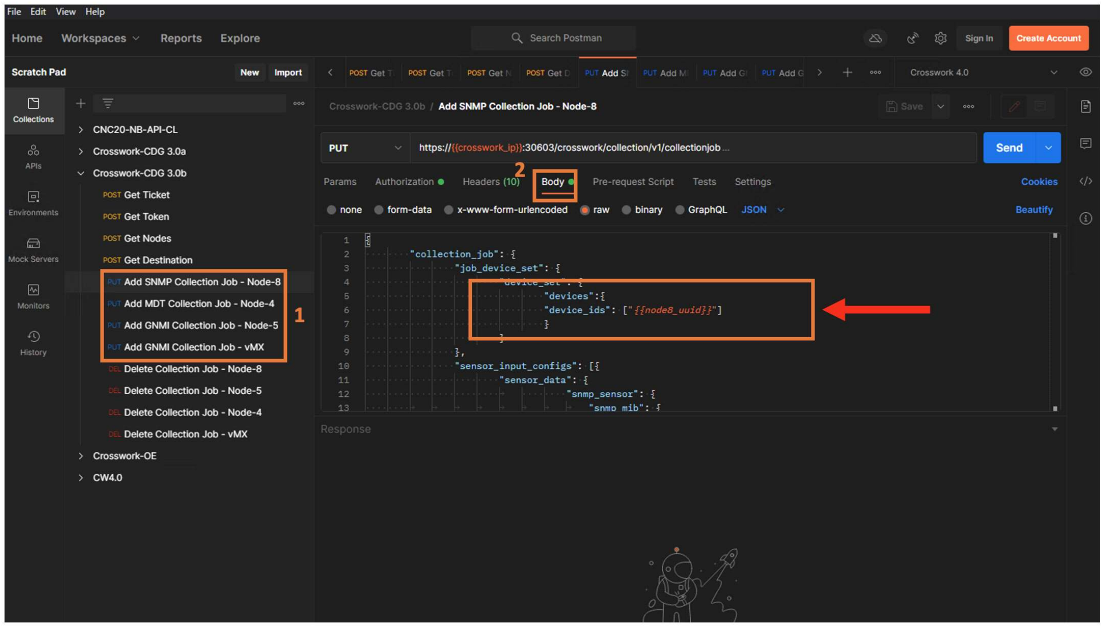


```
                "DESTINATION_TYPE": "destination_type_kafka",
                "ENCODING": "gpbkv",
                "IS_SECURITY_ENABLED": "true",
                "IS_SYSTEM_DEFINED": "true",
                "LINGER_MS_CONFIG": "5000",
                "MAX_REQUEST_SIZE_CONFIG": "100000000"
            }
        },
        {
            "uuid": "02979fc3-460f-4d02-91c1-243722621f43",
            "name": "EXT_Kafka",
            "connectivity_info": [
                {
```

## Step 13

Select one API command from the **Add Collection Job** list then select the **Body** tab. This tab provides access to the API payload content. User can notice the device_ids section is using a Postman variable rather than its actual value.

(Optional) User may want to scroll down and review the whole payload content and all its sections details.

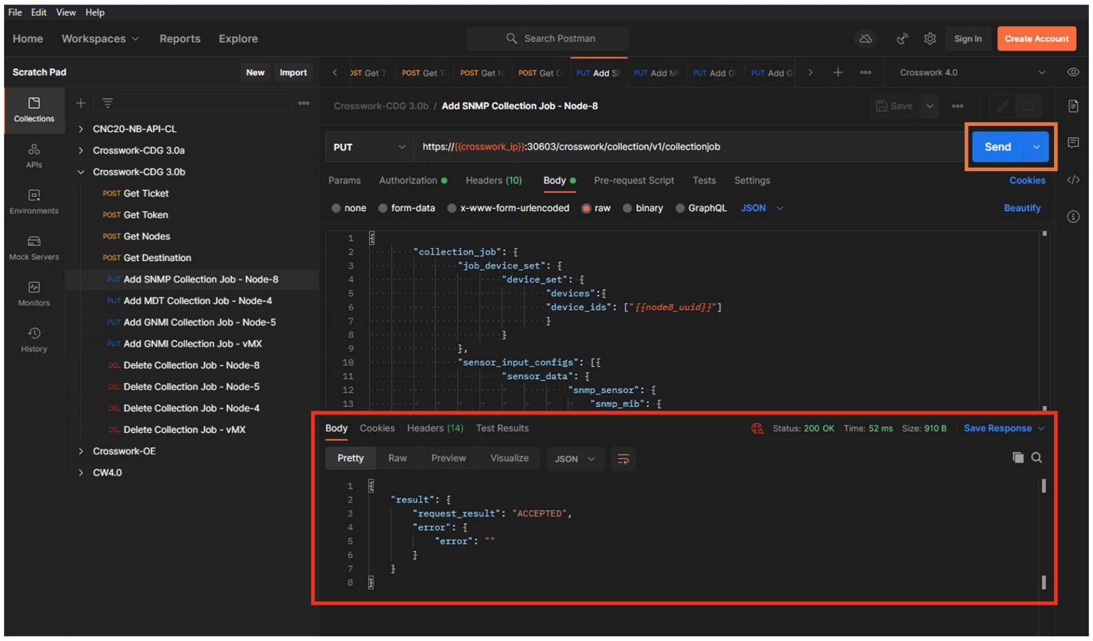


```
PUT https://{{crosswork_ip}}:30603/crosswork/collection/v1/collectionjob...
```

```
{
        "collection_job": {
                "job_device_set": {
                        "device_set": {
                                "devices":{
                                "device_ids": ["{{node8_uuid}}"]
                                }
                        }
                },
                "sensor_input_configs": [{
                        "sensor_data": {
                                "snmp_sensor": {
                                        "snmp_mib": {
```

## Step 14

Click **Send**. Verify **“requested_result”: “ACCEPTED”** is returned in the result section here in red.


Crosswork-CDG 3.0b / Add SNMP Collection Job - Node-8

PUT https://{{crosswork_ip}}:30603/crosswork/collection/v1/collectionjob

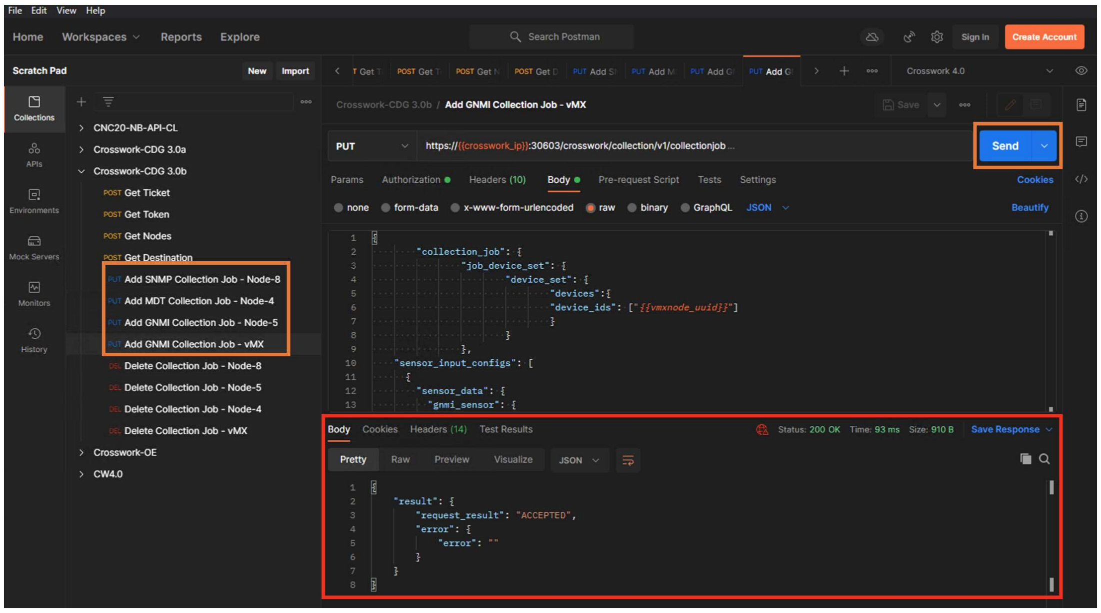
```
{
        "collection_job": {
                "job_device_set": {
                        "device_set": {
                                "devices":{
                                "device_ids": ["{{node8_uuid}}"]
                                }
                        }
                },
                "sensor_input_configs": [{
                        "sensor_data": {
                                "snmp_sensor": {
                                        "snmp_mib": {
```

Status: 200 OK  Time: 52 ms  Size: 910 B

```
{
    "result": {
        "request_result": "ACCEPTED",
        "error": {
            "error": ""
        }
    }
}
```

## Step 15

Repeat **previous** two steps for all remaining commands in the **Add Collection Job** list


Scratch Pad
CNC20-NB-API-CL
Crosswork-CDG 3.0a
Crosswork-CDG 3.0b
POST Get Ticket
POST Get Token
POST Get Nodes
POST Get Destination
PUT Add SNMP Collection Job - Node-8
PUT Add MDT Collection Job - Node-4
PUT Add GNMI Collection Job - Node-5
PUT Add GNMI Collection Job - vMX
DEL Delete Collection Job - Node-8
DEL Delete Collection Job - Node-5
DEL Delete Collection Job - Node-4
DEL Delete Collection Job - vMX
Crosswork-OE
CW4.0
Crosswork-CDG 3.0b / Add GNMI Collection Job - vMX
Send

```
PUT https://{{crosswork_ip}}:30603/crosswork/collection/v1/collectionjob ...
```

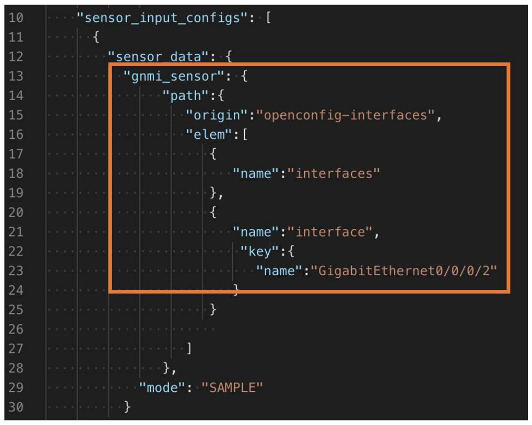
```
{
        "collection_job": {
                "job_device_set": {
                        "device_set": {
                                "devices":{
                                "device_ids": ["{{vmxnode_uuid}}"]
                                }
                        }
                },
    "sensor_input_configs": [
      {
        "sensor_data": {
          "gnmi_sensor": {
```

Status: 200 OK Time: 93 ms Size: 910 B

```
{
    "result": {
        "request_result": "ACCEPTED",
        "error": {
            "error": ""
        }
    }
}
```

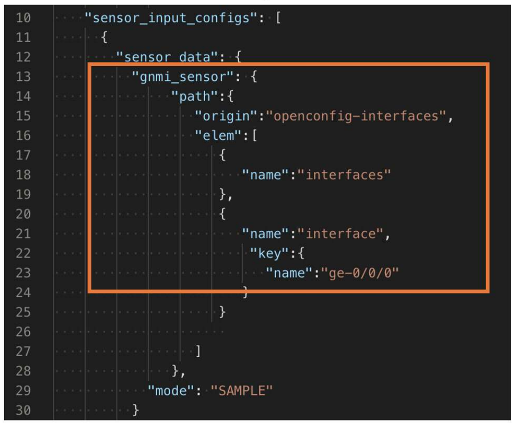
It **worth** to highlight in the payload body for **Add gNMI Collection Job – Node-5** and **Add gNMI Collection Job - vMX**, the gNMI sensor for the Cisco and Juniper devices is exactly the same. The only difference is the selected device's interface.

**Cisco**



```
    "sensor_input_configs": [
      {
        "sensor_data": {
          "gnmi_sensor": {
               "path":{
                  "origin":"openconfig-interfaces",
                  "elem":[
                    {
                       "name":"interfaces"
                    },
                    {
                       "name":"interface",
                        "key":{
                          "name":"GigabitEthernet0/0/0/2"
                        }
                    }

                  ]
               },
            "mode": "SAMPLE"
          }
```

**Juniper**

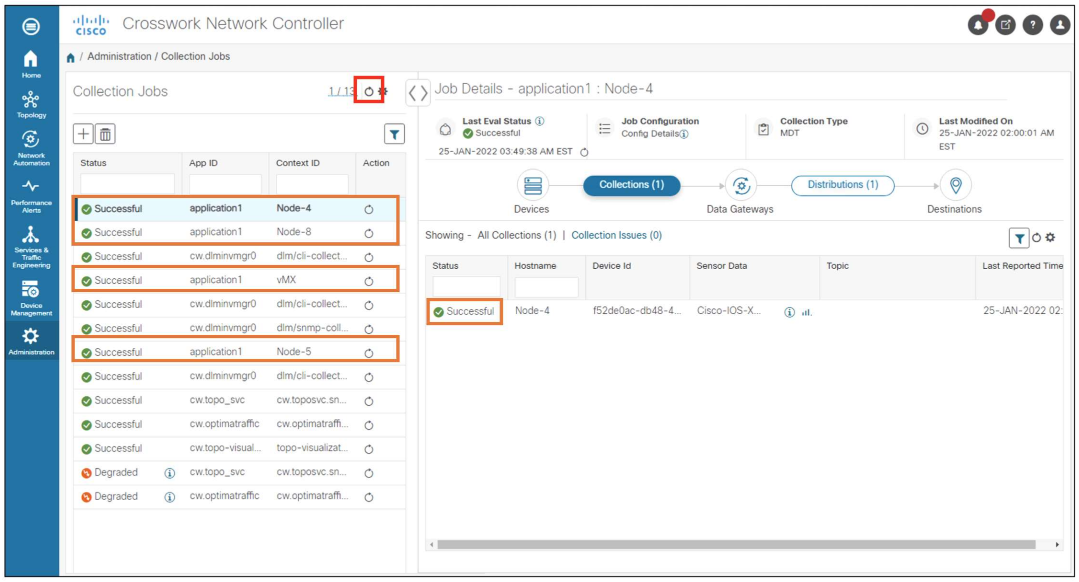


```
    "sensor_input_configs": [
      {
        "sensor_data": {
          "gnmi_sensor": {
               "path":{
                  "origin":"openconfig-interfaces",
                  "elem":[
                    {
                       "name":"interfaces"
                    },
                    {
                       "name":"interface",
                        "key":{
                          "name":"ge-0/0/0"
                        }
                    }

                  ]
               },
            "mode": "SAMPLE"
          }
```

For these collection jobs, we have been using an OpenConfig data model (openconfig-interfaces). This enables customers to use Crosswork API, independently from the vendor. OpenConfig data model support may change based on the platform and software version as well.

## Step 16

Navigate back to the Crosswork UI and navigate to **Administration > Collection Jobs** to check collection jobs just created. If collection job list was already open, click the refresh button to update the list. Verify the above collection jobs have been created. Those will have **application1** in the **App ID** column and **Node-4**, **Node-5**, **Node-8** and **vMX** in the **Context ID** column. Verify all are reported with status **Successful** on the list.



Collection Jobs 1 / 13

| Status | App ID | Context ID | Action |
|---|---|---|---|
| Successful | application1 | Node-4 | |
| Successful | application1 | Node-8 | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | application1 | vMX | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.dlminvmgr0 | dlm/snmp-coll... | |
| Successful | application1 | Node-5 | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.topo_svc | cw.toposvc.sn... | |
| Successful | cw.optimatraffic | cw.optimatraffi... | |
| Successful | cw.topo-visual... | topo-visualizat... | |
| Degraded | cw.topo_svc | cw.toposvc.sn... | |
| Degraded | cw.optimatraffic | cw.optimatraffi... | |

Job Details - application1 : Node-4

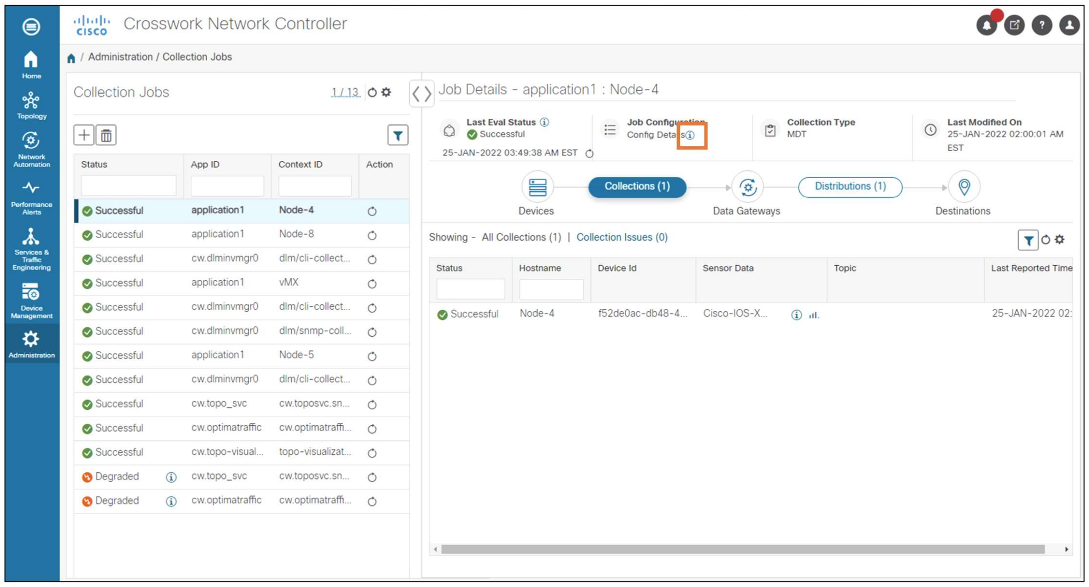
Last Eval Status: Successful
25-JAN-2022 03:49:38 AM EST
Job Configuration: Config Details
Collection Type: MDT
Last Modified On: 25-JAN-2022 02:00:01 AM EST

Showing - All Collections (1) | Collection Issues (0)

| Status | Hostname | Device Id | Sensor Data | Topic | Last Reported Time |
|---|---|---|---|---|---|
| Successful | Node-4 | f52de0ac-db48-4... | Cisco-IOS-X... | | 25-JAN-2022 02: |

## Step 17

In the left pane, select one of those collection jobs just created and click the "**i**" icon close to "**Config Details**"



Crosswork Network Controller

Administration / Collection Jobs

Collection Jobs 1 / 13

| Status | App ID | Context ID | Action |
|---|---|---|---|
| Successful | application1 | Node-4 | |
| Successful | application1 | Node-8 | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | application1 | vMX | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.dlminvmgr0 | dlm/snmp-coll... | |
| Successful | application1 | Node-5 | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.topo_svc | cw.toposvc.sn... | |
| Successful | cw.optimatraffic | cw.optimatraffi... | |
| Successful | cw.topo-visual... | topo-visualizat... | |
| Degraded | cw.topo_svc | cw.toposvc.sn... | |
| Degraded | cw.optimatraffic | cw.optimatraffi... | |

Job Details - application1 : Node-4

Last Eval Status
Successful
25-JAN-2022 03:49:38 AM EST

Job Configuration
Config Details

Collection Type
MDT

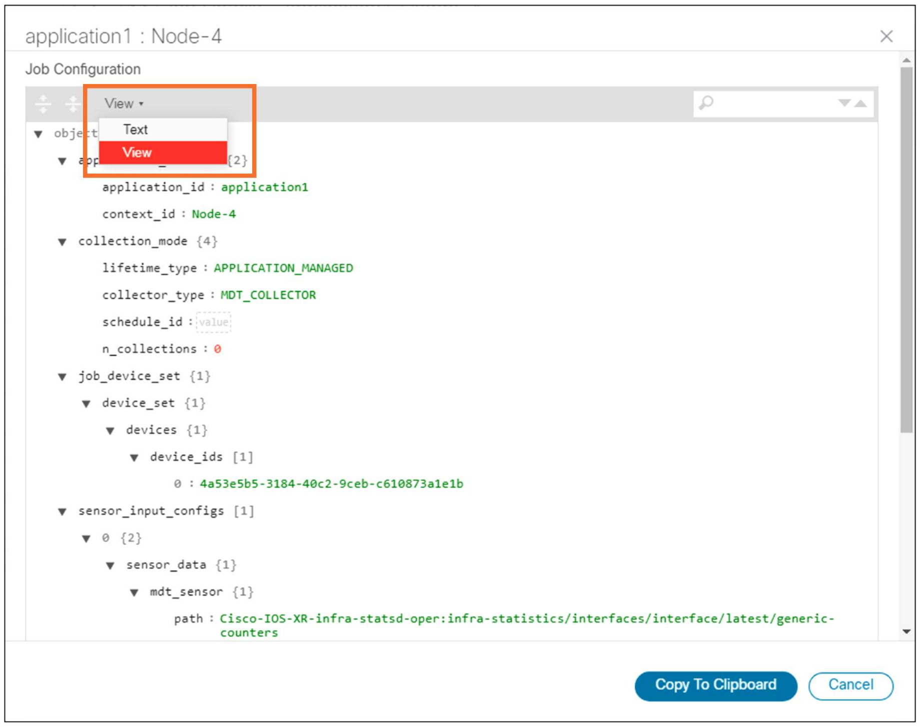
Last Modified On
25-JAN-2022 02:00:01 AM EST

Showing - All Collections (1) | Collection Issues (0)

| Status | Hostname | Device Id | Sensor Data | Topic | Last Reported Time |
|---|---|---|---|---|---|
| Successful | Node-4 | f52de0ac-db48-4... | Cisco-IOS-X... | | 25-JAN-2022 02: |

## Step 18

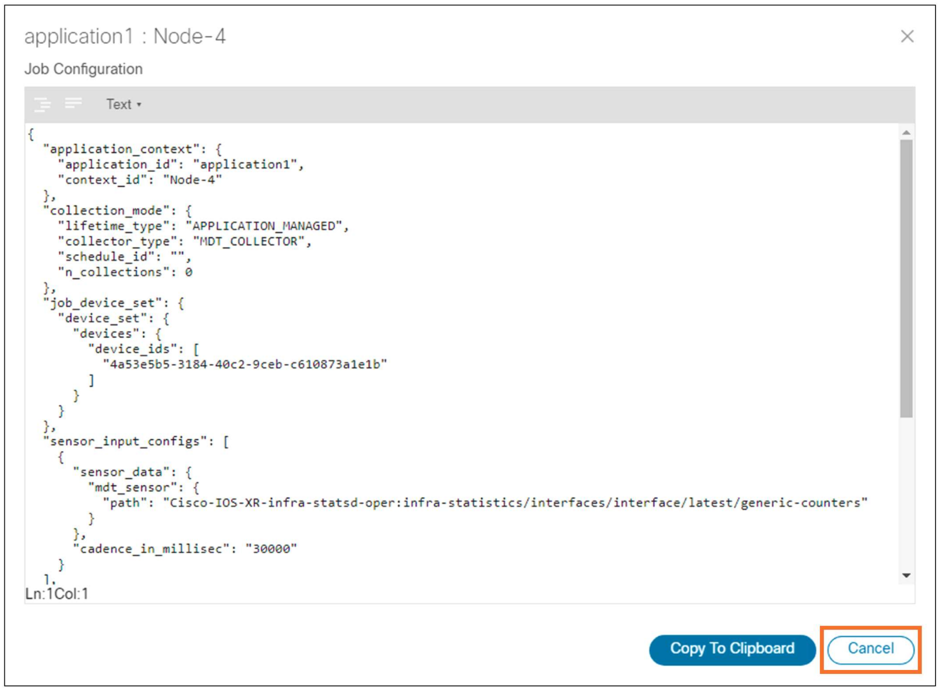
The Collection Job **Config Details** opens. Expand **View** and select **Text**



```
▼ object
  ▼ app... {2}
      application_id : application1
      context_id : Node-4
  ▼ collection_mode {4}
      lifetime_type : APPLICATION_MANAGED
      collector_type : MDT_COLLECTOR
      schedule_id : value
      n_collections : 0
  ▼ job_device_set {1}
    ▼ device_set {1}
      ▼ devices {1}
        ▼ device_ids [1]
            0 : 4a53e5b5-3184-40c2-9ceb-c610873a1e1b
  ▼ sensor_input_configs [1]
    ▼ 0 {2}
      ▼ sensor_data {1}
        ▼ mdt_sensor {1}
            path : Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/latest/generic-
                   counters
```

## Step 19

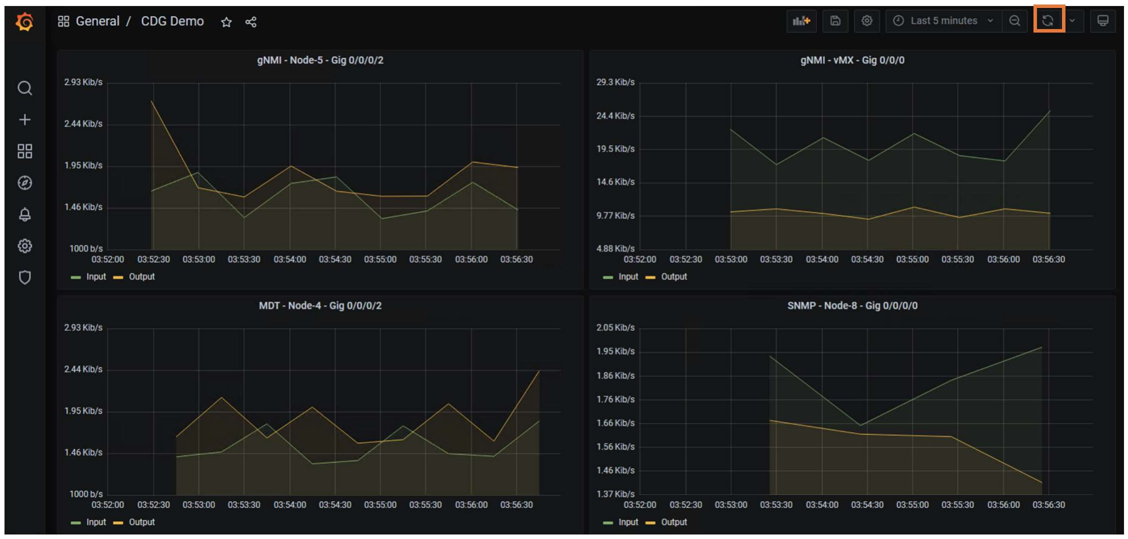
A payload similar to the one used in Postman will appear here. Review content as needed, then click on **Cancel**.



```
{
  "application_context": {
    "application_id": "application1",
    "context_id": "Node-4"
  },
  "collection_mode": {
    "lifetime_type": "APPLICATION_MANAGED",
    "collector_type": "MDT_COLLECTOR",
    "schedule_id": "",
    "n_collections": 0
  },
  "job_device_set": {
    "device_set": {
      "devices": {
        "device_ids": [
          "4a53e5b5-3184-40c2-9ceb-c610873a1e1b"
        ]
      }
    }
  },
  "sensor_input_configs": [
    {
      "sensor_data": {
        "mdt_sensor": {
          "path": "Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/latest/generic-counters"
        }
      },
      "cadence_in_millisec": "30000"
    }
  ],
Ln:1Col:1
```

## Step 20

Navigate back to **Grafana**. Wait a few minutes and click the refresh button in the upper right-hand corner of the window. Verify that all graphs are being populated. You may have to click the refresh button a few times to see the change.


General / CDG Demo
Last 5 minutes
gNMI - Node-5 - Gig 0/0/0/2
gNMI - vMX - Gig 0/0/0
MDT - Node-4 - Gig 0/0/0/2
SNMP - Node-8 - Gig 0/0/0/0
Input
Output

## Data Gateway Messages

The Data Gateway message format follows a protobuf proto definition. The Data Gateway proto can be compiled using multiple programming languages or customer choice.
The Data Gateway proto is required to parse messages that the Data Gateway posts on the Kafka or gRPC messaging bus.

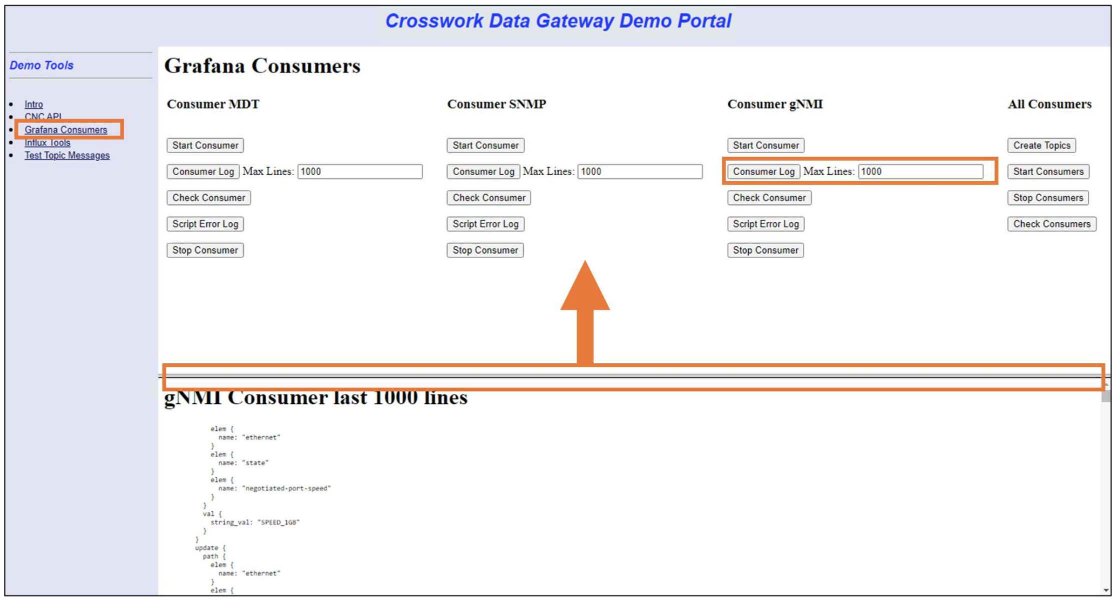
Here an extract:

```
syntax = "proto3";
package output;

option java_package = "com.cisco.dg.protobuf.output";
option java_outer_classname = "TelemetryDataOutput";
option go_package = "telemetry";
import "thirdparty/gnmi/gnmi.proto";

message Telemetry {
    oneof node_id {
        string node_id_str = 1;
    }
    bytes node_id_uuid = 2;
    oneof subscription {
        string subscription_id_str = 3;
    }
    string encoding_path = 6;
    uint64 collection_id = 8;
    uint64 collection_start_time = 9;
    uint64 msg_timestamp = 10;
    repeated TelemetryField data_gpbkv = 11;
    TelemetryGPBTable data_gpb = 12;
    uint64 collection_end_time = 13;
    bytes collector_uuid = 15;
    CollectionStatus status = 16;
    ModelData model_data = 17;
    SensorData sensor_data = 18;
    repeated ApplicationContext application_contexts = 19;
    string version = 20;
}

message ApplicationContext {
    string application_id = 1;
    string context_id = 2;
}

message TelemetryField {
    uint64 timestamp = 1;
    string name = 2;
    oneof value_by_type {
        bytes bytes_value = 4;
        string string_value = 5;
        bool bool_value = 6;
        uint32 uint32_value = 7;
        uint64 uint64_value = 8;
        sint32 sint32_value = 9;
        sint64 sint64_value = 10;
        double double_value = 11;
        float float_value = 12;
        SnmpData snmp_data = 13;
        SnmpTrapData snmp_trap = 14;
        GnmiData gnmi_data = 16;
    }
    repeated TelemetryField fields = 15;
}

...
...
```

Now we will take a closer look on the actual messages in our lab.

## Step 21

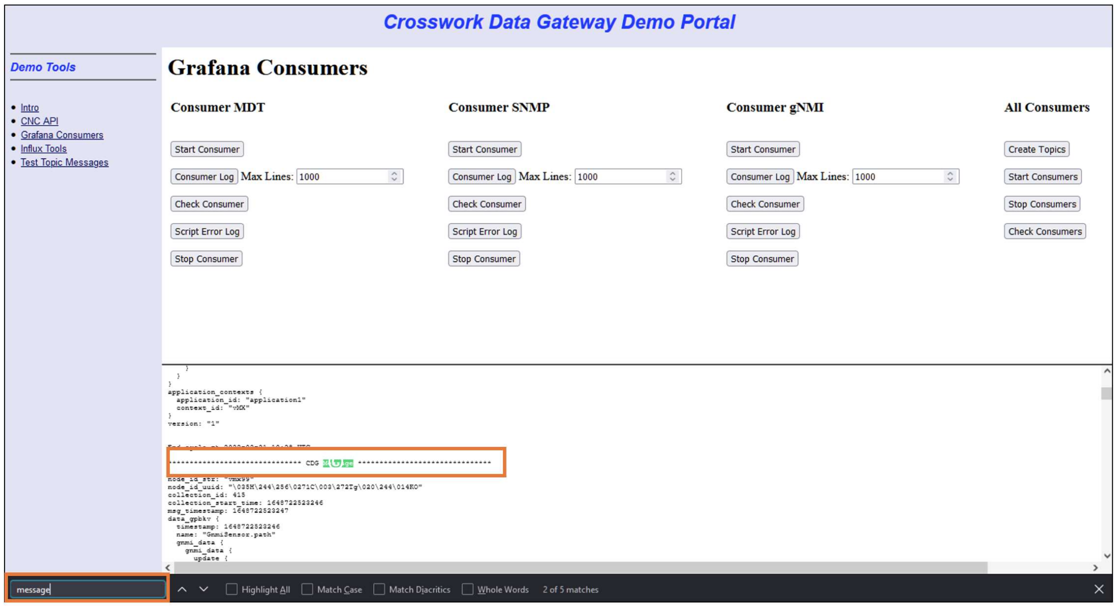
Navigate to the **CDG Demo Portal**, click **Grafana Consumers** in the left pane, and then click one of the three **Consumer Log** buttons from the three available. You may play with the number of lines to extend or reduce log output. In the following case, we selected the gNMI **Consumer Log** with the default 1000 lines. You can move the center bar up to extend the log view.



```
        elem {
          name: "ethernet"
        }
        elem {
          name: "state"
        }
        elem {
          name: "negotiated-port-speed"
        }
      }
      val {
        string_val: "SPEED_1GB"
      }
    }
    update {
      path {
        elem {
          name: "ethernet"
        }
        elem {
```

## Step 22

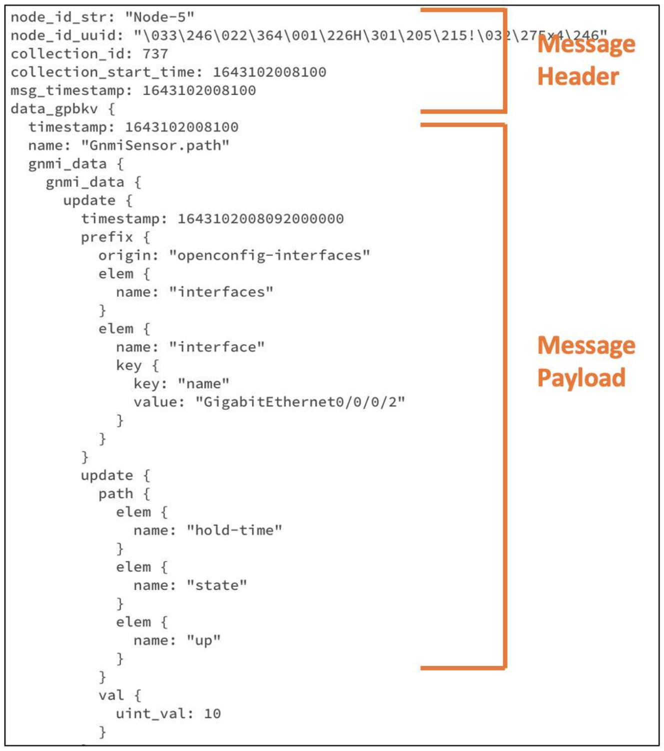
The consumer log contains both Data Gateway messages and some notes on the parsed data before storing counters on the InfluxDB. Consumer process adds a line with “**CDG Message”** string at the beginning of each message in the log. Press **CTRL+ F** and enter **Message** in the search field. This moves you to the first CDG message in the log.



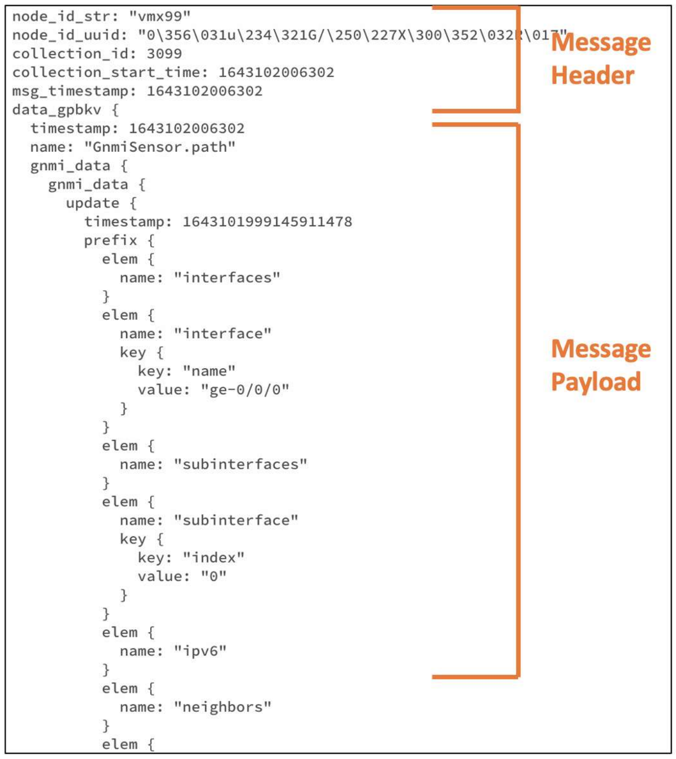
```
    }
  }
}
application_contexts {
  application_id: "application1"
  context_id: "vMX"
}
version: "1"


****************************** CDG Message ******************************
node_id_str: "vmx99"
node_id_uuid: "\035H\244\256\0271C\003\272Tg\020\244\014KO"
collection_id: 415
collection_start_time: 1648722523246
msg_timestamp: 1648722523247
data_gpbkv {
  timestamp: 1648722523246
  name: "GnmiSensor.path"
  gnmi_data {
    gnmi_data {
      update {
```

## Message Examples

The actual **message payload** included in the Data Gateway message starts with the line **data_gpbkv**



```
node_id_str: "Node-5"
node_id_uuid: "\033\246\022\364\001\226H\301\205\215!\032\275x4\246"
collection_id: 737
collection_start_time: 1643102008100
msg_timestamp: 1643102008100
data_gpbkv {
  timestamp: 1643102008100
  name: "GnmiSensor.path"
  gnmi_data {
    gnmi_data {
      update {
        timestamp: 1643102008092000000
        prefix {
          origin: "openconfig-interfaces"
          elem {
            name: "interfaces"
          }
          elem {
            name: "interface"
            key {
              key: "name"
              value: "GigabitEthernet0/0/0/2"
            }
          }
        }
        update {
          path {
            elem {
              name: "hold-time"
            }
            elem {
              name: "state"
            }
            elem {
              name: "up"
            }
          }
          val {
            uint_val: 10
          }
```



```
node_id_str: "vmx99"
node_id_uuid: "0\356\031u\234\321G/\250\227X\300\352\032R\017"
collection_id: 3099
collection_start_time: 1643102006302
msg_timestamp: 1643102006302
data_gpbkv {
  timestamp: 1643102006302
  name: "GnmiSensor.path"
  gnmi_data {
    gnmi_data {
      update {
        timestamp: 1643101999145911478
        prefix {
          elem {
            name: "interfaces"
          }
          elem {
            name: "interface"
            key {
              key: "name"
              value: "ge-0/0/0"
            }
          }
          elem {
            name: "subinterfaces"
          }
          elem {
            name: "subinterface"
            key {
              key: "index"
              value: "0"
            }
          }
          elem {
            name: "ipv6"
          }
          elem {
            name: "neighbors"
          }
          elem {
```

**This concludes task 3.**

## Task 4: Create Collection Jobs using Crosswork UI

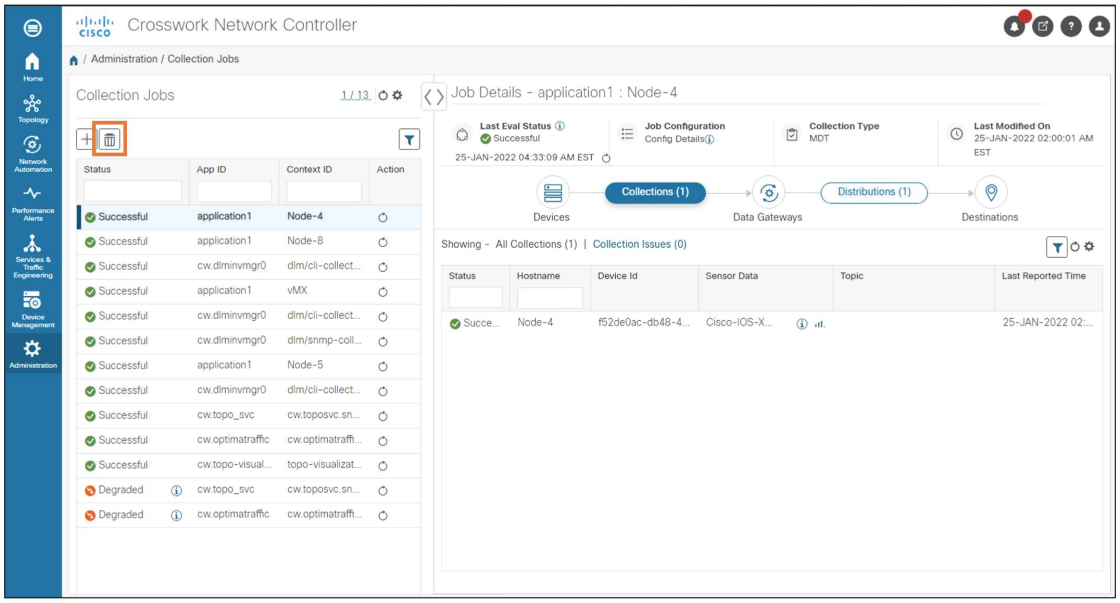
Crosswork Data Gateway provides an option to create collection jobs for external destination directly from its GUI. Currently, this is limited to CLI or SNMP protocols only. The objective for this task is to practice with this feature and create a collection job for external destination using SNMP as collection protocol.

If you ran the whole task 3, you should have four user created collections jobs running. Let's delete them again, this time using Crosswork UI.

### Step 1

Go back to the Crosswork UI and navigate to **Administration > Collection Jobs**. One by one, select the entry for the jobs with **application1** in the **App ID** column and **Node-4**, **Node-5**, **Node-**8 and **vMX** in the **Context ID** column, and click the

**delete** icon.


Crosswork Network Controller

Administration / Collection Jobs

Collection Jobs 1 / 13

| Status | App ID | Context ID | Action |
|---|---|---|---|
| Successful | application1 | Node-4 | |
| Successful | application1 | Node-8 | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | application1 | vMX | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.dlminvmgr0 | dlm/snmp-coll... | |
| Successful | application1 | Node-5 | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.topo_svc | cw.toposvc.sn... | |
| Successful | cw.optimatraffic | cw.optimatraffi... | |
| Successful | cw.topo-visual... | topo-visualizat... | |
| Degraded | cw.topo_svc | cw.toposvc.sn... | |
| Degraded | cw.optimatraffic | cw.optimatraffi... | |

Job Details - application1 : Node-4

Last Eval Status: Successful, 25-JAN-2022 04:33:09 AM EST

Job Configuration: Config Details

Collection Type: MDT

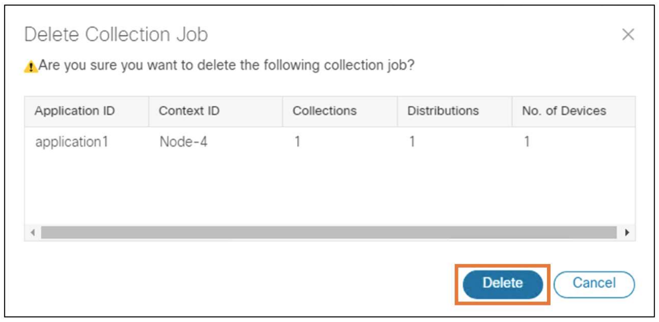
Last Modified On: 25-JAN-2022 02:00:01 AM EST

Showing - All Collections (1) | Collection Issues (0)

| Status | Hostname | Device Id | Sensor Data | Topic | Last Reported Time |
|---|---|---|---|---|---|
| Succe... | Node-4 | f52de0ac-db48-4... | Cisco-IOS-X... | | 25-JAN-2022 02:... |

## Step 2

Click **Delete** on the confirmation dialog.

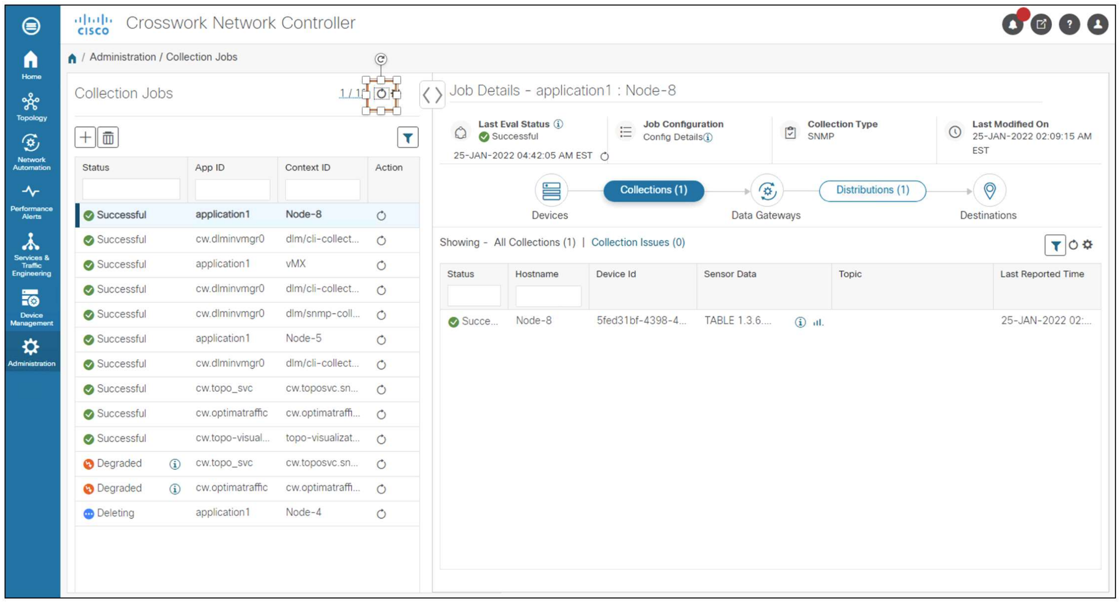

Delete Collection Job

Are you sure you want to delete the following collection job?

| Application ID | Context ID | Collections | Distributions | No. of Devices |
|---|---|---|---|---|
| application1 | Node-4 | 1 | 1 | 1 |

## Step 3

Click on the refresh icon as you may end up with the list not updated (like in the following snapshot) and a collection job might be pending with **Deleting** state.



| Status | App ID | Context ID | Action |
|---|---|---|---|
| Successful | application1 | Node-8 | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | application1 | vMX | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.dlminvmgr0 | dlm/snmp-coll... | |
| Successful | application1 | Node-5 | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.topo_svc | cw.toposvc.sn... | |
| Successful | cw.optimatraffic | cw.optimatraffi... | |
| Successful | cw.topo-visual... | topo-visualizat... | |
| Degraded | cw.topo_svc | cw.toposvc.sn... | |
| Degraded | cw.optimatraffic | cw.optimatraffi... | |
| Deleting | application1 | Node-4 | |

Job Details - application1 : Node-8

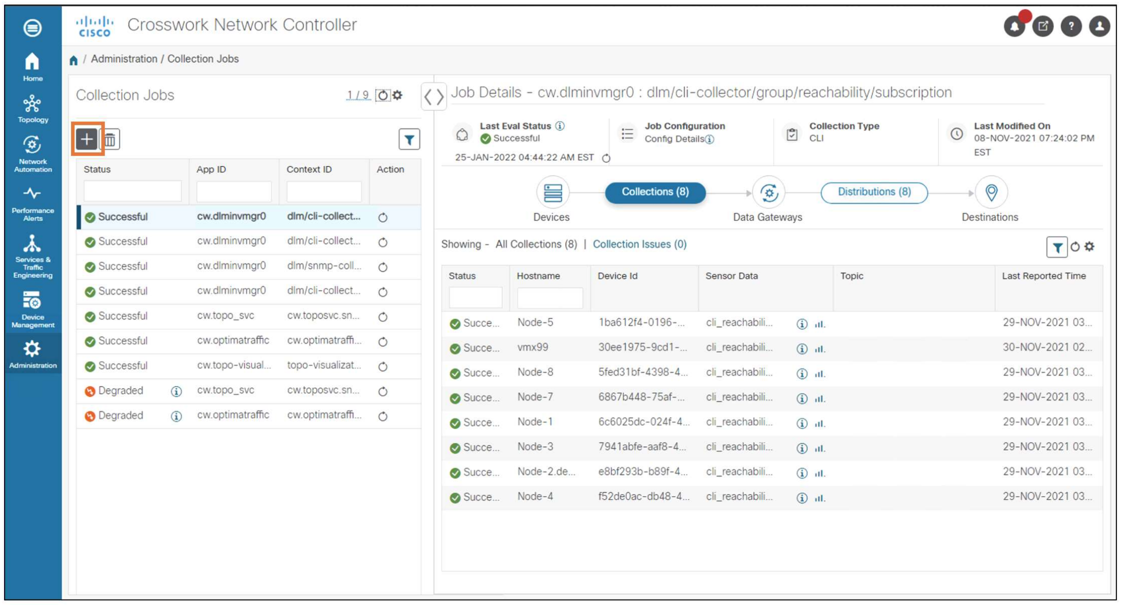
Last Eval Status: Successful
25-JAN-2022 04:42:05 AM EST
Job Configuration: Config Details
Collection Type: SNMP
Last Modified On: 25-JAN-2022 02:09:15 AM EST

Showing - All Collections (1) | Collection Issues (0)

| Status | Hostname | Device Id | Sensor Data | Topic | Last Reported Time |
|---|---|---|---|---|---|
| Succe... | Node-8 | 5fed31bf-4398-4... | TABLE 1.3.6.... | | 25-JAN-2022 02:... |

## Step 4

Now click **Add Job** (**+**).


Crosswork Network Controller
Home
Topology
Network Automation
Performance Alerts
Services & Traffic Engineering
Device Management
Administration
Administration / Collection Jobs

Collection Jobs 1 / 9

| Status | App ID | Context ID | Action |
|---|---|---|---|
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.dlminvmgr0 | dlm/snmp-coll... | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.topo_svc | cw.toposvc.sn... | |
| Successful | cw.optimatraffic | cw.optimatraffi... | |
| Successful | cw.topo-visual... | topo-visualizat... | |
| Degraded | cw.topo_svc | cw.toposvc.sn... | |
| Degraded | cw.optimatraffic | cw.optimatraffi... | |

Job Details - cw.dlminvmgr0 : dlm/cli-collector/group/reachability/subscription

Last Eval Status: Successful
25-JAN-2022 04:44:22 AM EST

Job Configuration: Config Details

Collection Type: CLI

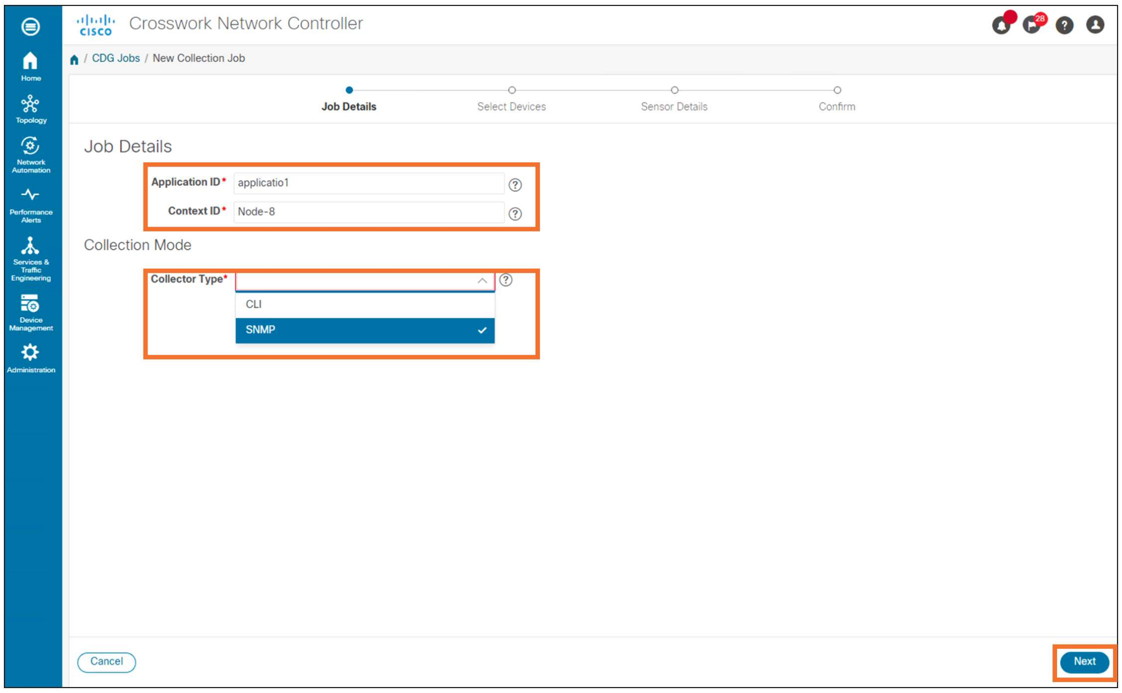
Last Modified On: 08-NOV-2021 07:24:02 PM EST

Showing - All Collections (8) | Collection Issues (0)

| Status | Hostname | Device Id | Sensor Data | Topic | Last Reported Time |
|---|---|---|---|---|---|
| Succe... | Node-5 | 1ba612f4-0196-... | cli_reachabili... | | 29-NOV-2021 03... |
| Succe... | vmx99 | 30ee1975-9cd1-... | cli_reachabili... | | 30-NOV-2021 02... |
| Succe... | Node-8 | 5fed31bf-4398-4... | cli_reachabili... | | 29-NOV-2021 03... |
| Succe... | Node-7 | 6867b448-75af-... | cli_reachabili... | | 29-NOV-2021 03... |
| Succe... | Node-1 | 6c6025dc-024f-4... | cli_reachabili... | | 29-NOV-2021 03... |
| Succe... | Node-3 | 7941abfe-aaf8-4... | cli_reachabili... | | 29-NOV-2021 03... |
| Succe... | Node-2.de... | e8bf293b-b89f-4... | cli_reachabili... | | 29-NOV-2021 03... |
| Succe... | Node-4 | f52de0ac-db48-4... | cli_reachabili... | | 29-NOV-2021 03... |

## Step 5

In the Application ID and Context ID, type **application1** (no spaces) and **Node-8**, respectively. Select **SNMP** form the **Collector Type** pull-down menu. Then click **Next**.



Job Details

Application ID* applicatio1

Context ID* Node-8

Collection Mode

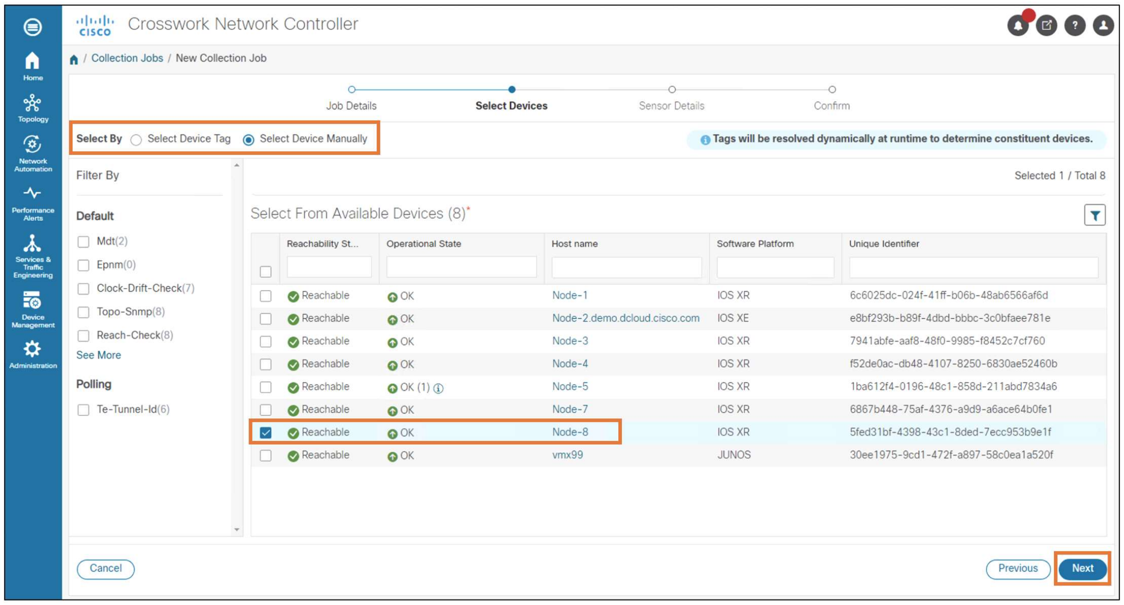
Collector Type*

CLI

SNMP ✓

## Step 6

Click the **Select Device Manually** radio button, then select **Node-8** from the available devices. Click **Next**.



Select By [ ] Select Device Tag [✓ Select Device Manually]

Tags will be resolved dynamically at runtime to determine constituent devices.

Filter By

Default

- Mdt(2)
- Epnm(0)
- Clock-Drift-Check(7)
- Topo-Snmp(8)
- Reach-Check(8)

See More

Polling

- Te-Tunnel-Id(6)

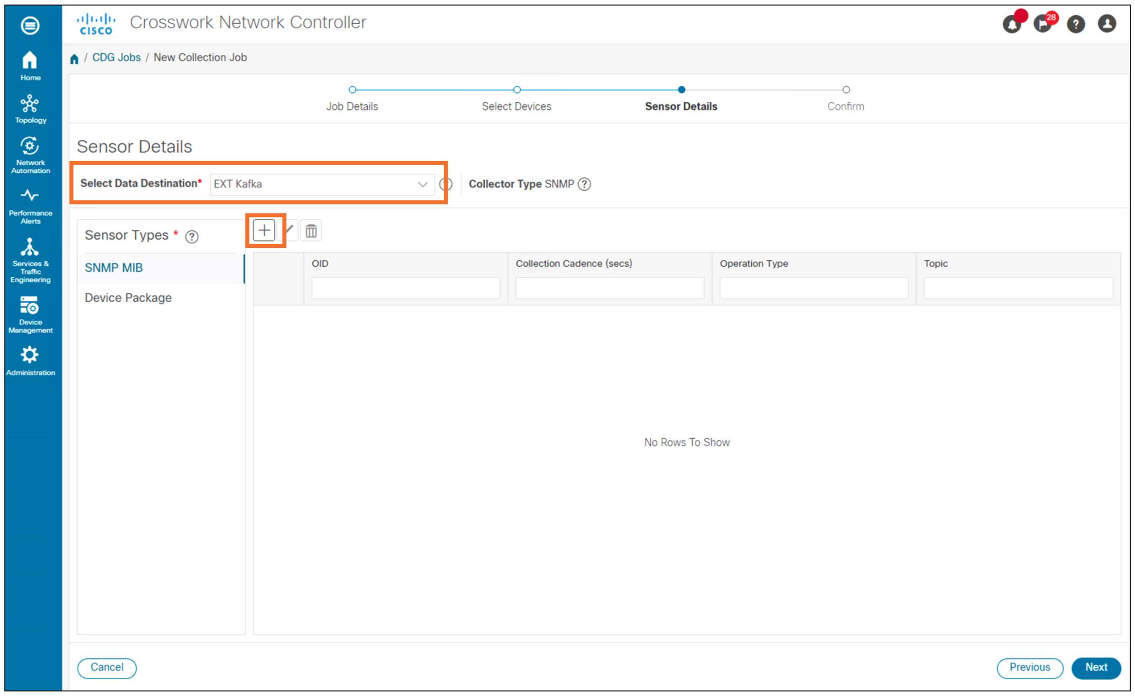
Selected 1 / Total 8

Select From Available Devices (8)*

| | Reachability St... | Operational State | Host name | Software Platform | Unique Identifier |
|---|---|---|---|---|---|
| ☐ | Reachable | OK | Node-1 | IOS XR | 6c6025dc-024f-41ff-b06b-48ab6566af6d |
| ☐ | Reachable | OK | Node-2.demo.dcloud.cisco.com | IOS XE | e8bf293b-b89f-4dbd-bbbc-3c0bfaee781e |
| ☐ | Reachable | OK | Node-3 | IOS XR | 7941abfe-aaf8-48f0-9985-f8452c7cf760 |
| ☐ | Reachable | OK | Node-4 | IOS XR | f52de0ac-db48-4107-8250-6830ae52460b |
| ☐ | Reachable | OK (1) | Node-5 | IOS XR | 1ba612f4-0196-48c1-858d-211abd7834a6 |
| ☐ | Reachable | OK | Node-7 | IOS XR | 6867b448-75af-4376-a9d9-a6ace64b0fe1 |
| ✓ | Reachable | OK | Node-8 | IOS XR | 5fed31bf-4398-43c1-8ded-7ecc953b9e1f |
| ☐ | Reachable | OK | vmx99 | JUNOS | 30ee1975-9cd1-472f-a897-58c0ea1a520f |

## Step 7

Select **EXT_Kafka** from the **Select Data Destination** pull-down menu. Then click **add** (**+**).

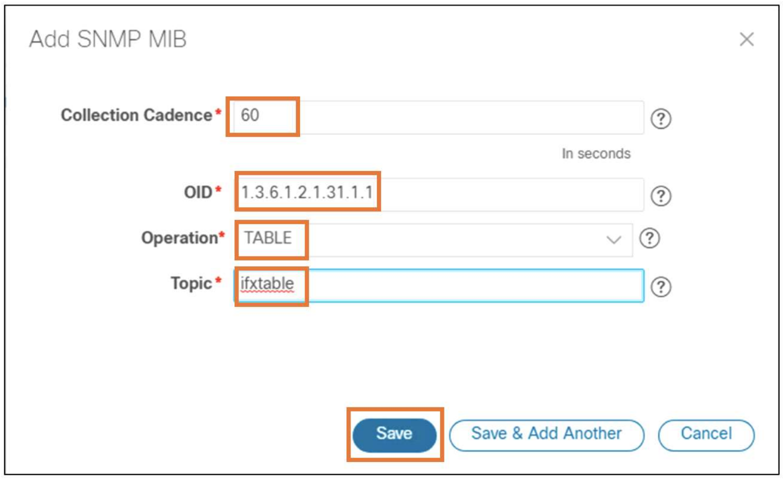


## Step 8

Make the following entries:

- **Collection Cadence**: leave default **60** secs.
- **OID**: **1.3.6.1.2.1.31.1.1** (you can copy paste from here)
- **Operation Type**: Select **TABLE** from the operation pull-down menu.
- **Topic**: **ifxtable** (you can copy paste from here)

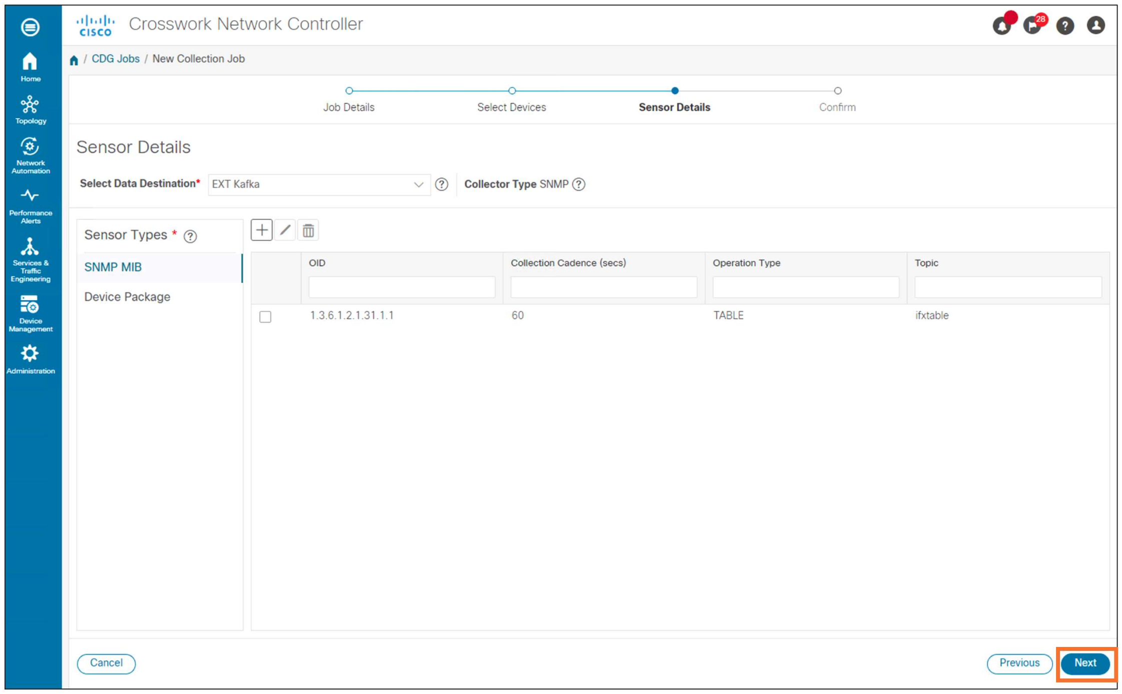
Then click **Save**



| | |
|---|---|
| Collection Cadence* | 60 |
| | In seconds |
| OID* | 1.3.6.1.2.1.31.1.1 |
| Operation* | TABLE |
| Topic* | ifxtable |

## Step 9

In this exercise we want a one SNMP operation limit. So just click **Next**.



Sensor Details

Select Data Destination* EXT Kafka

Collector Type SNMP

Sensor Types *

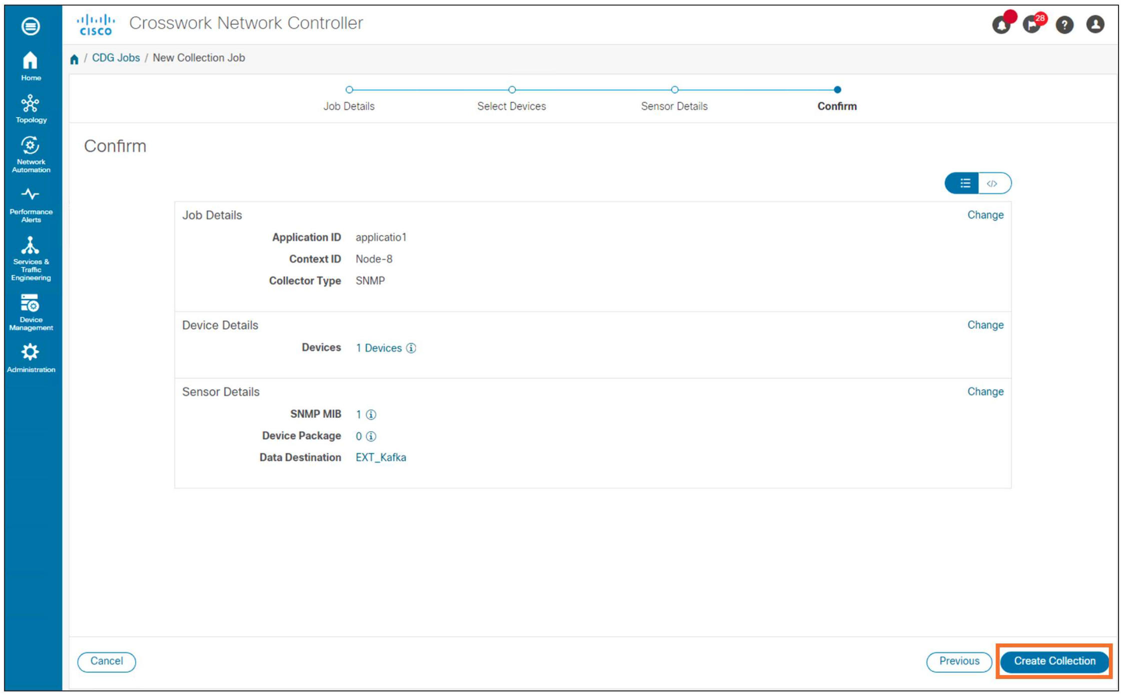
SNMP MIB

Device Package

| OID | Collection Cadence (secs) | Operation Type | Topic |
|---|---|---|---|
| 1.3.6.1.2.1.31.1.1 | 60 | TABLE | ifxtable |

## Step 10

Review the **Confirm** dialog then click on **Create Collection**



Confirm

Job Details

| | |
|---|---|
| Application ID | applicatio1 |
| Context ID | Node-8 |
| Collector Type | SNMP |

Device Details

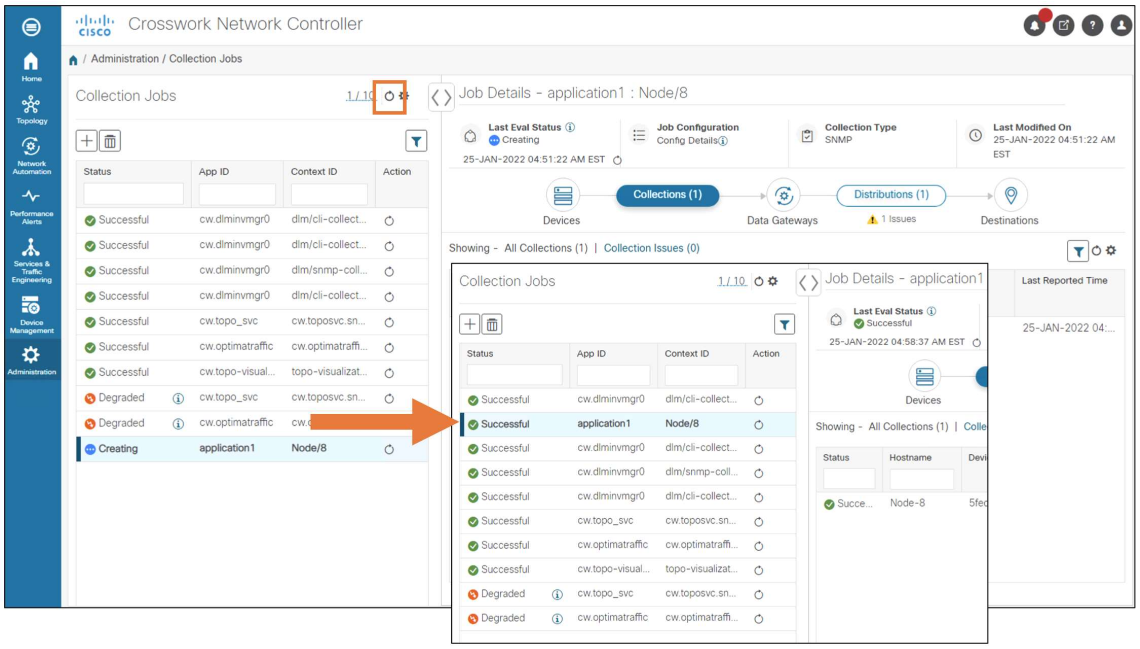
| | |
|---|---|
| Devices | 1 Devices |

Sensor Details

| | |
|---|---|
| SNMP MIB | 1 |
| Device Package | 0 |
| Data Destination | EXT_Kafka |

## Step 11

Refresh the job list. Verify the created collection job is reported as **Successful**.



| Status | App ID | Context ID | Action |
|---|---|---|---|
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.dlminvmgr0 | dlm/snmp-coll... | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.topo_svc | cw.toposvc.sn... | |
| Successful | cw.optimatraffic | cw.optimatraffi... | |
| Successful | cw.topo-visual... | topo-visualizat... | |
| Degraded | cw.topo_svc | cw.toposvc.sn... | |
| Degraded | cw.optimatraffic | cw. | |
| Creating | application1 | Node/8 | |

Job Details - application1 : Node/8

Last Eval Status: Creating
25-JAN-2022 04:51:22 AM EST

Job Configuration: Config Details

Collection Type: SNMP

Last Modified On: 25-JAN-2022 04:51:22 AM EST

Showing - All Collections (1) | Collection Issues (0)

| Last Reported Time |
|---|
| 25-JAN-2022 04:... |

| Status | App ID | Context ID | Action |
|---|---|---|---|
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | application1 | Node/8 | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.dlminvmgr0 | dlm/snmp-coll... | |
| Successful | cw.dlminvmgr0 | dlm/cli-collect... | |
| Successful | cw.topo_svc | cw.toposvc.sn... | |
| Successful | cw.optimatraffic | cw.optimatraffi... | |
| Successful | cw.topo-visual... | topo-visualizat... | |
| Degraded | cw.topo_svc | cw.toposvc.sn... | |
| Degraded | cw.optimatraffic | cw.optimatraffi... | |

Job Details - application1

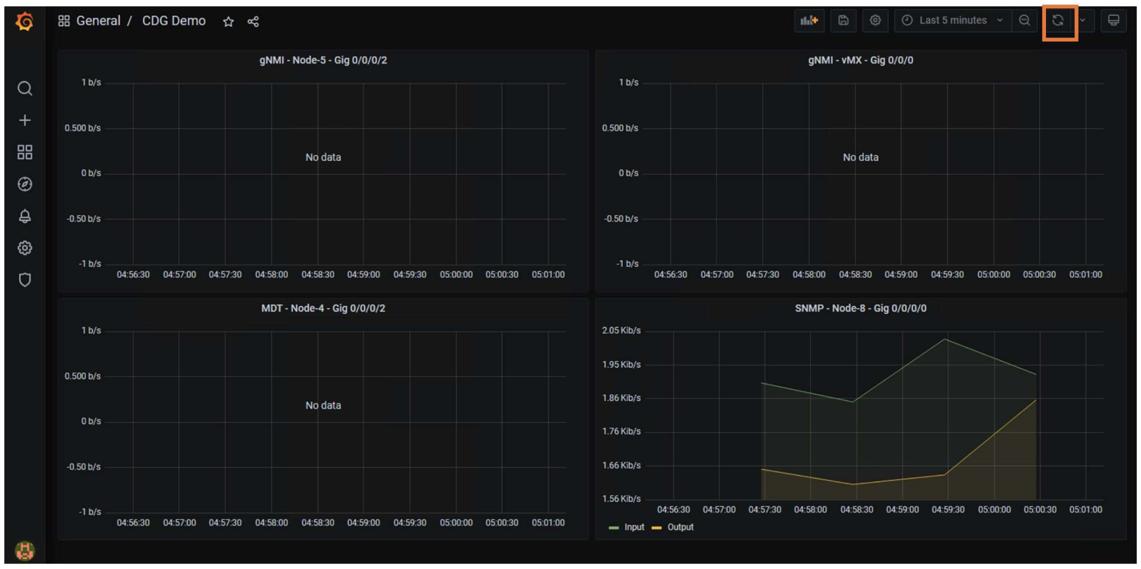
Last Eval Status: Successful
25-JAN-2022 04:58:37 AM EST

Showing - All Collections (1) | Colle

| Status | Hostname | Devi |
|---|---|---|
| Succe... | Node-8 | 5fe |

## Step 12

(Optional step) Navigate back to Grafana and click on the **Refresh** icon. Verify SNMP graph is being populated.

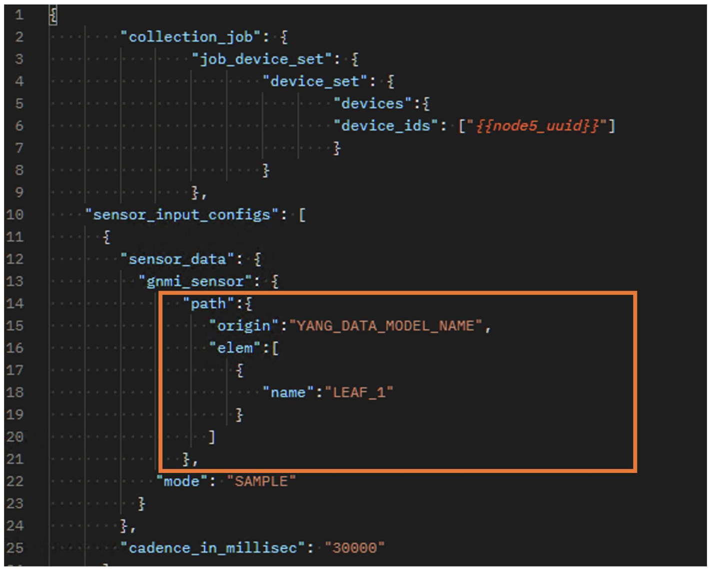


**This concludes task 4.**

## Task 5: Editing a gNMI collection job payload

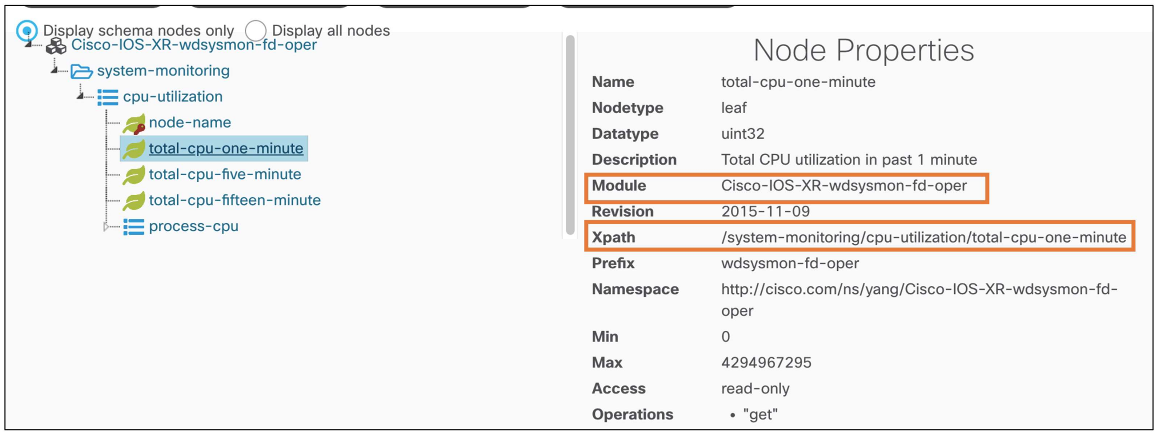
If you reviewed past collection jobs payloads, you have seen input sensors for **SNMP** and **MDT** collection jobs are made by a simple **SNMP OID** or **MDT sensor path**. When using **gNMI**, input sensor is a **JSON object** made of a hierarchical sequence of name/value pairs and key filters.

In this task, we will start from a collection job payload partially populated with dummy values for the input and output sensor configs sections



```
{
        "collection_job": {
                "job_device_set": {
                        "device_set": {
                                "devices":{
                                "device_ids": ["{{node5_uuid}}"]
                                }
                        }
                },
    "sensor_input_configs": [
      {
        "sensor_data": {
          "gnmi_sensor": {
              "path":{
                 "origin":"YANG_DATA_MODEL_NAME",
                 "elem":[
                    {
                        "name":"LEAF_1"
                    }
                 ]
              },
            "mode": "SAMPLE"
          }
        },
        "cadence_in_millisec": "30000"
```

We will update it to collect **"1 minute CPU counter"** KPI and use it on **Node-5**. To that end, we will be using Cisco-IOS-XR-wdsysmon-fd-oper YANG data model and the Xpath highlighted below



Node Properties

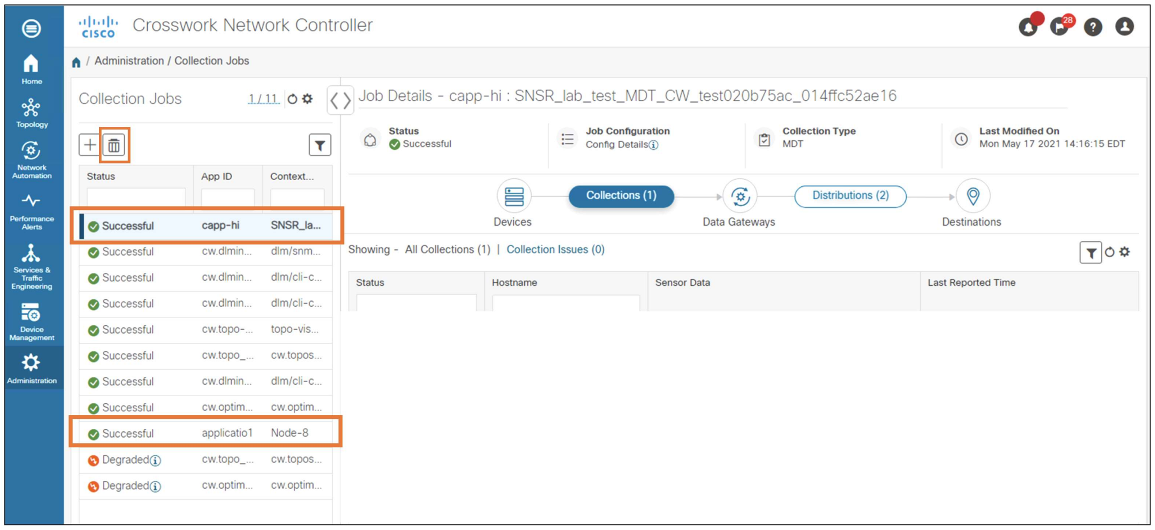
| | |
|---|---|
| **Name** | total-cpu-one-minute |
| **Nodetype** | leaf |
| **Datatype** | uint32 |
| **Description** | Total CPU utilization in past 1 minute |
| **Module** | Cisco-IOS-XR-wdsysmon-fd-oper |
| **Revision** | 2015-11-09 |
| **Xpath** | /system-monitoring/cpu-utilization/total-cpu-one-minute |
| **Prefix** | wdsysmon-fd-oper |
| **Namespace** | http://cisco.com/ns/yang/Cisco-IOS-XR-wdsysmon-fd-oper |
| **Min** | 0 |
| **Max** | 4294967295 |
| **Access** | read-only |
| **Operations** | • "get" |

Following rule will be used to map it to API payload:

$$\underbrace{\text{Cisco-IOS-XR-wdsysmon-fd-oper}}_{\text{origin}}:\underbrace{\text{system-monitoring}}_{\text{elem\#1}}/\underbrace{\text{cpu-utilization}}_{\text{elem\#2}}/\underbrace{\text{total-cpu-one-minute}}_{\text{elem\#3}}$$

## Step 1

On the Crosswork GUI, select the two collection jobs created in previous tasks, then click on the **Delete**


Crosswork Network Controller

Administration / Collection Jobs

Collection Jobs 1 / 11

| Status | App ID | Context... |
|---|---|---|
| Successful | capp-hi | SNSR_la... |
| Successful | cw.dlmin... | dlm/snm... |
| Successful | cw.dlmin... | dlm/cli-c... |
| Successful | cw.dlmin... | dlm/cli-c... |
| Successful | cw.topo-... | topo-vis... |
| Successful | cw.topo_... | cw.topos... |
| Successful | cw.dlmin... | dlm/cli-c... |
| Successful | cw.optim... | cw.optim... |
| Successful | applicatio1 | Node-8 |
| Degraded | cw.topo_... | cw.topos... |
| Degraded | cw.optim... | cw.optim... |

Job Details - capp-hi : SNSR_lab_test_MDT_CW_test020b75ac_014ffc52ae16

Status: Successful | Job Configuration: Config Details | Collection Type: MDT | Last Modified On: Mon May 17 2021 14:16:15 EDT

Showing - All Collections (1) | Collection Issues (0)

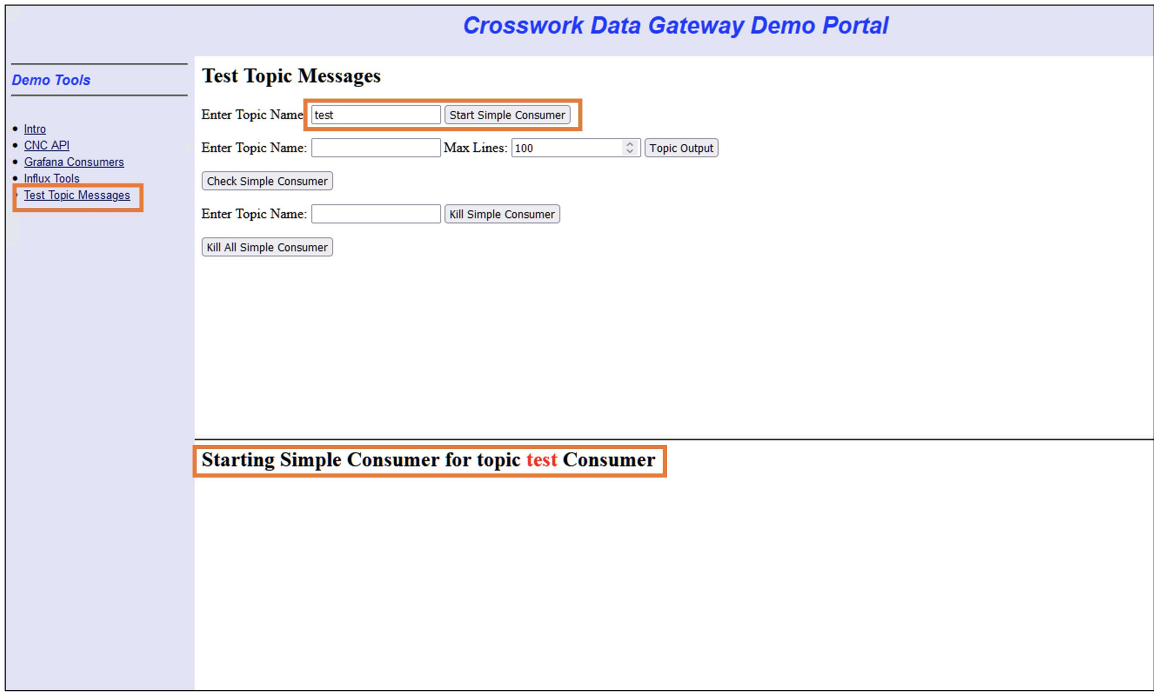
| Status | Hostname | Sensor Data | Last Reported Time |
|---|---|---|---|

The Health Insight collection job (created in **Task 5**) will have **capp-hi** as App ID

Use the refresh button to verify collection jobs have been deleted

## Step 2

Navigate back to the CDG Demo Portal then click **Test Topic Messages**, enter **test** in the topic name field then click on **Start Simple Consumer**. Verify that you see a status message of "Starting Simple Consumer for topic test Consumer" in the bottom pane.



Test Topic Messages

Enter Topic Name: test

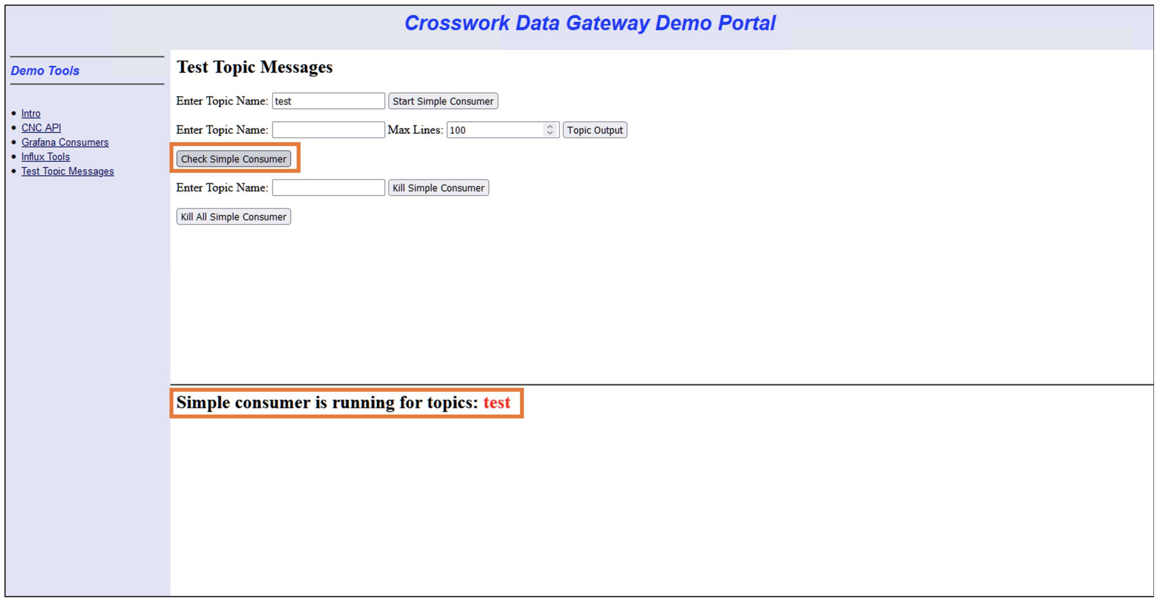
Max Lines: 100

Starting Simple Consumer for topic test Consumer

As opposed to previous used consumers, this one doesn't perform any data parsing nor stores any data in the InfluxDB. It's simply a Kafka consumer script whom target is just to write on a log file Crosswork Data Gateway messages from topic **test**

## Step 3

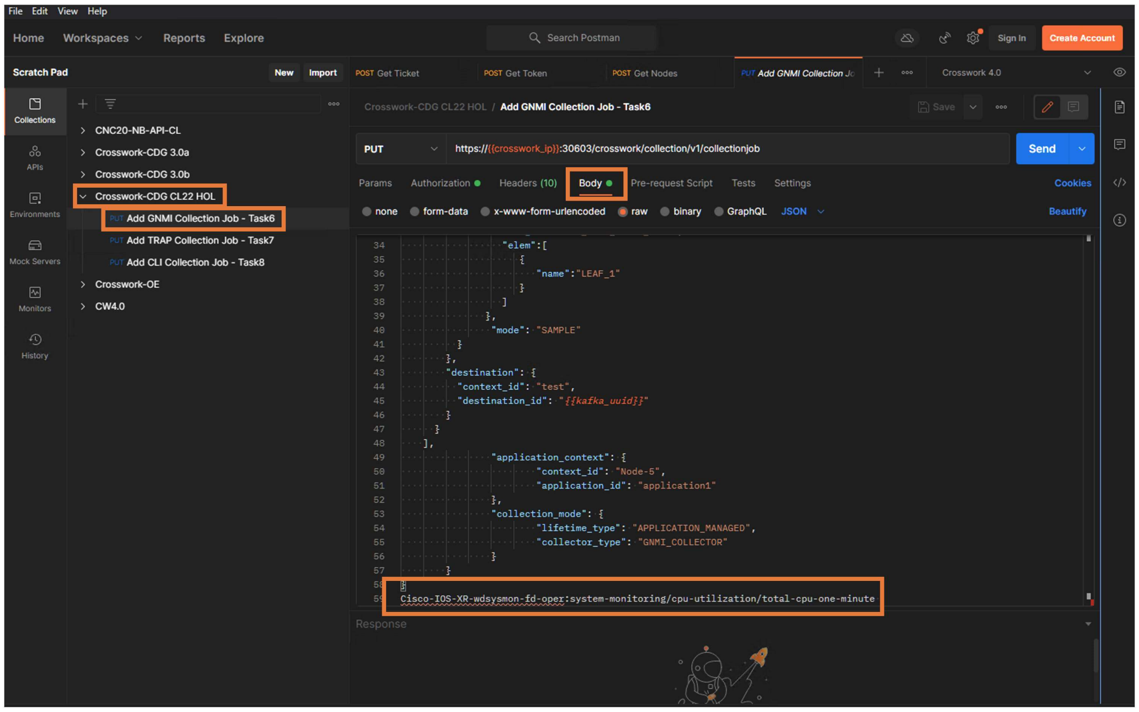
On the CDG Portal, click on **Check Simple Consumer**. Verify Simple consumer for topic test is running as shown below.



Simple consumer is running for topics: test

## Step 4

Navigate back to Postman client, expand the **Crosswork-CDG CL22 HOL** folder, then select **Add gNMI Collection Job - Task6** then select **Body.** Scroll down till the end. We left the whole sensor path string where you can copy/paste from, to facilitate this exercise. **Remember** to remove that line before clicking Send



```
PUT  https://{{crosswork_ip}}:30603/crosswork/collection/v1/collectionjob
```

```
                    "elem":[
                        {
                            "name":"LEAF_1"
                        }
                    ]
                },
                "mode": "SAMPLE"
            }
        },
        "destination": {
            "context_id": "test",
            "destination_id": "{{kafka_uuid}}"
        }
    }
  ],
            "application_context": {
                "context_id": "Node-5",
                "application_id": "application1"
            },
            "collection_mode": {
                "lifetime_type": "APPLICATION_MANAGED",
                "collector_type": "GNMI_COLLECTOR"
            }
        }
}
Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization/total-cpu-one-minute
```

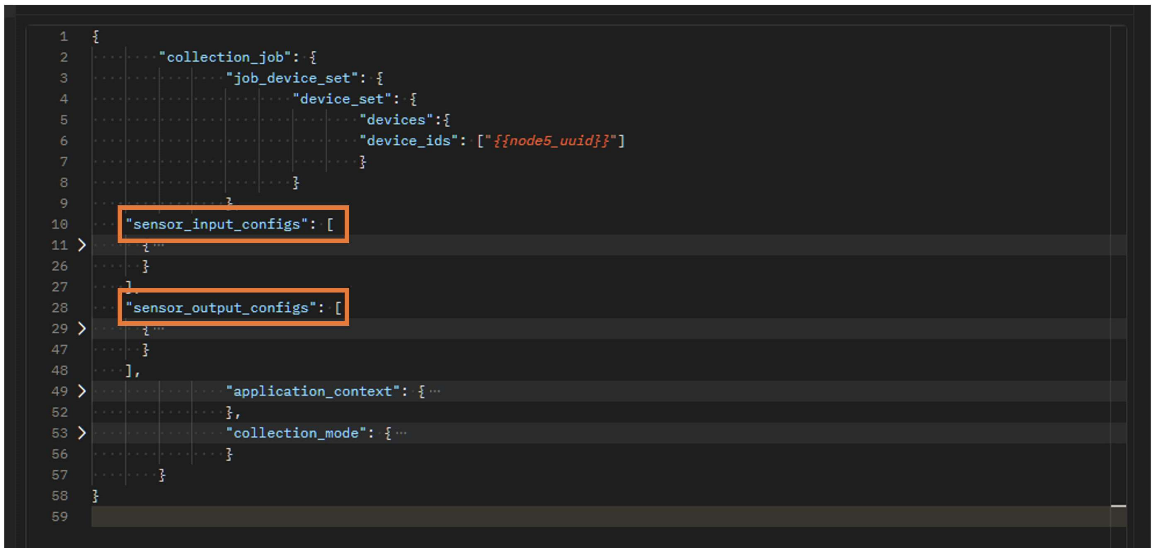
## Step 5

Update the gnmi_sensor section in the API payload, as follows:

```
"gnmi_sensor": {
     "path":{
        "origin":"YANG_DATA_MODEL_NAME",
        "elem":[
           {
              "name":"LEAF_1"
           }
        ]
     },
```

```
"gnmi_sensor": {
     "path":{
        "origin":"Cisco-IOS-XR-wdsysmon-fd-oper",
        "elem":[
          {
              "name":"system-monitoring"
          },
          {
              "name":"cpu-utilization"
          },
          {
              "name":"total-cpu-one-minute"
          }
        ]
     },
```

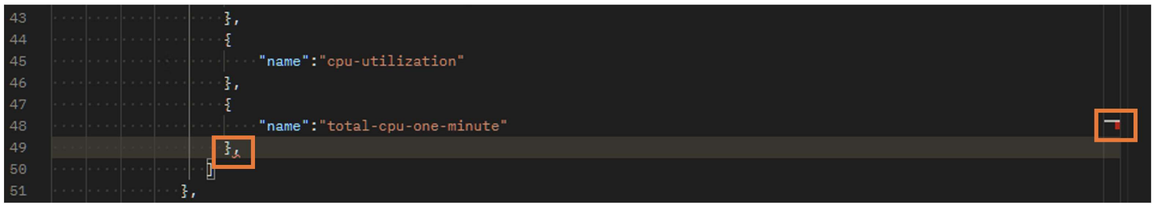
This needs to be done on “both” **sensor_input_configs** and **sensor_output_configs** sections (collapsed in following picture to fit the Postman pane)



```
{
        "collection_job": {
                "job_device_set": {
                        "device_set": {
                                "devices":{
                                "device_ids": ["{{node5_uuid}}"]
                                }
                        }
                },
    "sensor_input_configs": [
        {⋯
        }
    ],
    "sensor_output_configs": [
        {⋯
        }
    ],
                "application_context": {⋯
                },
                "collection_mode": {⋯
                }
        }
}
```

Name/values pairs in the tree are separated by a comma (make sure last one doesn't have it)

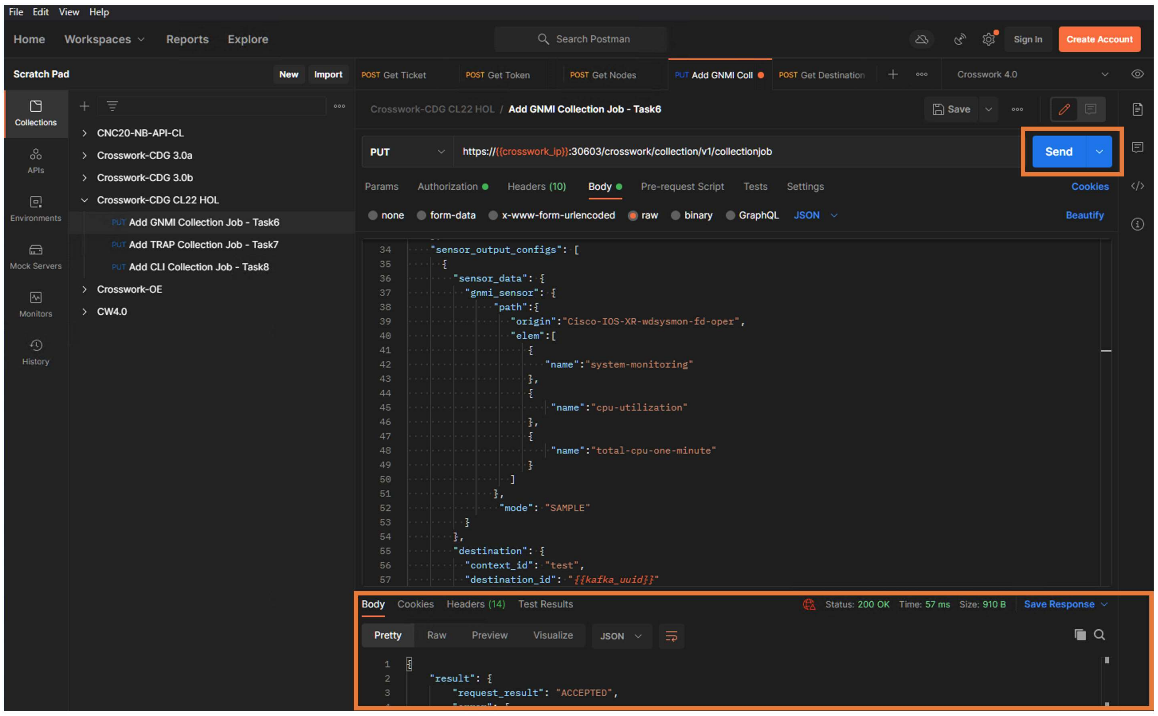
JSON format errors will be highlighted in red, both online and on the right bar



```
                    },
                    {
                        "name":"cpu-utilization"
                    },
                    {
                        "name":"total-cpu-one-minute"
                    },
                  ]
                },
```

## Step 6

Once all looks good and no errors are highlighted, click on the Send. Verify that **requested_result**: **ACCEPTED** is returned in the result section as highlighted below.



PUT https://{{crosswork_ip}}:30603/crosswork/collection/v1/collectionjob

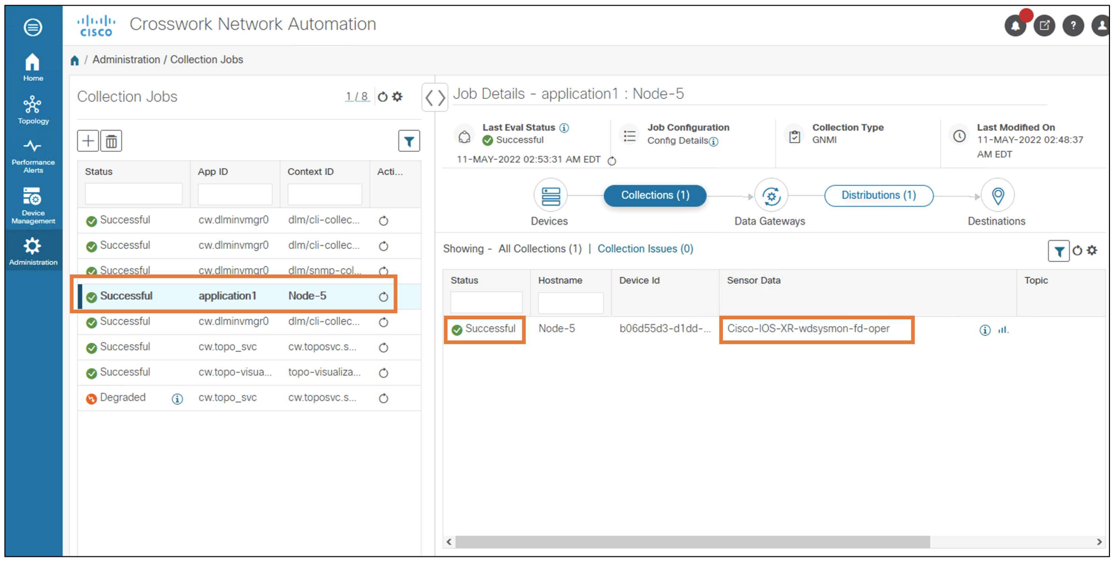
```
    "sensor_output_configs": [
      {
        "sensor_data": {
          "gnmi_sensor": {
               "path":{
                  "origin":"Cisco-IOS-XR-wdsysmon-fd-oper",
                  "elem":[
                     {
                        "name":"system-monitoring"
                     },
                     {
                        "name":"cpu-utilization"
                     },
                     {
                        "name":"total-cpu-one-minute"
                     }
                  ]
               },
                "mode": "SAMPLE"
          }
        },
        "destination": {
          "context_id": "test",
          "destination_id": "{{kafka_uuid}}"
```

Status: 200 OK Time: 57 ms Size: 910 B

```
{
    "result": {
        "request_result": "ACCEPTED",
```

## Step 7

In Crosswork, navigate to **Administration -> Collection Jobs**. Verify the collection job gets created and is reported in the collection job list with **Successful** State



| Status | App ID | Context ID | Acti... |
|---|---|---|---|
| Successful | cw.dlminvmgr0 | dlm/cli-collec... | |
| Successful | cw.dlminvmgr0 | dlm/cli-collec... | |
| Successful | cw.dlminvmgr0 | dlm/snmp-col... | |
| Successful | application1 | Node-5 | |
| Successful | cw.dlminvmgr0 | dlm/cli-collec... | |
| Successful | cw.topo_svc | cw.toposvc.s... | |
| Successful | cw.topo-visua... | topo-visualiza... | |
| Degraded | cw.topo_svc | cw.toposvc.s... | |

Job Details - application1 : Node-5

Last Eval Status: Successful
11-MAY-2022 02:53:31 AM EDT

Job Configuration: Config Details

Collection Type: GNMI

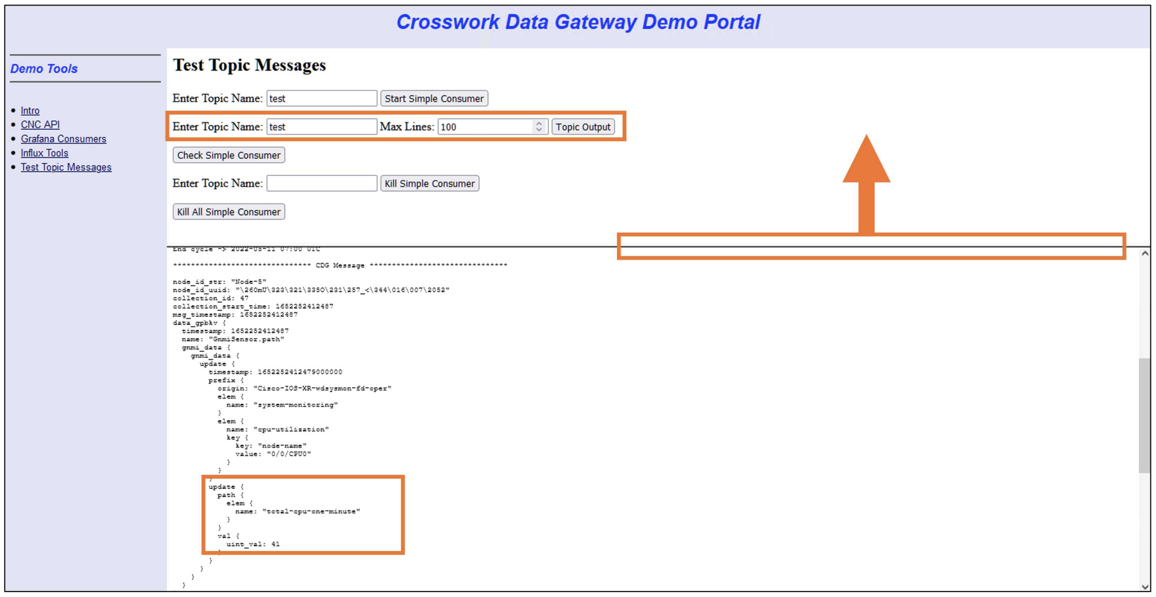
Last Modified On: 11-MAY-2022 02:48:37 AM EDT

Showing - All Collections (1) | Collection Issues (0)

| Status | Hostname | Device Id | Sensor Data | Topic |
|---|---|---|---|---|
| Successful | Node-5 | b06d55d3-d1dd-... | Cisco-IOS-XR-wdsysmon-fd-oper | |

## Step 8

Go back on the CDG Portal, enter **test** in the topic name field on the second textfield as shown in the picture, leave the default 100 lines, then click on **Topic Output**. Slide the centre bar up to enlarge the bottom pane. Verify the one minute CPU counter is present in the gNMI update section as shown below.



```
End cycle -> 2022-05-11 07:00 UTC

****************************** CDG Message ******************************

node_id_str: "Node-5"
node_id_uuid: "\260mU\323\321\3350\231\257_<\344\016\007\2052"
collection_id: 47
collection_start_time: 1652252412487
msg_timestamp: 1652252412487
data_gpbkv {
  timestamp: 1652252412487
  name: "GnmiSensor.path"
  gnmi_data {
    gnmi_data {
      update {
        timestamp: 1652252412479000000
        prefix {
          origin: "Cisco-IOS-XR-wdsysmon-fd-oper"
          elem {
            name: "system-monitoring"
          }
          elem {
            name: "cpu-utilization"
            key {
              key: "node-name"
              value: "0/0/CPU0"
            }
          }
        }
        update {
          path {
            elem {
              name: "total-cpu-one-minute"
            }
          }
          val {
            uint_val: 41
          }
        }
      }
    }
  }
```

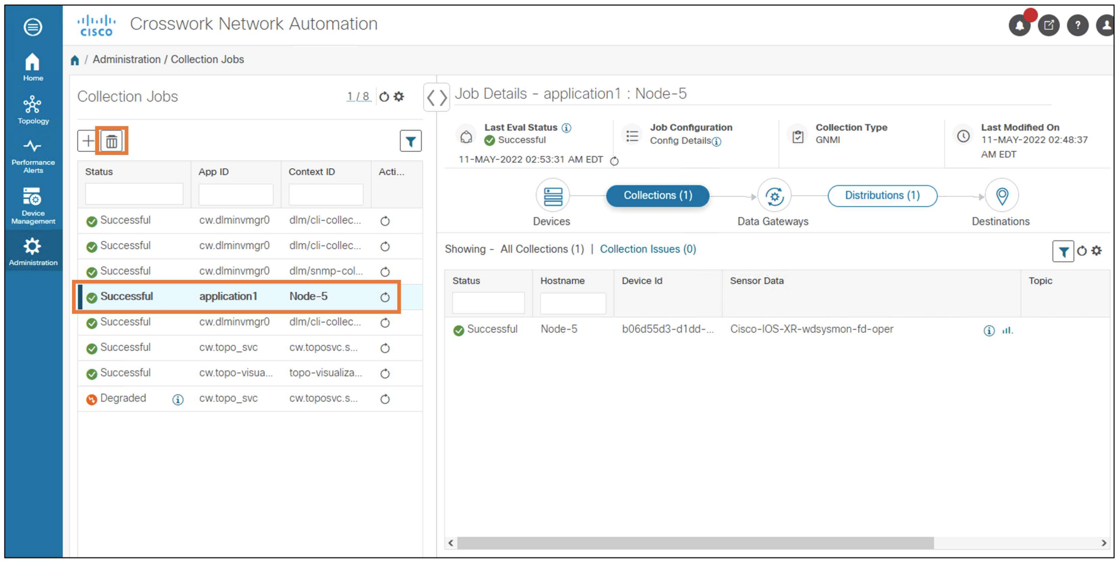
**This concludes task 5.**

## Task 6: Create a Collection job for TRAP

Although not strictly related to telemetry collection, in this task we will use Crosswork Data Gateway to collect device TRAP and send them to the External Kafka server. Crosswork Data Gateway capabilities extend the telemetry area and can also be used as common collector for collecting and forwarding device events (SYSLOG or TRAP).

## Step 1

On Crosswork GUI, select the collection Job created in previous task, then click **Delete**



Job Details - application1 : Node-5

Last Eval Status: Successful, 11-MAY-2022 02:53:31 AM EDT
Job Configuration: Config Details
Collection Type: GNMI
Last Modified On: 11-MAY-2022 02:48:37 AM EDT

| Status | App ID | Context ID | Acti... |
|---|---|---|---|
| Successful | cw.dlminvmgr0 | dlm/cli-collec... | |
| Successful | cw.dlminvmgr0 | dlm/cli-collec... | |
| Successful | cw.dlminvmgr0 | dlm/snmp-col... | |
| Successful | application1 | Node-5 | |
| Successful | cw.dlminvmgr0 | dlm/cli-collec... | |
| Successful | cw.topo_svc | cw.toposvc.s... | |
| Successful | cw.topo-visua... | topo-visualiza... | |
| Degraded | cw.topo_svc | cw.toposvc.s... | |

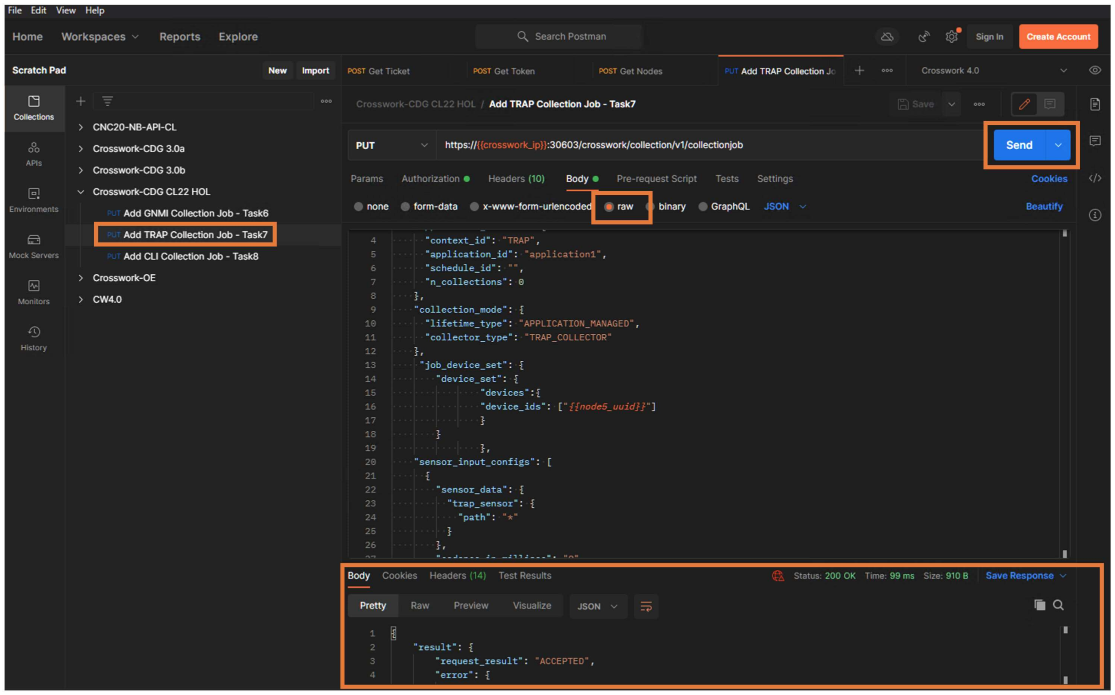
Showing - All Collections (1) | Collection Issues (0)

| Status | Hostname | Device Id | Sensor Data | Topic |
|---|---|---|---|---|
| Successful | Node-5 | b06d55d3-d1dd-... | Cisco-IOS-XR-wdsysmon-fd-oper | |

Use the refresh button to verify collection jobs have been deleted. In this task we will reusing topic **test**, so we need to make sure previous job stops sending messages to same.

## Step 2

On Postman client select **Add TRAP Collection Job – Task7**, select **Body** and click **Send**. Verify that **requested_result**: **ACCEPTED** is returned in the result section as highlighted below.


File Edit View Help
Home Workspaces Reports Explore
Search Postman
Sign In
Create Account
Scratch Pad
New Import
POST Get Ticket
POST Get Token
POST Get Nodes
PUT Add TRAP Collection Jo
Crosswork 4.0
Collections
APIs
Environments
Mock Servers
Monitors
History
CNC20-NB-API-CL
Crosswork-CDG 3.0a
Crosswork-CDG 3.0b
Crosswork-CDG CL22 HOL
PUT Add GNMI Collection Job - Task6
PUT Add TRAP Collection Job - Task7
PUT Add CLI Collection Job - Task8
Crosswork-OE
CW4.0
Crosswork-CDG CL22 HOL / Add TRAP Collection Job - Task7
Save
Send
Params Authorization Headers (10) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
Body Cookies Headers (14) Test Results
Save Response
Pretty Raw Preview Visualize JSON

PUT https://{{crosswork_ip}}:30603/crosswork/collection/v1/collectionjob

```
    "context_id": "TRAP",
    "application_id": "application1",
    "schedule_id": "",
    "n_collections": 0
  },
  "collection_mode": {
    "lifetime_type": "APPLICATION_MANAGED",
    "collector_type": "TRAP_COLLECTOR"
  },
   "job_device_set": {
     "device_set": {
               "devices":{
               "device_ids": ["{{node5_uuid}}"]
               }
     }
               },
  "sensor_input_configs": [
    {
      "sensor_data": {
        "trap_sensor": {
          "path": "*"
        }
      },
```

Status: 200 OK Time: 99 ms Size: 910 B

```
{
    "result": {
        "request_result": "ACCEPTED",
        "error": {
```

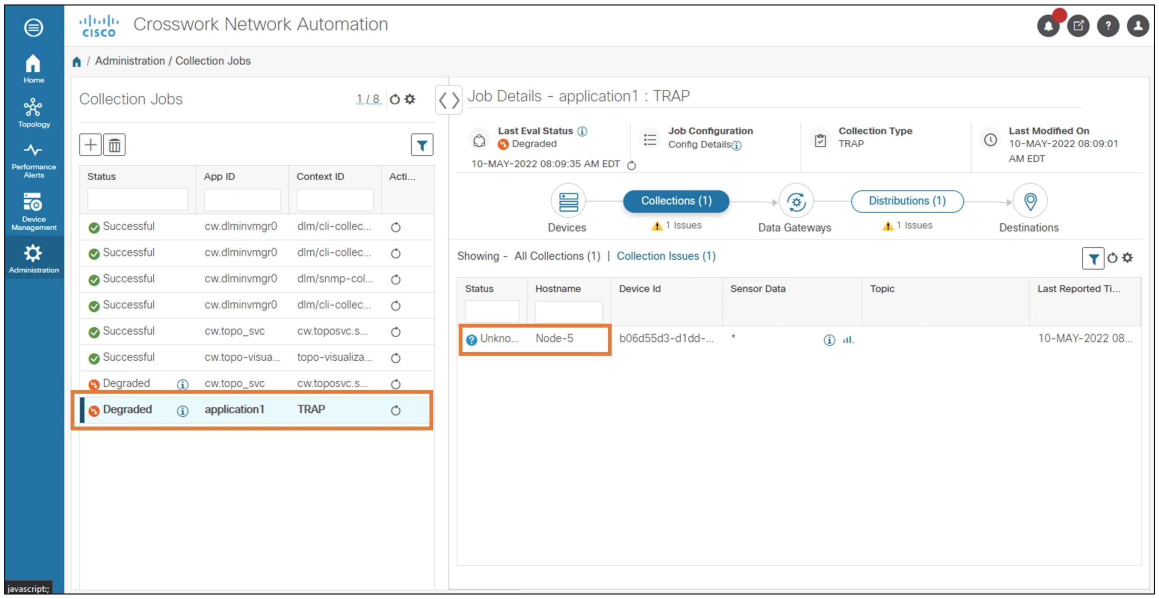
You may notice the new **sensor_data** in the API call payload

```
{
  "sensor_data": {
    "trap_sensor": {
      "path": "*"
    }
  },
  "cadence_in_millisec": "0"
}
```

Where the **trap_sensor** has “*” and cadence is set to **0**. Crosswork Data Gateway has filter options for TRAP like a given TRAP OID or a complete MIB. In this case, setting “*” means any TRAP and no filter at all. Cadence is set to **0** as there is no timer for TRAP. Once a device sends a TRAP, Crosswork Data Gateway will process it.

## Step 3

In Crosswork, navigate to **Administration -> Collection Jobs**. Verify the TRAP collection job gets created and is reported in the collection job list



Collection Jobs 1 / 8

| Status | App ID | Context ID | Acti... |
|---|---|---|---|
| Successful | cw.dlminvmgr0 | dlm/cli-collec... | |
| Successful | cw.dlminvmgr0 | dlm/cli-collec... | |
| Successful | cw.dlminvmgr0 | dlm/snmp-col... | |
| Successful | cw.dlminvmgr0 | dlm/cli-collec... | |
| Successful | cw.topo_svc | cw.toposvc.s... | |
| Successful | cw.topo-visua... | topo-visualiza... | |
| Degraded | cw.topo_svc | cw.toposvc.s... | |
| Degraded | application1 | TRAP | |

Job Details - application1 : TRAP

Last Eval Status: Degraded
10-MAY-2022 08:09:35 AM EDT
Job Configuration: Config Details
Collection Type: TRAP
Last Modified On: 10-MAY-2022 08:09:01 AM EDT

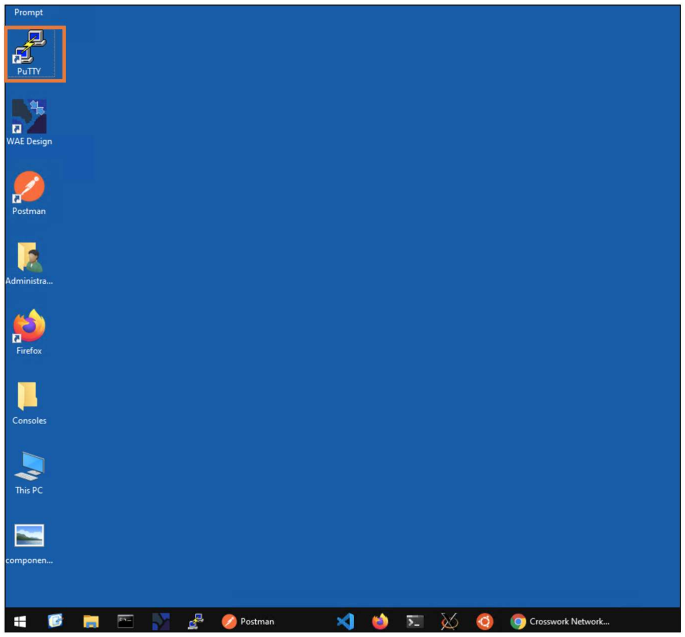
Showing - All Collections (1) | Collection Issues (1)

| Status | Hostname | Device Id | Sensor Data | Topic | Last Reported Ti... |
|---|---|---|---|---|---|
| Unkno... | Node-5 | b06d55d3-d1dd-... | * | | 10-MAY-2022 08... |

Collection Job will stay in **Degraded** Status and Job Details will report collection in **Unknown** state for Node-5. This is expected as no TRAP arrived yet. We need to generate a TRAP on Node-5 to see collection job changing its status.

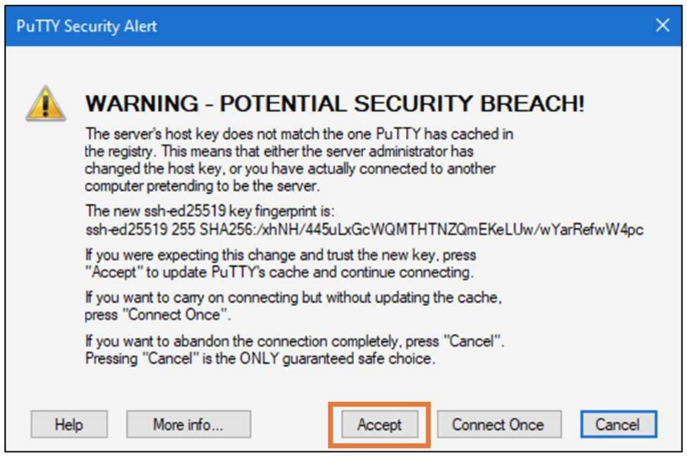
## Step 4

On your workstation desktop, launch the **PuTTY** shortcut, and login using **IP 198.19.1.5**, username **cisco**, password **cisco**



## Step 5

Click on **Accept** if warned



WARNING - POTENTIAL SECURITY BREACH!

The server's host key does not match the one PuTTY has cached in the registry. This means that either the server administrator has changed the host key, or you have actually connected to another computer pretending to be the server.

The new ssh-ed25519 key fingerprint is:
ssh-ed25519 255 SHA256:/xhNH/445uLxGcWQMTHTNZQmEKeLUw/wYarRefwW4pc

If you were expecting this change and trust the new key, press "Accept" to update PuTTY's cache and continue connecting.

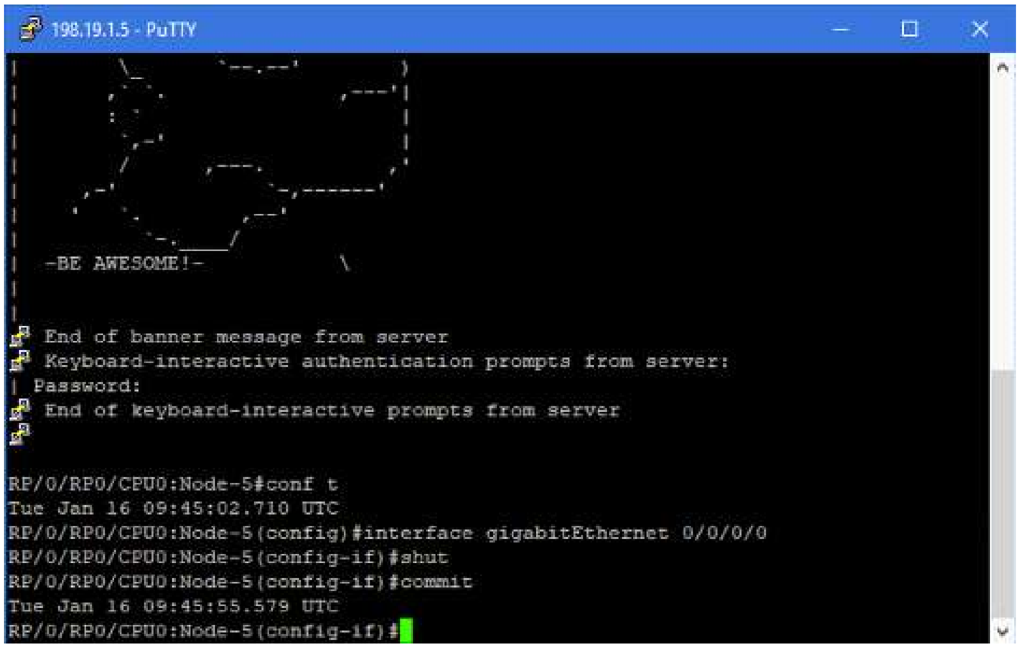
If you want to carry on connecting but without updating the cache, press "Connect Once".

If you want to abandon the connection completely, press "Cancel". Pressing "Cancel" is the ONLY guaranteed safe choice.

## Step 6

Issue the command sequence as follow:

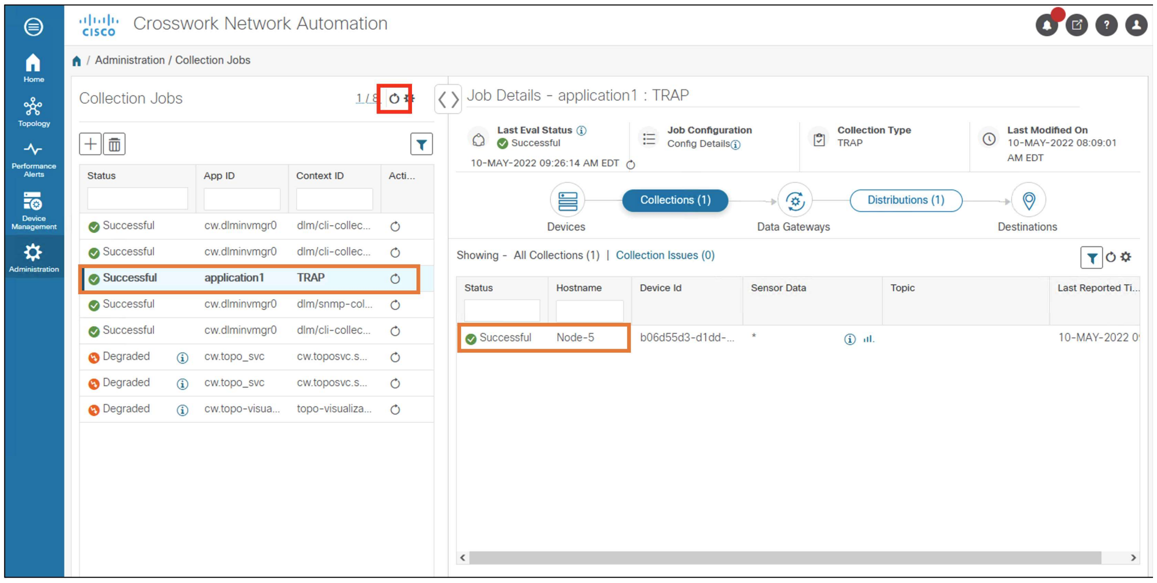
```
# conf t
(config)# interface gigabitEthernet 0/0/0/0
(config-if)# shut
(config-if)# commit
```



```
|        \_      `--.--'        )
|       ,`  .              ,---'|
|       : .                     |
|        `,-'                   |
|        /      ,---.          ,'
|     ,-'           `-,------'
|    '   `.       ,--'
|          `-.____/
|  -BE AWESOME!-           \
|
|
  End of banner message from server
  Keyboard-interactive authentication prompts from server:
| Password:
  End of keyboard-interactive prompts from server


RP/0/RP0/CPU0:Node-5#conf t
Tue Jan 16 09:45:02.710 UTC
RP/0/RP0/CPU0:Node-5(config)#interface gigabitEthernet 0/0/0/0
RP/0/RP0/CPU0:Node-5(config-if)#shut
RP/0/RP0/CPU0:Node-5(config-if)#commit
Tue Jan 16 09:45:55.579 UTC
RP/0/RP0/CPU0:Node-5(config-if)#
```

## Step 7

On Crosswork GUI, click on the refresh button highlighted in red. Verify the **TRAP** collection job now is in status **Successful** and Job details has collection for **Node-5** in **Successful** state as well



| Status | App ID | Context ID | Acti... |
|---|---|---|---|
| Successful | cw.dlminvmgr0 | dlm/cli-collec... | |
| Successful | cw.dlminvmgr0 | dlm/cli-collec... | |
| Successful | application1 | TRAP | |
| Successful | cw.dlminvmgr0 | dlm/snmp-col... | |
| Successful | cw.dlminvmgr0 | dlm/cli-collec... | |
| Degraded | cw.topo_svc | cw.toposvc.s... | |
| Degraded | cw.topo_svc | cw.toposvc.s... | |
| Degraded | cw.topo-visua... | topo-visualiza... | |

Job Details - application1 : TRAP

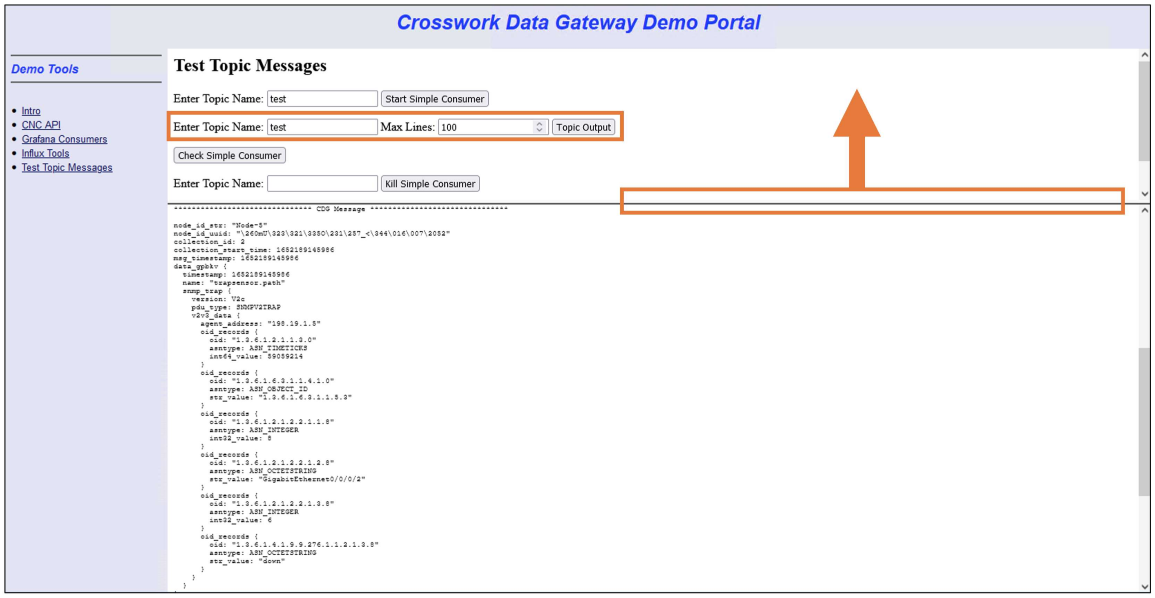
Last Eval Status: Successful
10-MAY-2022 09:26:14 AM EDT
Job Configuration: Config Details
Collection Type: TRAP
Last Modified On: 10-MAY-2022 08:09:01 AM EDT

Showing - All Collections (1) | Collection Issues (0)

| Status | Hostname | Device Id | Sensor Data | Topic | Last Reported Ti... |
|---|---|---|---|---|---|
| Successful | Node-5 | b06d55d3-d1dd-... | * | | 10-MAY-2022 0 |

## Step 8

Go back on the CDG Portal, enter **test** in the topic name field, leave the default 100 lines then click on **Topic Output**. Slide the centre bar up to enlarge the bottom pane. Verify the gigabitEthernet interface down TRAP is visualised as shown here.



```
****************************** CDG Message ******************************

node_id_str: "Node-5"
node_id_uuid: "\260mU\323\321\335O\231\257_<\344\016\007\2052"
collection_id: 2
collection_start_time: 1652189145986
msg_timestamp: 1652189145986
data_gpbkv {
  timestamp: 1652189145986
  name: "trapsensor.path"
  snmp_trap {
    version: V2c
    pdu_type: SNMPV2TRAP
    v2v3_data {
      agent_address: "198.19.1.5"
      oid_records {
        oid: "1.3.6.1.2.1.1.3.0"
        asntype: ASN_TIMETICKS
        int64_value: 59059214
      }
      oid_records {
        oid: "1.3.6.1.6.3.1.1.4.1.0"
        asntype: ASN_OBJECT_ID
        str_value: "1.3.6.1.6.3.1.1.5.3"
      }
      oid_records {
        oid: "1.3.6.1.2.1.2.2.1.1.8"
        asntype: ASN_INTEGER
        int32_value: 8
      }
      oid_records {
        oid: "1.3.6.1.2.1.2.2.1.2.8"
        asntype: ASN_OCTETSTRING
        str_value: "GigabitEthernet0/0/0/2"
      }
      oid_records {
        oid: "1.3.6.1.2.1.2.2.1.3.8"
        asntype: ASN_INTEGER
        int32_value: 6
      }
      oid_records {
        oid: "1.3.6.1.4.1.9.9.276.1.1.2.1.3.8"
        asntype: ASN_OCTETSTRING
        str_value: "down"
      }
    }
  }
```

**This concludes task 6.**

## Task 7: Create a Collection job for CLI

With task #7, you have been completing the most valuable tasks from this lab. If time permits and you are eager to learn more, feel free to move on with Task #8 and Task #9

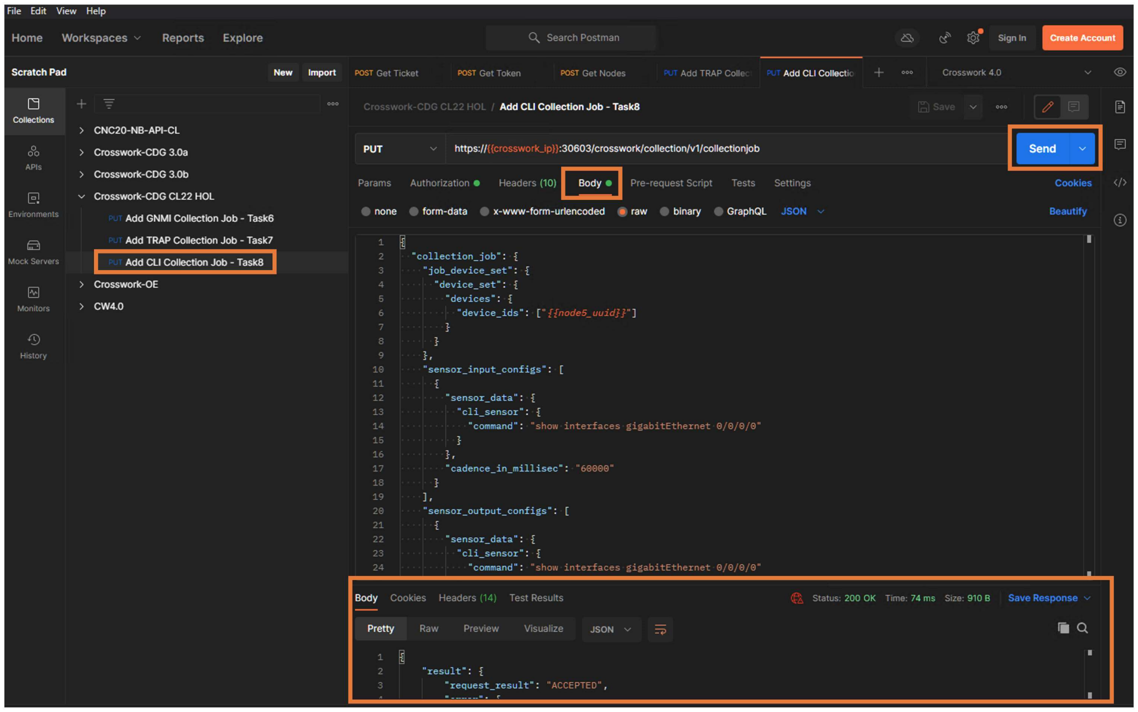
There are multiple ways of collecting telemetry data. With previous tasks, we have seen how to accomplish with this requirement leveraging widely used protocols such as SNMP, MDT (Cisco only) and gNMI. There might be cases where a given counter is not available with those protocols or the amount of traffic generated by the device to return a single KPI would be excessive if compared with the use case to be implemented. Or simply customer is interested on another use cases like retrieving device inventory data by CLI.

Crosswork Data Gateway collection jobs can also be used to run “show” CLI commands. API payload already include a predefined show command. Feel free to change it as you like as it won’t impact task execution (just remember to update both **input** and **output sensor sections**).

Unfortunately, consumer log won’t be completely visible on the Demo Portal as special characters in Data Gateway message prevents displaying the message payload. We will connect directly to the VM where consumers are running and check logs manually

## Step 1

On Postman client select **Add CLI Collection Job** , select **Body**, eventually change show command then click on **Send**. Verify that **requested_result**: **ACCEPTED** is returned in the result section as highlighted below.


File Edit View Help
Home Workspaces Reports Explore
Search Postman
Sign In Create Account
Scratch Pad New Import
POST Get Ticket
POST Get Token
POST Get Nodes
PUT Add TRAP Collec
PUT Add CLI Collectio
Crosswork 4.0
Collections
APIs
Environments
Mock Servers
Monitors
History
CNC20-NB-API-CL
Crosswork-CDG 3.0a
Crosswork-CDG 3.0b
Crosswork-CDG CL22 HOL
PUT Add GNMI Collection Job - Task6
PUT Add TRAP Collection Job - Task7
PUT Add CLI Collection Job - Task8
Crosswork-OE
CW4.0
Crosswork-CDG CL22 HOL / Add CLI Collection Job - Task8
Save
Send
Params Authorization Headers (10) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
Body Cookies Headers (14) Test Results
Status: 200 OK Time: 74 ms Size: 910 B
Save Response
Pretty Raw Preview Visualize JSON

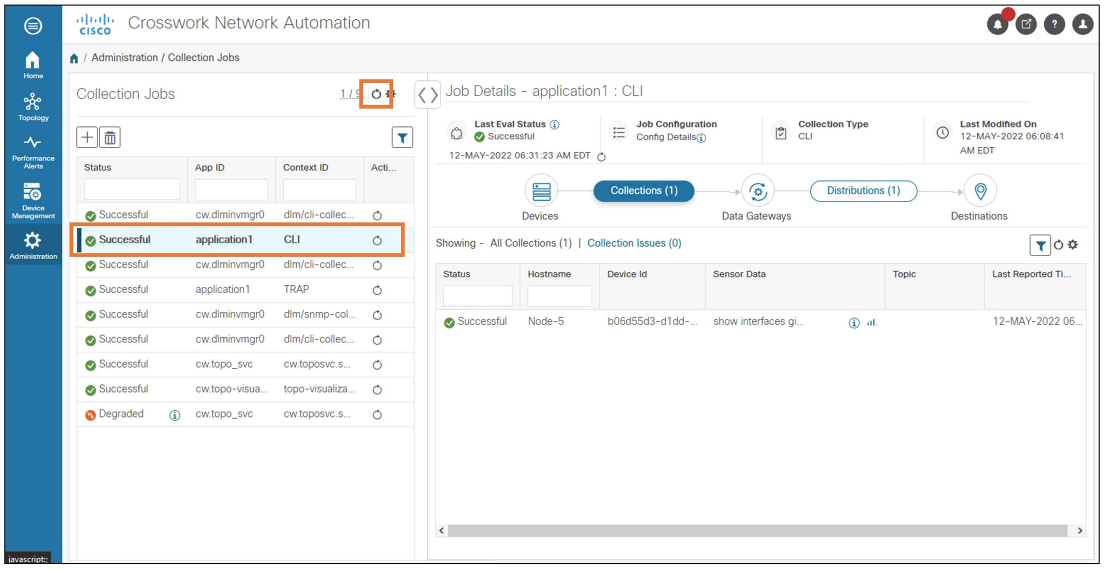
```
PUT https://{{crosswork_ip}}:30603/crosswork/collection/v1/collectionjob
```

```
{
  "collection_job": {
    "job_device_set": {
      "device_set": {
        "devices": {
          "device_ids": ["{{node5_uuid}}"]
        }
      }
    },
    "sensor_input_configs": [
      {
        "sensor_data": {
          "cli_sensor": {
            "command": "show interfaces gigabitEthernet 0/0/0/0"
          }
        },
        "cadence_in_millisec": "60000"
      }
    ],
    "sensor_output_configs": [
      {
        "sensor_data": {
          "cli_sensor": {
            "command": "show interfaces gigabitEthernet 0/0/0/0"
```

```
{
    "result": {
        "request_result": "ACCEPTED",
```

## Step 2

On Crosswork GUI, click on the refresh button highlighted. Verify the **CLI** collection job now is in status **Successful** and Job details has collection for **Node-5** in **Successful** state as well



Administration / Collection Jobs

Collection Jobs 1 / 9

| Status | App ID | Context ID | Acti... |
|---|---|---|---|
| Successful | cw.dlminvmgr0 | dlm/cli-collec... | |
| Successful | application1 | CLI | |
| Successful | cw.dlminvmgr0 | dlm/cli-collec... | |
| Successful | application1 | TRAP | |
| Successful | cw.dlminvmgr0 | dlm/snmp-col... | |
| Successful | cw.dlminvmgr0 | dlm/cli-collec... | |
| Successful | cw.topo_svc | cw.toposvc.s... | |
| Successful | cw.topo-visua... | topo-visualiza... | |
| Degraded | cw.topo_svc | cw.toposvc.s... | |

Job Details - application1 : CLI

Last Eval Status: Successful
12-MAY-2022 06:31:23 AM EDT

Job Configuration: Config Details

Collection Type: CLI

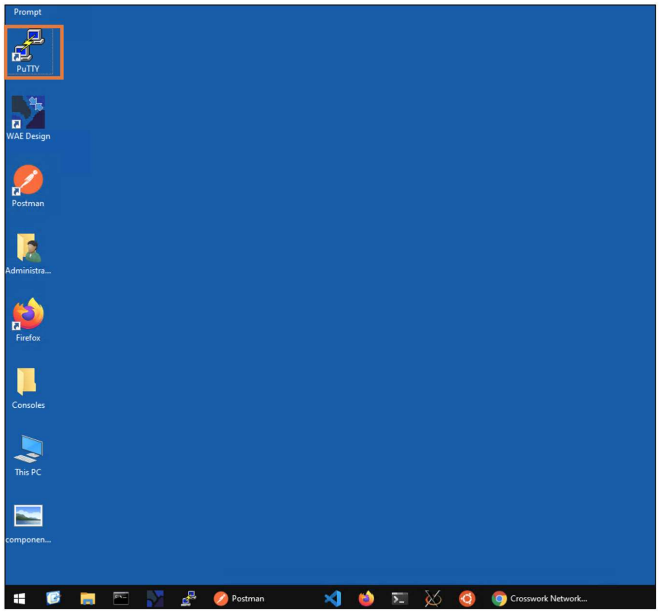
Last Modified On: 12-MAY-2022 06:08:41 AM EDT

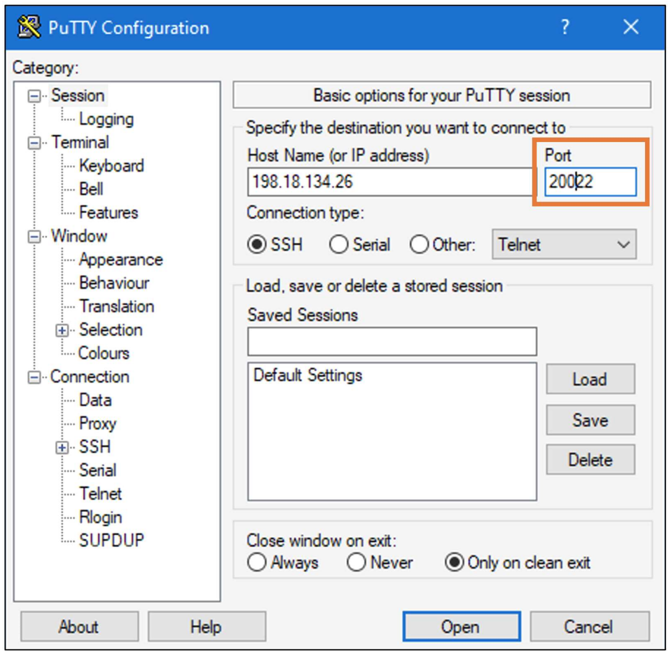
Showing - All Collections (1) | Collection Issues (0)

| Status | Hostname | Device Id | Sensor Data | Topic | Last Reported Ti... |
|---|---|---|---|---|---|
| Successful | Node-5 | b06d55d3-d1dd-... | show interfaces gi... | | 12-MAY-2022 06... |

## Step 3

On your workstation desktop, launch the **PuTTY** shortcut, and login using **IP 198.18.134.26**, and port **20022** (not the default 22).





Specify the destination you want to connect to

Host Name (or IP address): 198.18.134.26

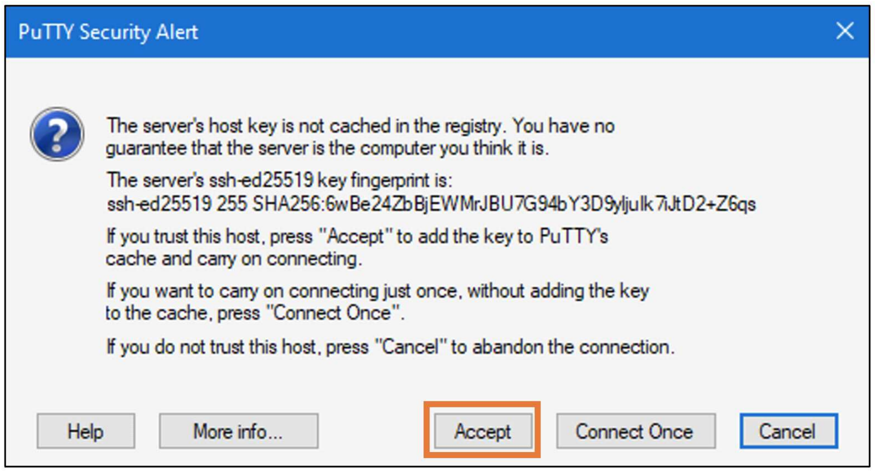
Port: 20022

Connection type: [✓ SSH] Serial Other: Telnet

Close window on exit: Always Never [✓ Only on clean exit]

## Step 4

Click on **Accept** if warned



The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.

The server's ssh-ed25519 key fingerprint is:
ssh-ed25519 255 SHA256:6wBe24ZbBjEWMrJBU7G94bY3D9yljulk7iJtD2+Z6qs

If you trust this host, press "Accept" to add the key to PuTTY's cache and carry on connecting.

If you want to carry on connecting just once, without adding the key to the cache, press "Connect Once".

If you do not trust this host, press "Cancel" to abandon the connection.

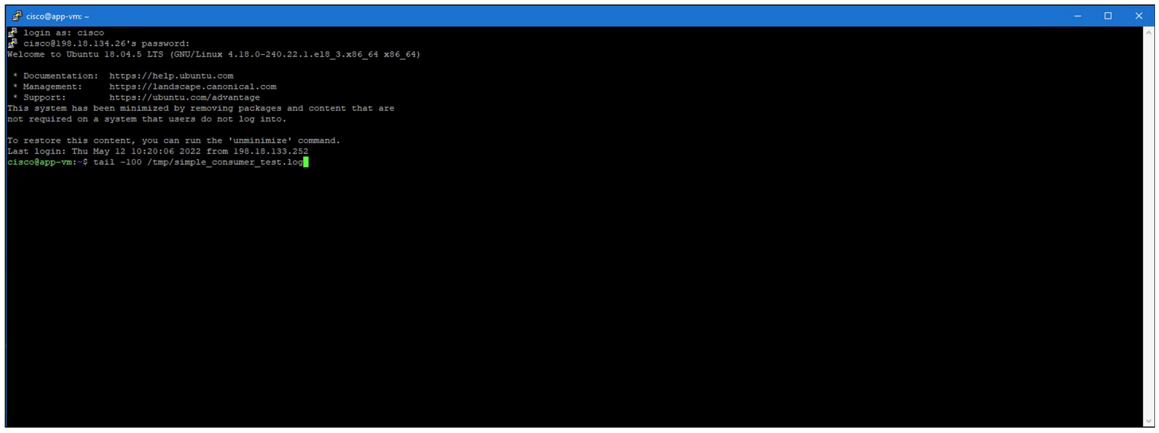
Use username **cisco**, password **cisco** to login

## Step 5

Enlarge the Putty window then issue following command:

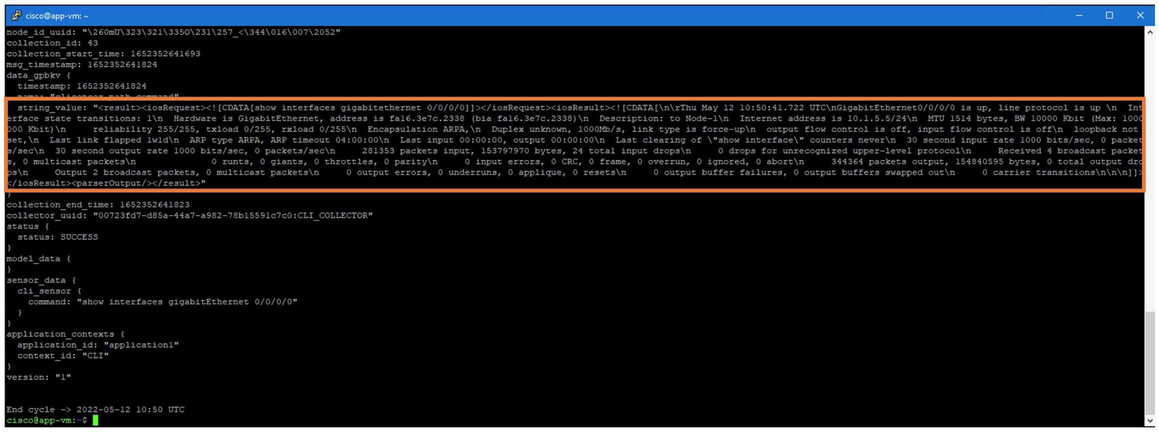
**tail -100 /tmp/simple_consumer_test.log**

**TIP**: press tab after **tail -f /tmp/si** to autocomplete log filename



```
login as: cisco
cisco@198.18.134.26's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.18.0-240.22.1.el8_3.x86_64 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Thu May 12 10:20:06 2022 from 198.18.133.252
cisco@app-vm:~$ tail -100 /tmp/simple_consumer_test.log
```

Device session, with both executed command and the device reply, will be show in the “string_value” parameter (we will analyse further in next task)



```
node_id_uuid: "\260mU\323\321\3350\231\257_<\344\016\007\2052"
collection_id: 43
collection_start_time: 1652352641693
msg_timestamp: 1652352641824
data_gpbkv {
  timestamp: 1652352641824
  name: "clisensor_path_command"
  string_value: "<result><iosRequest><![CDATA[show interfaces gigabitethernet 0/0/0/0]]></iosRequest><iosResult><![CDATA[\n\rThu May 12 10:50:41.722 UTC\nGigabitEthernet0/0/0/0 is up, line protocol is up \n  Int
erface state transitions: 1\n  Hardware is GigabitEthernet, address is fa16.3e7c.2338 (bia fa16.3e7c.2338)\n  Description: to Node-1\n  Internet address is 10.1.5.5/24\n  MTU 1514 bytes, BW 10000 Kbit (Max: 1000
000 Kbit)\n     reliability 255/255, txload 0/255, rxload 0/255\n  Encapsulation ARPA,\n  Duplex unknown, 1000Mb/s, link type is force-up\n  output flow control is off, input flow control is off\n  loopback not 
set,\n  Last link flapped 1w1d\n  ARP type ARPA, ARP timeout 04:00:00\n  Last input 00:00:00, output 00:00:00\n  Last clearing of \"show interface\" counters never\n  30 second input rate 1000 bits/sec, 0 packet
s/sec\n  30 second output rate 1000 bits/sec, 0 packets/sec\n     281353 packets input, 153797970 bytes, 24 total input drops\n     0 drops for unrecognized upper-level protocol\n     Received 4 broadcast packet
s, 0 multicast packets\n              0 runts, 0 giants, 0 throttles, 0 parity\n     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort\n     344364 packets output, 154840595 bytes, 0 total output dro
ps\n     Output 2 broadcast packets, 0 multicast packets\n     0 output errors, 0 underruns, 0 applique, 0 resets\n     0 output buffer failures, 0 output buffers swapped out\n     0 carrier transitions\n\n\n]]>
</iosResult><parserOutput/></result>"
}
collection_end_time: 1652352641823
collector_uuid: "00723fd7-d85a-44a7-a982-78b15591c7c0:CLI_COLLECTOR"
status {
  status: SUCCESS
}
model_data {
}
sensor_data {
  cli_sensor {
    command: "show interfaces gigabitEthernet 0/0/0/0"
  }
}
application_contexts {
  application_id: "application1"
  context_id: "CLI"
}
version: "1"


End cycle -> 2022-05-12 10:50 UTC
cisco@app-vm:~$
```

(**NOTE**: If you check with the CDG Demo Portal, using the test consumer, message payload will be shown empty)

**This concludes task 7.**

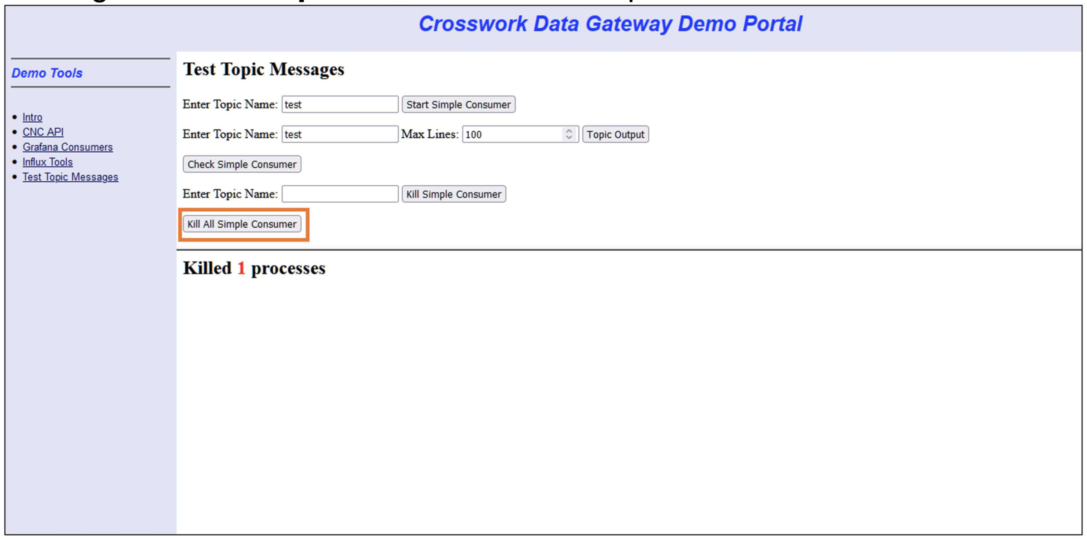
## Task 8: Implementing a simple consumer in Python

In this task we will see how to use Python to read Cisco Data Gateway messages from the Kafka server and extract message payload. We will leverage the protoc message definition shown on **Task #3**.

This task requires basic knowledge of **vi** editor and **Python**

### Step 1

On the CDG Portal click **Kill All Simple Consumers**. Verify that you see a status message of “**Killed 1 process**” in the bottom pane.



Killed 1 processes

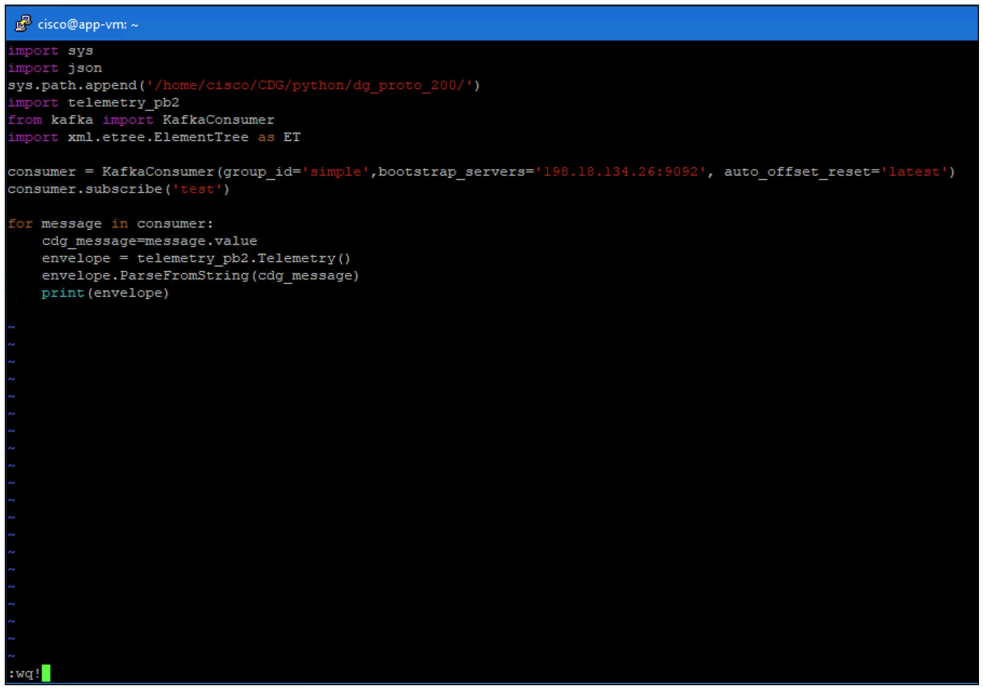
## Step 2

On Putty terminal (you open on previous task), create a new python file using the vi editor (ex: vi test.py). You can create it on the cisco home folder or under /tmp

Copy/paste from below (use mouse right-click to paste on the terminal):

```
import sys
import json
sys.path.append('/home/cisco/CDG/python/dg_proto_200/')
import telemetry_pb2
from kafka import KafkaConsumer
import xml.etree.ElementTree as ET

consumer = KafkaConsumer(group_id='simple',bootstrap_servers='198.18.134.26:9092',
auto_offset_reset='latest')
consumer.subscribe('test')

for message in consumer:
    cdg_message=message.value
    envelope = telemetry_pb2.Telemetry()
    envelope.ParseFromString(cdg_message)
    print(envelope)
```

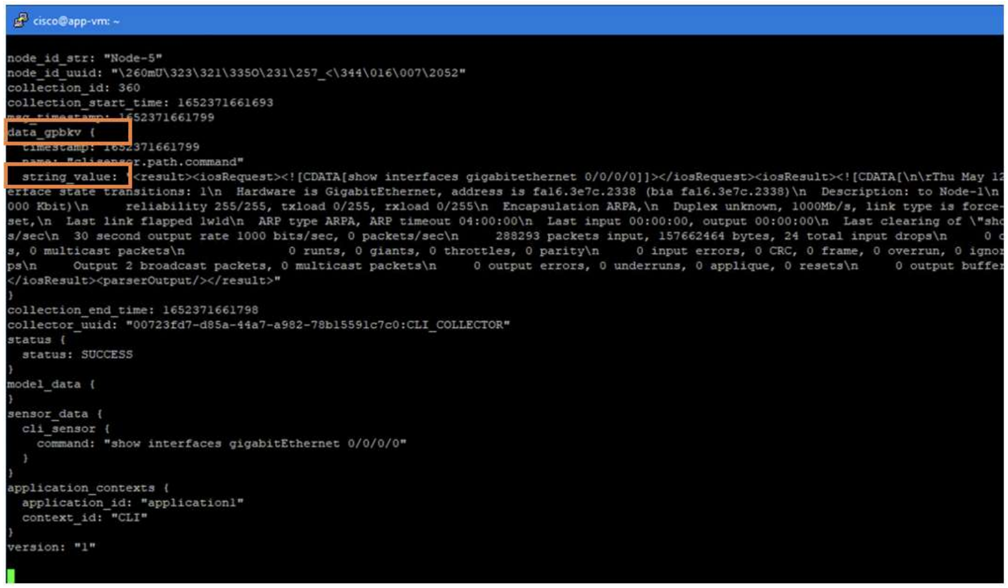
and save



```
import sys
import json
sys.path.append('/home/cisco/CDG/python/dg_proto_200/')
import telemetry_pb2
from kafka import KafkaConsumer
import xml.etree.ElementTree as ET

consumer = KafkaConsumer(group_id='simple',bootstrap_servers='198.18.134.26:9092', auto_offset_reset='latest')
consumer.subscribe('test')

for message in consumer:
    cdg_message=message.value
    envelope = telemetry_pb2.Telemetry()
    envelope.ParseFromString(cdg_message)
    print(envelope)

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:wq!
```

## Step 3

Execute the python script (ex: **python test.py**). Script output will show same device session as shown on previous task. It will show Cisco Data gateway message header and the actual device session message in the string_value filed (it may take up to 60 secs to update)

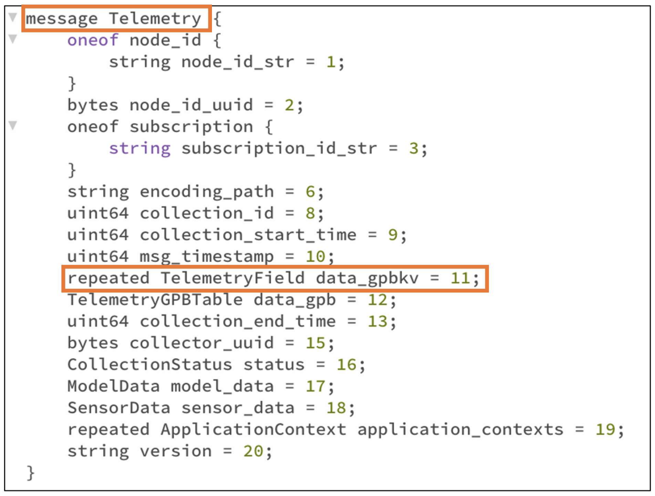


```
node_id_str: "Node-5"
node_id_uuid: "\260mU\323\321\3350\231\257_<\344\016\007\2052"
collection_id: 360
collection_start_time: 1652371661693
msg_timestamp: 1652371661799
data_gpbkv {
  timestamp: 1652371661799
  name: "clisensor.path.command"
  string_value: "<result><iosRequest><![CDATA[show interfaces gigabitethernet 0/0/0/0]]></iosRequest><iosResult><![CDATA[\n\rThu May 12
erface state transitions: 1\n  Hardware is GigabitEthernet, address is fa16.3e7c.2338 (bia fa16.3e7c.2338)\n  Description: to Node-1\n
000 Kbit)\n     reliability 255/255, txload 0/255, rxload 0/255\n  Encapsulation ARPA,\n  Duplex unknown, 1000Mb/s, link type is force-
set,\n  Last link flapped 1w1d\n  ARP type ARPA, ARP timeout 04:00:00\n  Last input 00:00:00, output 00:00:00\n  Last clearing of \"sho
s/sec\n  30 second output rate 1000 bits/sec, 0 packets/sec\n     288293 packets input, 157662464 bytes, 24 total input drops\n     0 c
s, 0 multicast packets\n              0 runts, 0 giants, 0 throttles, 0 parity\n     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignor
ps\n     Output 2 broadcast packets, 0 multicast packets\n     0 output errors, 0 underruns, 0 applique, 0 resets\n     0 output buffer
</iosResult><parserOutput/></result>"
}
collection_end_time: 1652371661798
collector_uuid: "00723fd7-d85a-44a7-a982-78b15591c7c0:CLI_COLLECTOR"
status {
  status: SUCCESS
}
model_data {
}
sensor_data {
  cli_sensor {
    command: "show interfaces gigabitEthernet 0/0/0/0"
  }
}
application_contexts {
  application_id: "application1"
  context_id: "CLI"
}
version: "1"
```

## Step 4

Now we want to remove the Cisco Data gateway message header and extract the session content only.

From the proto message definition on **Task3** you may remember the Telemetry message made of repeated TelemetryFiled "**data_gpbkv**". It will be basically an array of device messages



```
message Telemetry {
    oneof node_id {
        string node_id_str = 1;
    }
    bytes node_id_uuid = 2;
    oneof subscription {
        string subscription_id_str = 3;
    }
    string encoding_path = 6;
    uint64 collection_id = 8;
    uint64 collection_start_time = 9;
    uint64 msg_timestamp = 10;
    repeated TelemetryField data_gpbkv = 11;
    TelemetryGPBTable data_gpb = 12;
    uint64 collection_end_time = 13;
    bytes collector_uuid = 15;
    CollectionStatus status = 16;
    ModelData model_data = 17;
    SensorData sensor_data = 18;
    repeated ApplicationContext application_contexts = 19;
    string version = 20;
}
```

## Step 5

We need to extract **data_gpbkv** (just first one) and, from there, only print the **string_value** parameter seen before.

Terminate script execution with **Ctrl-C**.
Open the python script again and:

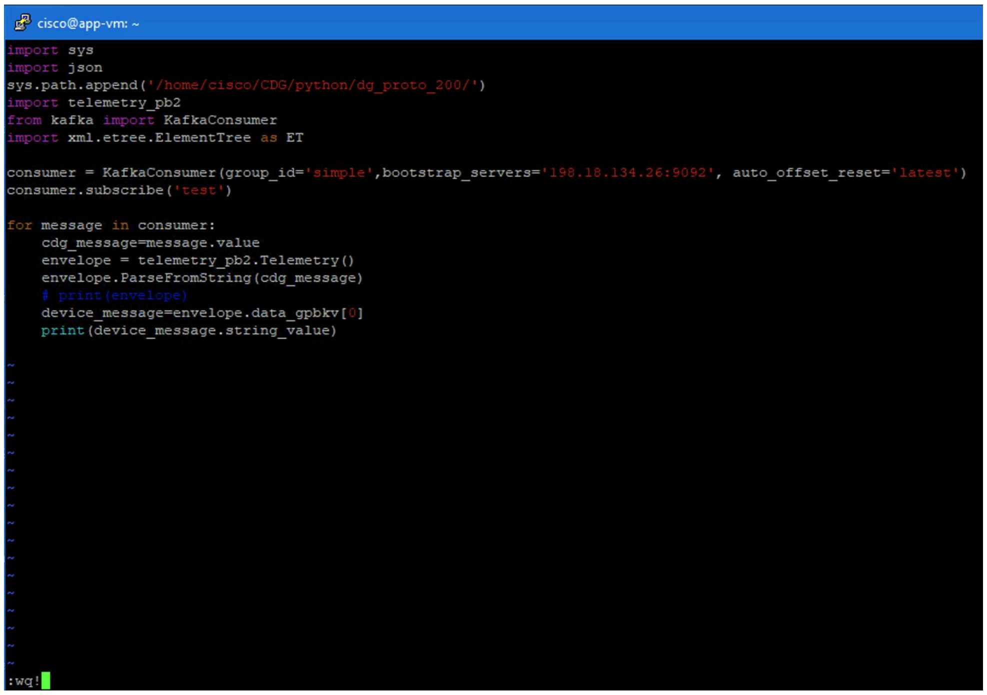
- Comment line

  ```
  print(envelope)
  ```

- Add lines:

  ```
  device_message=envelope.data_gpbkv[0]
  print(device_message.string_value)
  ```

Save it



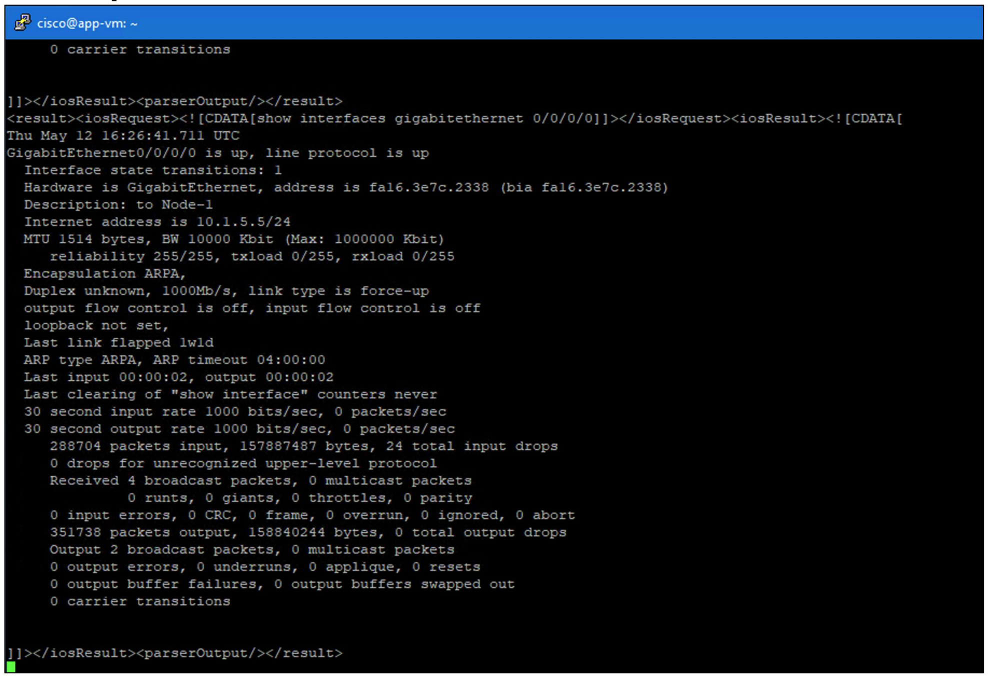
cisco@app-vm: ~

```
import sys
import json
sys.path.append('/home/cisco/CDG/python/dg_proto_200/')
import telemetry_pb2
from kafka import KafkaConsumer
import xml.etree.ElementTree as ET

consumer = KafkaConsumer(group_id='simple',bootstrap_servers='198.18.134.26:9092', auto_offset_reset='latest')
consumer.subscribe('test')

for message in consumer:
    cdg_message=message.value
    envelope = telemetry_pb2.Telemetry()
    envelope.ParseFromString(cdg_message)
    # print(envelope)
    device_message=envelope.data_gpbkv[0]
    print(device_message.string_value)

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:wq!
```

## Step 6

Execute same python script again (it may take up to 60 secs to update). This time script output will only show the session with Cisco data Gateway in **XML** format, made of two elements you can recognize by the XML tag

**<iosRequest>** and **<iosResult>**



```
cisco@app-vm: ~
    0 carrier transitions


]]></iosResult><parserOutput/></result>
<result><iosRequest><![CDATA[show interfaces gigabitethernet 0/0/0/0]]></iosRequest><iosResult><![CDATA[
Thu May 12 16:26:41.711 UTC
GigabitEthernet0/0/0/0 is up, line protocol is up
  Interface state transitions: 1
  Hardware is GigabitEthernet, address is fa16.3e7c.2338 (bia fa16.3e7c.2338)
  Description: to Node-1
  Internet address is 10.1.5.5/24
  MTU 1514 bytes, BW 10000 Kbit (Max: 1000000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation ARPA,
  Duplex unknown, 1000Mb/s, link type is force-up
  output flow control is off, input flow control is off
  loopback not set,
  Last link flapped 1w1d
  ARP type ARPA, ARP timeout 04:00:00
  Last input 00:00:02, output 00:00:02
  Last clearing of "show interface" counters never
  30 second input rate 1000 bits/sec, 0 packets/sec
  30 second output rate 1000 bits/sec, 0 packets/sec
     288704 packets input, 157887487 bytes, 24 total input drops
     0 drops for unrecognized upper-level protocol
     Received 4 broadcast packets, 0 multicast packets
              0 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     351738 packets output, 158840244 bytes, 0 total output drops
     Output 2 broadcast packets, 0 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions


]]></iosResult><parserOutput/></result>
```

## Step 7

Having time, you can further play with the output.

As said before, this task target was to show how to "consume" CDG messages from the Kafka server and extract message payload using python. All steps run here applies to all protocols (gNMI, SNMP, MDT, etc). Further output parsing would then depend on the specific protocol being used.

Following two steps apply to CLI only and can be used to extract the <iosResult> only (it will be the second element in the XML tree hierarchy).

Terminate script execution with **Ctrl-C**.
Open the python script again and:
Comment line

```
print(device_message.string_value)
```

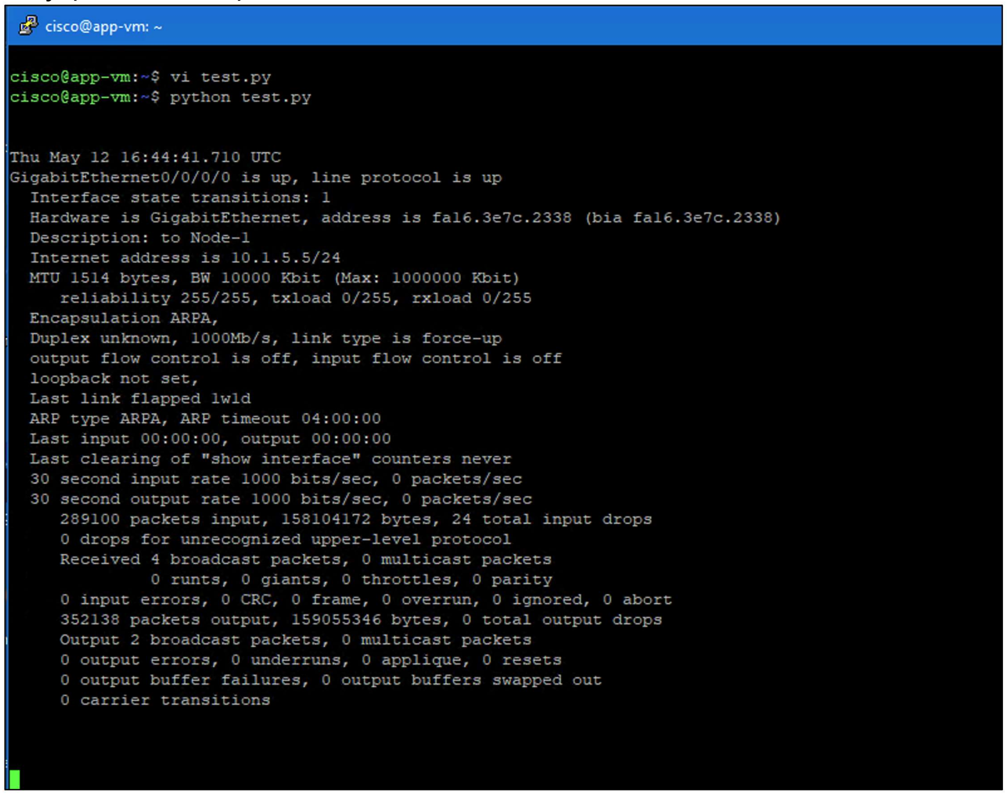
Add lines:

```
root = ET.fromstring(device_message.string_value)
print(root[1].text)
```

Save it

## Step 8

Execute same python script again. This time script output will show the device reply only (<iosResult>)



cisco@app-vm: ~

```
cisco@app-vm:~$ vi test.py
cisco@app-vm:~$ python test.py


Thu May 12 16:44:41.710 UTC
GigabitEthernet0/0/0/0 is up, line protocol is up
  Interface state transitions: 1
  Hardware is GigabitEthernet, address is fa16.3e7c.2338 (bia fa16.3e7c.2338)
  Description: to Node-1
  Internet address is 10.1.5.5/24
  MTU 1514 bytes, BW 10000 Kbit (Max: 1000000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation ARPA,
  Duplex unknown, 1000Mb/s, link type is force-up
  output flow control is off, input flow control is off
  loopback not set,
  Last link flapped 1w1d
  ARP type ARPA, ARP timeout 04:00:00
  Last input 00:00:00, output 00:00:00
  Last clearing of "show interface" counters never
  30 second input rate 1000 bits/sec, 0 packets/sec
  30 second output rate 1000 bits/sec, 0 packets/sec
     289100 packets input, 158104172 bytes, 24 total input drops
     0 drops for unrecognized upper-level protocol
     Received 4 broadcast packets, 0 multicast packets
              0 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     352138 packets output, 159055346 bytes, 0 total output drops
     Output 2 broadcast packets, 0 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
```

**This concludes task 8.**