Automate Service Provisioning with EPNM API

Tahsin Chowdhury, Technical Solutions Specialist

Rex Spell, Technical Solutions Architect

Amer Arch MIG, Cisco Systems Inc.

This project has been developed to show the Northbound API capability of Cisco's Evolved Network Programmable Manager (EPNM). EPNM has a good set of REST APIs and RESTConf APIs to achieve several tasks through third party interfaces (scripts, tools, etc.). This project focuses on bulk service provisioning task across the network using EPNM's RESTConf API.

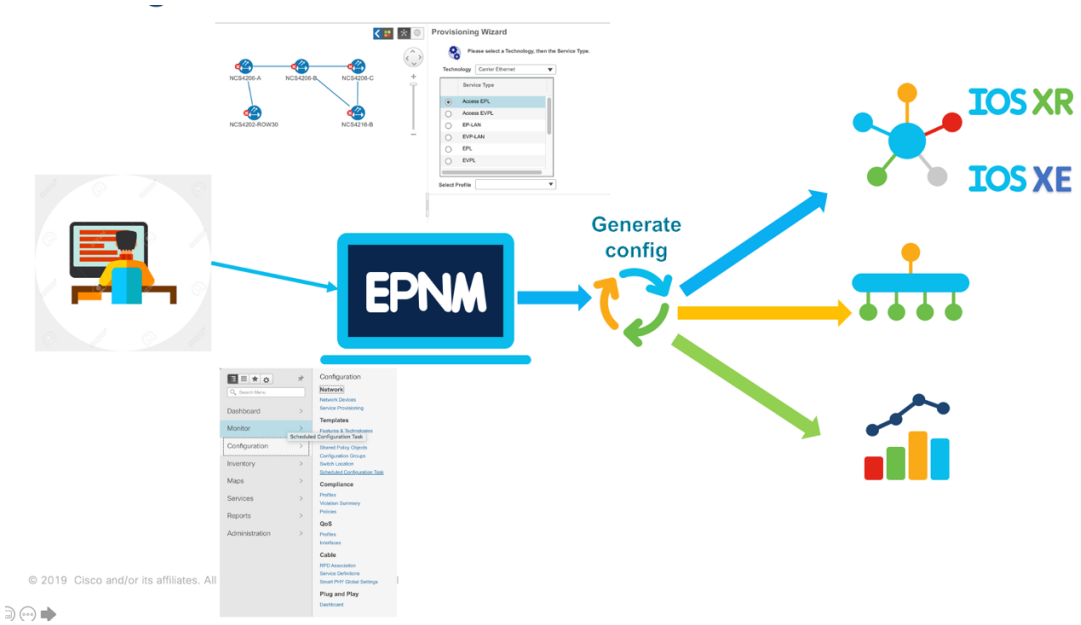
Customers' Need:

- Bulk service provisioning:
    - Customers have hundreds or thousands of services to provision to support end users.
    - It is hard to create services one by one when there are many.
    - Copying & Pasting a large config file is not the ultimate solution. This is error prone.
- Automate the service provisioning procedure.
    - Automate the services provisioning steps.
    - Less error prone.
    - One place to change parameters, applies everywhere accordingly.

Provisioning with EPNM Web Client vs Automated Script/Tool using EPNM Northbound API:

Provisioning with EPNM Web Client:

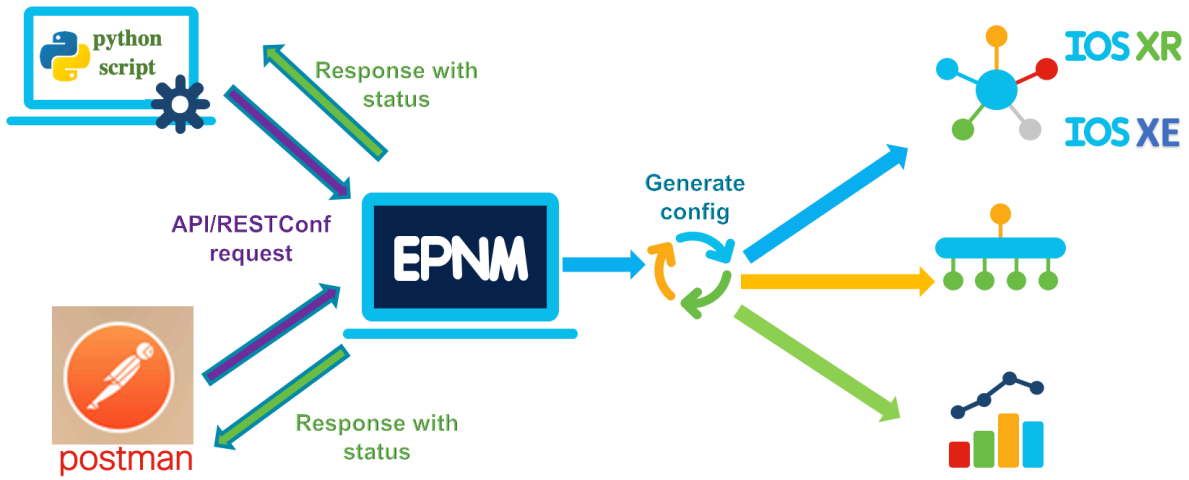
- Time consuming: users need to use provisioning wizard every time they create a new service, i.e. one service provisioning at a time.



- Services will only be discovered in EPNM but requires the extra step of promoting the service in EPNM which is also one service promotion at a time.

Automated Script/Tool using EPNM Northbound API:

- Using script instead of Provisioning Wizard.
- Much easier than provisioning each service individually through 'point and click' GUI.
- Ready to manage: Services will be treated as those have been created via EPNM.



API Resources

- **RESTful API:** REST Based Web services by calling HTTP Methods .
- **RESTConf API:** RESTCONF uses HTTP methods to provide CRUD operations on a conceptual datastore containing YANG-defined data.

HTTP Verb	Typical Action (CRUD)
POST	**Create**
GET	**Read**
PUT	**Update**
PATCH	Update
DELETE	Delete

API Structure:

https://{EPNM_SERVER_IP}/webacs/api/v1/op/info/version

- Protocol
- Server/host
- Resource path
- (optional) Parameters (based on context of filtering)

Common REST API status:

Status Code	Status Message	Meaning
200	OK	All looks good
201	Created	New resource created
400	Bad Request	Request was invalid
401	Unauthorized	Authentication missing or incorrect
403	Forbidden	Request was understood but not allowed
404	Not Found	Resource not found
500	Internal Server Error	Something wrong with the server
503	Service Unavailable	Server is unable to complete request

Data Format (Either as payload for POST/PUT API call or as response):

- **JSON:**


```
{
  "mgmtResponse": {
    "@responseType": "operation",
    "@requestUrl": "https://{EPNM_IP}/webacs/api/v1/op/info/version"
,
    "@rootUrl": "https://{EPNM_IP}/webacs/api/v1/op",
    "versionInfoDTO": {
      "result": "3.1.0.0.000"
    }
  }
}
```


Where,

- **data is 'key: value' pair**

- **XML:**

```
<?xml version="1.0" ?>

<mgmtResponse responseType="operation" requestUrl="https://{EPNM_IP}/webacs/api/v1/op/info/version" rootUrl="https://{EPNM_IP}/webacs/api/v1/op">

  <versionInfoDTO>

    <result>3.1.0.0.000</result>

  </versionInfoDTO>

</mgmtResponse>
```

  Where the contents can be viewed as,

  - document node: **mgmtResponse**
  - **elements: versionInfoDTO** (Parent), **result** (Child)
  - data: **3.1.0.0.000**

Targeted Platform for this project:

- We have targeted Cisco's **NCS4200** platform for provisioning TDM services. The project focuses on the services types we generally discuss with our customers in North America. The project supports both SAToP (framed and unframed) and CEP services expanded as follows:
  - T1 SAToP (framed and unframed) using T1 controller
  - T1 SAToP (framed and unframed) using T3 controller (channelized)
  - T3 using T3 controller
  - STS-1, STS-3c, STS-12c, STS-48c CEP on SONET controller
  - T1 under CT3 or VT1.5 mode of STS-1 channel on SONET controller
  - T3 under T3 mode of STS-1 channel on SONET controller
  - VT CEP under VT1.5 mode of STS-1 channel on SONET controller
  - DS3 CEP on T3 controller, EPNM will treat it as STS-1 CEP service Type
- The project does not support currently the following items:
  - DS0 CESoP
  - STS1e. EPNM does not support STS1e yet, but NCS4200 platform does support this service type.

Prerequisite in System (EPNM and Platform):

- EPNM Environment (This script was tested with EPNM 3.1.2 and 4.0 versions)
- NCS4200 with required software image load (This script was tested with IOS-XE 16.12.1 and 17.1.1 software images).
- The script only focuses on service provisioning or deleting task only. Users need to enable the controller ports (T1, T3, SONET), framing types i.e. CBIT/M13, SF/ESF, aka all the controller settings either using CLI or EPNM web client, to keep the script simple.

Tools used for developing the project:

1. Postman: At first, the postman tool was used to check and verify EPNM's API resources and payload. Postman also provides code for the API call in various programming language.
2. **Python:** Finally, python was used to develop the script. PyCharm has been used as the development IDE. Python version 3.7.6 was used here.
3. **Excel:** An excel (.xlsx) file was used for gathering all the required parameters for creating multiple services.

The steps are very simple in general:

1. Extract data from an excel file.
2. Create payload (json or xml format) using the data for EPNM's API call.
3. Use the payload with the API call for the specific task.
4. Check the response of the API call.

Structure of the Project:

The project has the following folders/scripts:

1. platforms:

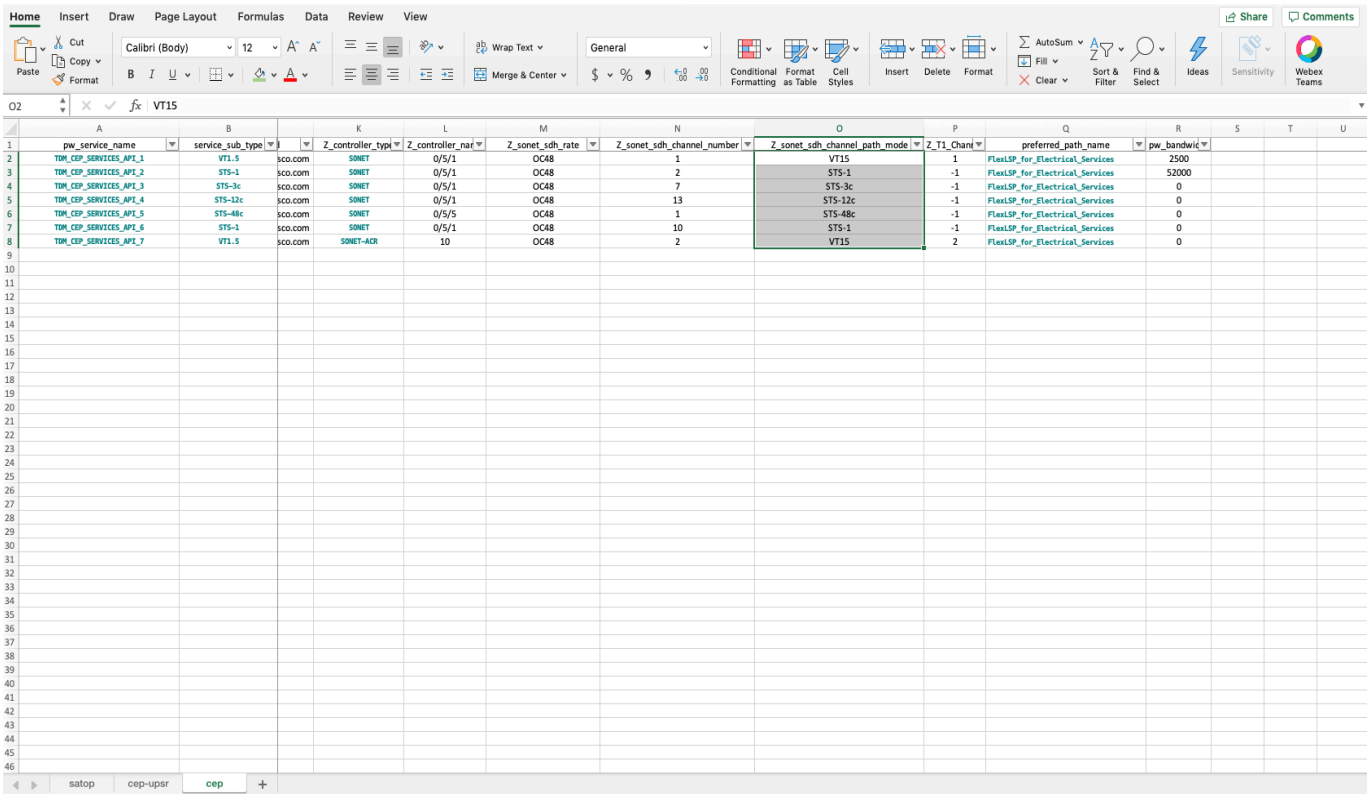
This section has all the user defined functions in different python files to create data payload format for API calls, data extraction from excel, several utility functions and examples of the actions for specific platform.

a. **ncs4200:** All functions, utils and examples related to cem service provisioning using NCS4200.

   i. **utils:** This folder contains,

      - python file for SAToP payload template creation
      - python file for CEP payload template creation

- python file for service deletion payload template
- python file for extract data from an excel file
- Excel files with different service types under different sheet names.
    - Providing excel files for both EPNM 3.1 and EPNM 4.0 since some of the data have been modified in EPNM 4.0 which required changes in the data to create payload for API call.
    - The python files above were created based on the data in excel. Any change in column name of the excel file or additional columns in the excel, the user is required to change in the function definitions in the python files.
- For cem service provisioning, EPNM provides a default Bandwidth for PSeudowire which cannot be changed with any API parameter but it can be achieved by calling an EPNM template via EPNM API. Users need to import this template if they want to change the default pseudowire bandwidth value. In the future this will be removed with an EPNM API, and there will be a BW parameter for CEM pseudowires. Users can avoid using the changed bandwidth value with the EPNM template by simply putting 0 under “pw_bandwidth field” (shown below).



- The excel file was divided into SAToP, CEP and CEP-UPSR. CEP-UPSR was created separately for simplicity. The user needs to choose the exact name matching with the excel sheetnames while choosing service types in the script.
- MPLS Tunnel path has to be already created in EPNM, and the name of the path will be added as preferred_path section in the excel. The user can simply keep it blank if they do not want to use it.
- For T1 channels, SONET sts channel numbers, simply put -1 if not applicable for a particular service type. For VT type, T1 channel number will be converted (coded in the script) into specific VTG X VT Y. As an example, T1 Channel 28 will become VTG 7 T1 4 for T1 under VT1.5 mode for T1 SAToP service type under SONET controller or VTG 7 VT 4 for VT1.5 CEP service type under SONET controller.
- Use "NA" where other fields are not required. Looking at the example excel will provide an idea where to put NA or "-1" or 0 or blank in the excel file.

ii. action_modules:

This folder contains action modules:

- service provisioning module file containing a single user defined function which calls all the other required utility and functions to create cem services.
- Service deletion module file containing a single user defined function which calls all the other required utility and function to delete cem services.

iii. action_examples:

- This folder contains two examples of calling service provisioning and deletion module with users input.
- These files are enough to use for cem service provisioning and deletion in bulk if the user has information based on the example excel template provided.
- A simple GUI as an example of how different tools can be developed using the action_module sections.

b. Any other platform (see later on the section how to add another platform)

c. Two folders which provide final reports of python script execution and error occurred while deploying the service in EPNM (any error occurred on southbound side i.e. from EPNM to Network devices direction)

2. utils:

This folder contains some common utility functions can be used across all the platforms added in the platform folder.

i. This has user defined class “epnm_request_lib” for calling EPNM API by wrapping python’s “request” library.

ii. A common file providing EPNM API resources for different purposes applicable for the project. More can be added depending the future scope of the project.

iii. User defined function for checking and reporting API call response.

- **WebEx Teams Bot notifier:** The script provides a cool feature to notify any WebEx team user the brief response of every service creation of deletion with the status either successful or failed. Please look at the reference section to go to the link where the user can create WebEx teams bot. The script will ask for the Bot’s access token and recipient’s WebEx team email id if this option is enabled. This option is added with action_modules.
- **Email notification:** The script also provides a way to send the final response as text file using Gmail SMTP server. Users need to give a sender’s Gmail id, sender’s google app password (see reference section for generating app password) and receiver’s email id. Receiver email can be any email id, i.e. Gmail, outlook etc. Multiple receivers can be added separating a comma. This feature is also optional and not added to action_modules. It has been added separated under the main entry point of the script “automate_epnm.py” for NCS4200 section.

3. automate_epnm.py:

This is the main entry point of the script. This can be treated as a bigger umbrella where actions for different platforms can be added. Python “Click” library has been used here to provide a command line environment while running the script. This will ask for user inputs for running the action_modules for a particular platform.

The purpose of making the simple 4-step script into different utils, platforms and action files is to make the code modular so that future development becomes easier and modules can be reused as needed. A very simple version of the script can be found in another GitHub link given below:

https://github.com/tchowdhu/Configuring-network-calling-EPNMs-North-Bound-API-through-python-script

Accessing and using the code:

Users have different options to utilize the developed code.

1. Simply calling “automate_epnm.py”. This is preferable.

   i. Users need to see available commands first.

   $ **python3 automate_epnm.py**

```
$ python3 automate_epnm.py

Usage: automate_epnm.py [OPTIONS] COMMAND [ARGS]...

Options:
  --help  Show this message and exit.

Commands:
  platform-ncs4200
  platform-xr-router
  sync-nodes-from-file
```

   ii. For platform-ncs4200, see the user input options:

   $ python3 automate_epnm.py platform-ncs4200 --help

```
$ python3 automate_epnm.py platform-ncs4200 --help

Usage: automate_epnm.py platform-ncs4200 [OPTIONS]

Options:
  --epnmip TEXT                   Enter EPNM IP address/url
  --username TEXT                 Enter EPNM Username
  --password TEXT
  --action [bulk-provision|bulk-deletion]
                                  Enter action
  --filename PATH                 Enter Data .xlsx file
  --cemtype [satop|cep|cep-upsr]  Enter cem type
  --service TEXT                  Enter number of services
                                  [all: select all
                                  or, start-end: provide range
                                   or Xn: Select
                                  number
                                   or, x1, x2, xn: select multiple]
  --enablewebexbot [True|False]   Enable enable webEx bot notifier
  --accesstoken TEXT              Provide access token of webEx bot access
                                  token['' to avoid]
  --toemail TEXT                  Provide notification receivers webEx
                                  email['' to avoid]
  --enablemailserver [True|False]
                                  Enable mail server (smtp gmail is
                                  implemented)
  --help                          Show this message and exit.
```

iii. Then simply running the script using the command:

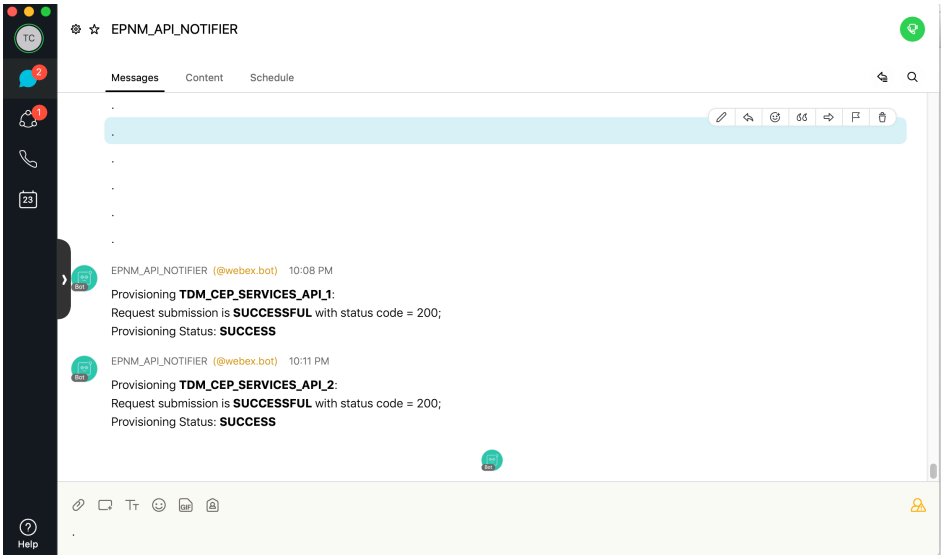
a. Like a terminal command with all the option in single command call:

   $ python3 automate_epnm.py platform-ncs4200 --epnmip 1.1.1.1 --username username ……

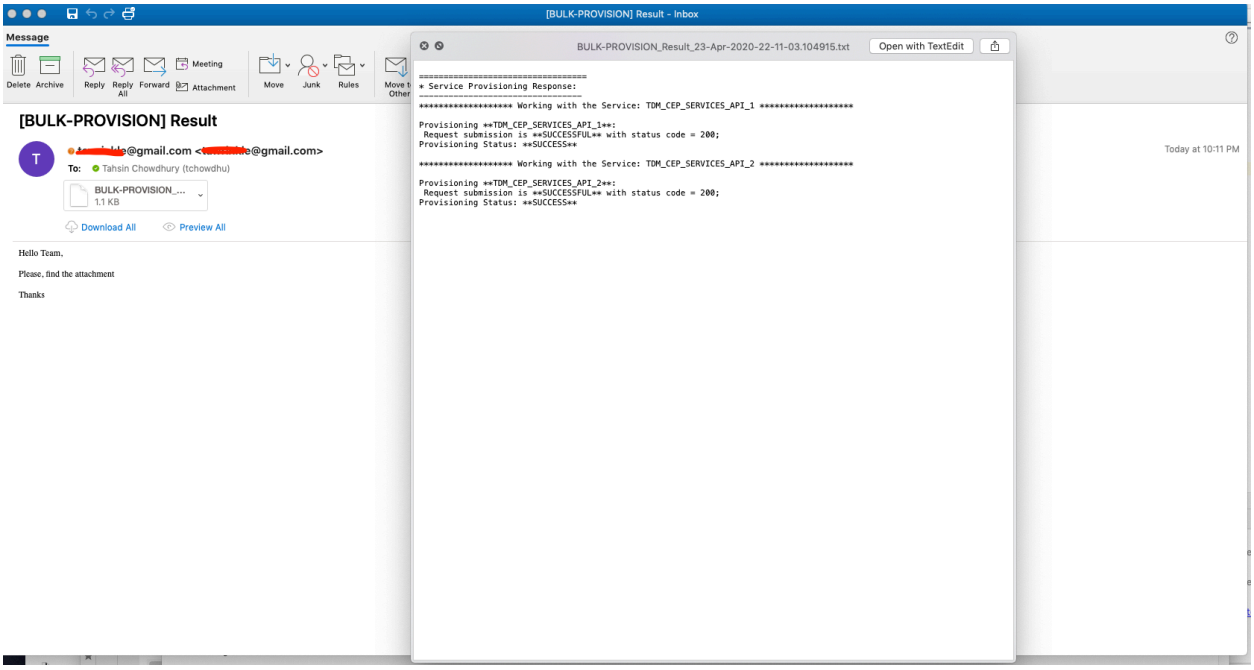
b. Just call automate_epnm.py with the command, the terminal will ask for user inputs one by one. The scripts waits until the EPNM response returns provisioning status as successful and completed.

```
TCHOWDHU-M-W492:Automate_Service_provisioning_with_EPNM_API tchowdhu$ python3 automate_epnm.py platform-ncs4200
Enter EPNM IP address/url:
Enter EPNM IP address/url: 10.89.204.105
Enter EPNM Username: root
Password:
Repeat for confirmation:
Enter action (bulk-provision, bulk-deletion): bulk-provision
Enter Data .xlsx file:
Enter Data .xlsx file: /Users/tchowdhu/PycharmProjects/Automate_Service_provisioning_with_EPNM_API/platforms/ncs4200/utils/tdm_services_list_epnm_4_0.xlsx
Enter cem type (satop, cep, cep-upsr):
Enter cem type (satop, cep, cep-upsr): cep
Enter number of services
[all: select all
 or, start-end: provide range
 or Xn: Select number
 or, x1, x2, xn: select multiple]: 2
Enable enable webEx bot notifier (True, False): True
Provide access token of webEx bot access token['' to avoid]: [redacted]
Provide notification receivers webEx email ['' to avoid]: tchowdhu@cisco.com
Enable mail server (smtp gmail is implemented) (True, False): True
Enter sender's email address: [redacted]@gmail.com
Password:
Enter receiver's email address: tchowdhu@cisco.com
Login response status:  200

==================================
* Service Provisioning Response:
----------------------------------

****************** Working with the Service: TDM_CEP_SERVICES_API_1 ******************

POST response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
Provisioning **TDM_CEP_SERVICES_API_1**:
 Request submission is **SUCCESSFUL** with status code = 200;
Provisioning Status: **SUCCESS**


****************** Working with the Service: TDM_CEP_SERVICES_API_2 ******************


****************** Working with the Service: TDM_CEP_SERVICES_API_2 ******************

POST response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
Provisioning **TDM_CEP_SERVICES_API_2**:
 Request submission is **SUCCESSFUL** with status code = 200;
Provisioning Status: **SUCCESS**

Email Sent Successfully:

Total time of execution: 0:06:33.420022

TCHOWDHU-M-W492:Automate_Service_provisioning_with_EPNM_API tchowdhu$
```

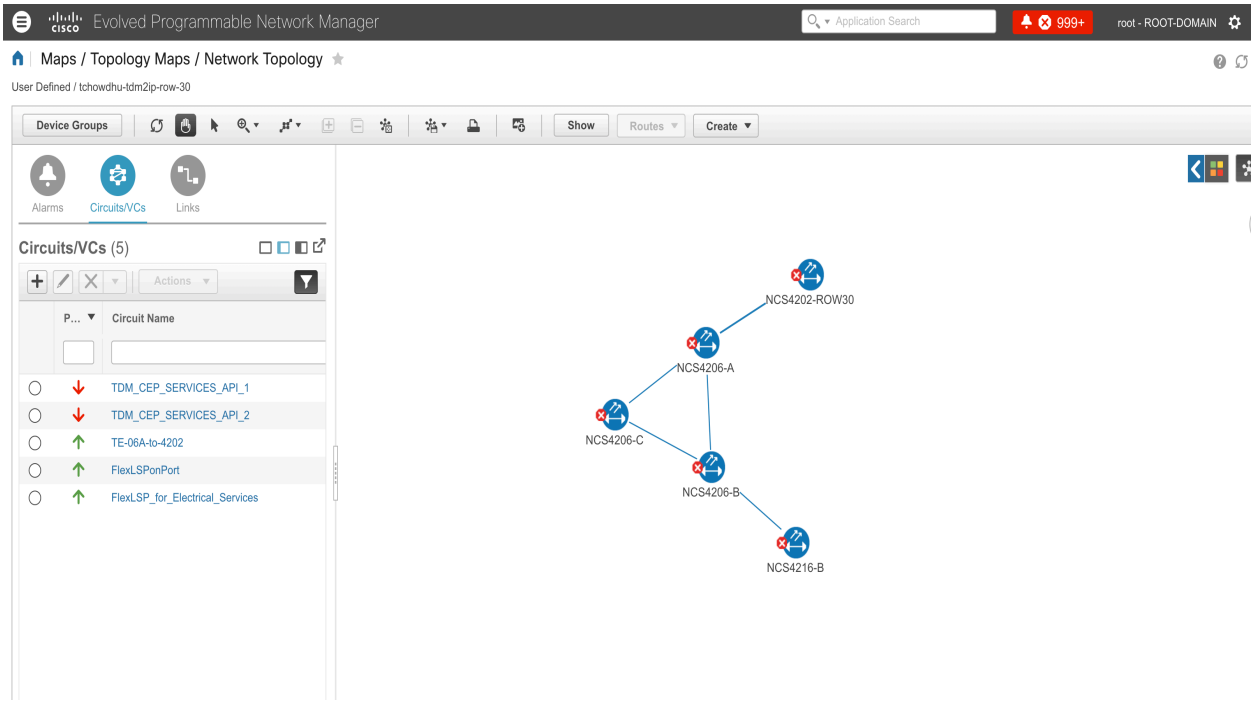
WebEx notification:



Email Notification:



Checking in EPNM:



2. Simply running the following files under actions_example folder.

   “ncs4200_bulk_cem_service_provision.py”

   “ncs4200_bulk_cem_service_deletion.py”

```
[TCHOWDHU-M-W492:Automate_Service_provisioning_with_EPNM_API tchowdhu$ pwd
/Users/tchowdhu/PycharmProjects/Automate_Service_provisioning_with_EPNM_API
[TCHOWDHU-M-W492:Automate_Service_provisioning_with_EPNM_API tchowdhu$ ls -l
total 24
drwxr-xr-x   3 tchowdhu  staff     96 Mar 31 09:49 __pycache__
-rwxr-xr-x   1 tchowdhu  staff  10268 Apr 23 20:06 automate_epnm.py
drwxr-xr-x   8 tchowdhu  staff    256 Apr 23 13:43 platforms
drwxr-xr-x  20 tchowdhu  staff    640 Apr 23 13:43 utils
[TCHOWDHU-M-W492:Automate_Service_provisioning_with_EPNM_API tchowdhu$ cd platforms/ncs4200/actions_example/
[TCHOWDHU-M-W492:actions_example tchowdhu$ python3 ncs4200_bulk_cem_service_provision.py

[TCHOWDHU-M-W492:actions_example tchowdhu$ python3 ncs4200_bulk_cem_service_deletion.py
```

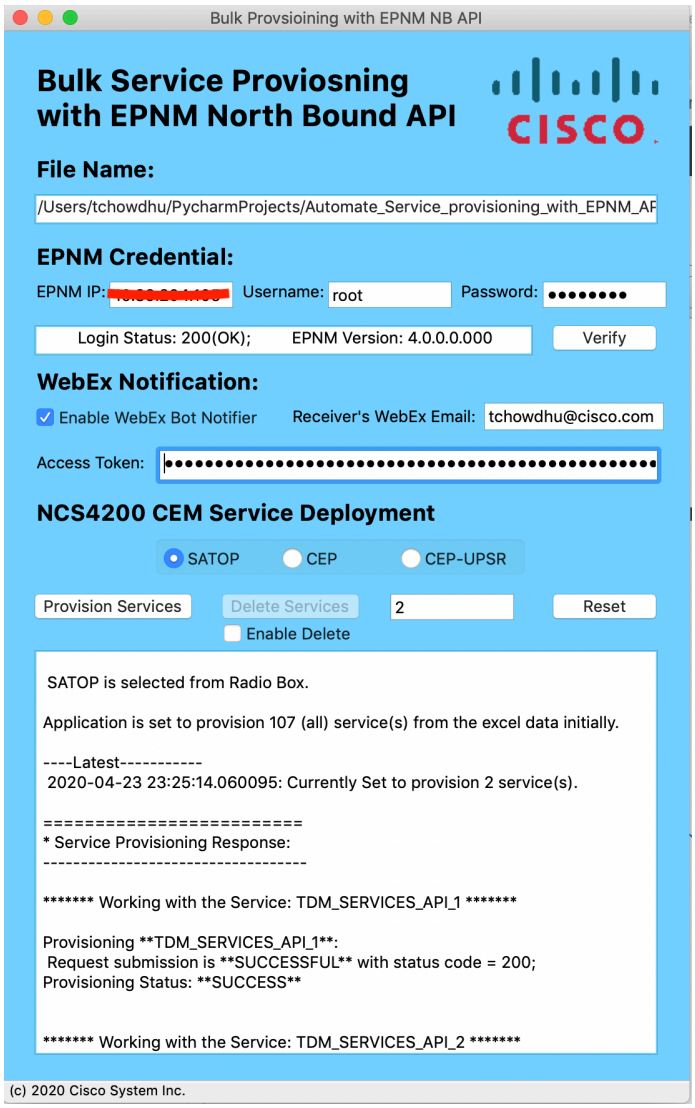
   All the user-inputs will be defined inside the example python scripts.

3. Running the GUI tool under actions_example folder.

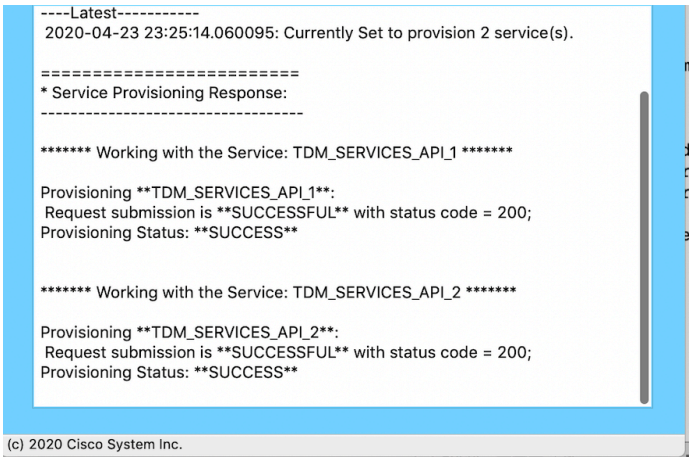
   “NCS4200_BULK_PROV_GUI.py”

   $ python3 NCS4200_BULK_PROV_GUI.py

GUI View of the tool:



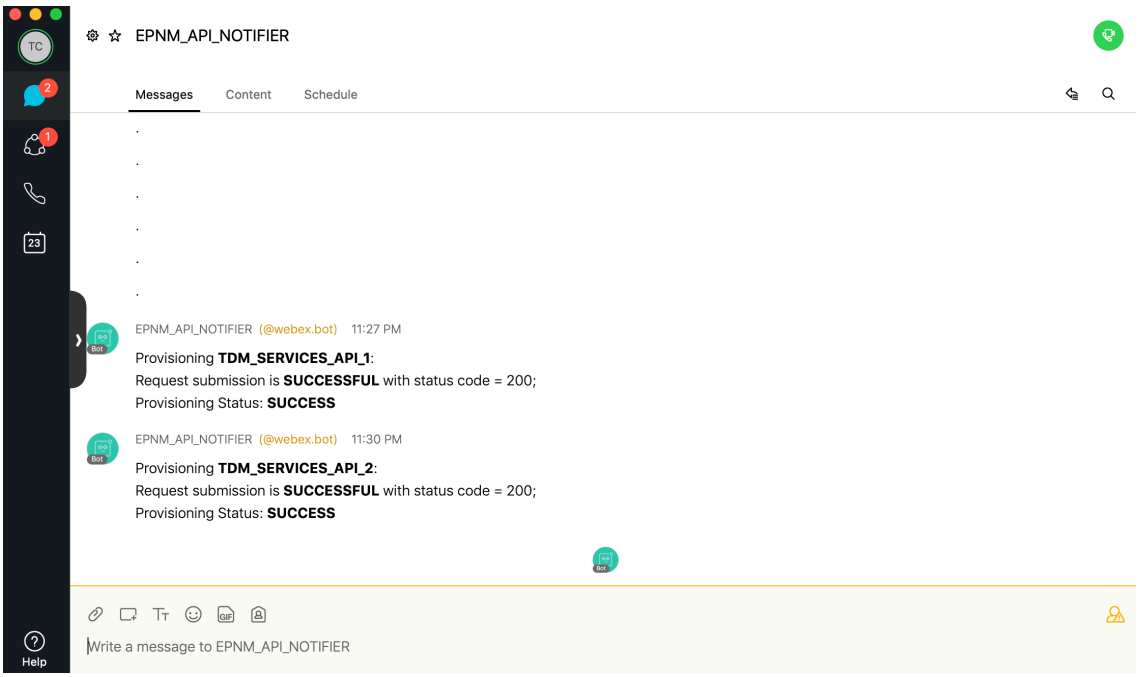
Scroll down to see full view in text field in the GUI tool



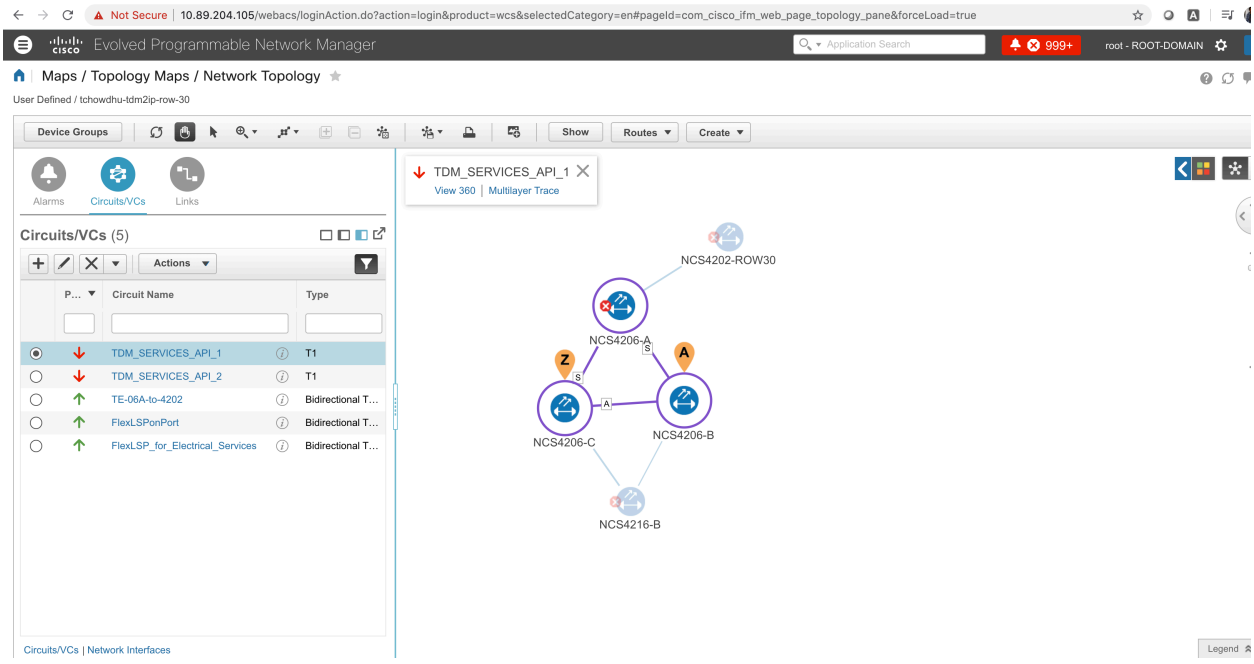
Terminal view running in the background:

```
TCHOWDHU-M-W492:actions_example tchowdhu$ pwd
/Users/tchowdhu/PycharmProjects/Automate_Service_provisioning_with_EPNM_API/platforms/ncs4200/actions_example
TCHOWDHU-M-W492:actions_example tchowdhu$
TCHOWDHU-M-W492:actions_example tchowdhu$ python3 NCS4200_BULK_PROV_GUI.py
0
objc[34988]: Class FIFinderSyncExtensionHost is implemented in both /System/Library/PrivateFrameworks/FinderKit.framework/Versions/A/FinderKit (0x7fffafe613d8) and /System/Library/PrivateFrameworks/FileProvider.framework/OverrideBundles/FinderSyncCollaborationFileProviderOverride.bundle/Contents/MacOS/FinderSyncCollaborationFileProviderOverride (0x129e47f50). One of the two will be used. Which one is undefined.
/Users/tchowdhu/PycharmProjects/Automate_Service_provisioning_with_EPNM_API/platforms/ncs4200/utils/tdm_services_list_epnm_4_0.xlsx
Enter pressed
2
Login response status:  200

==================================
* Service Provisioning Response:
----------------------------------

****************** Working with the Service: TDM_SERVICES_API_1 ******************

POST response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
Provisioning **TDM_SERVICES_API_1**:
 Request submission is **SUCCESSFUL** with status code = 200;
Provisioning Status: **SUCCESS**


****************** Working with the Service: TDM_SERVICES_API_2 ******************

POST response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
GET response status:  200
Provisioning **TDM_SERVICES_API_2**:
 Request submission is **SUCCESSFUL** with status code = 200;
Provisioning Status: **SUCCESS**
```

WebEx Notification:



EPNM Topology:



How to add another platform for service provisioning (A brief development guide):

The main purpose of the project was completed by developing TDM service provisioning script for NCS4200.

Now, say, we want to apply the same concept for other platforms.

An example use case:

"EPL service between xr routers, say between NCS5500 and ASR9K"

Steps:

1. Creating **xr-router** folder under platforms folder
2. Create a **utils** folder under **xr-router**
3. Under utils,
    a. Create python file for data extraction from excel or use the one from main utils folder (here using resource the main utils folder)
    b. Create python file for service creation template,
    c. Create python file for service deletion template (user can skip this)
    d. Excel file for data
4. Create **action_module** folder under **xr-router** folder
    a. Create python file to create a main function for service creation.
    b. Create python file to create a main function for service deletion (user can skip this)

    This step is very simple, the function will include:

    i. Data extraction (calling module from 3a)
    ii. Create provisioning data payload (using templates on step 3b, 3c)
    iii. Apply payload using EPNM API
    iv. Check API response

    for iii, iv, user can use the user define functions and classes from the main **utils** folder.
5. (optional) Create **actions_example** where user can show how to use the function for creating/deleting services. This is an example. The user can achieve everything from the file where the main functions were created (step 4)

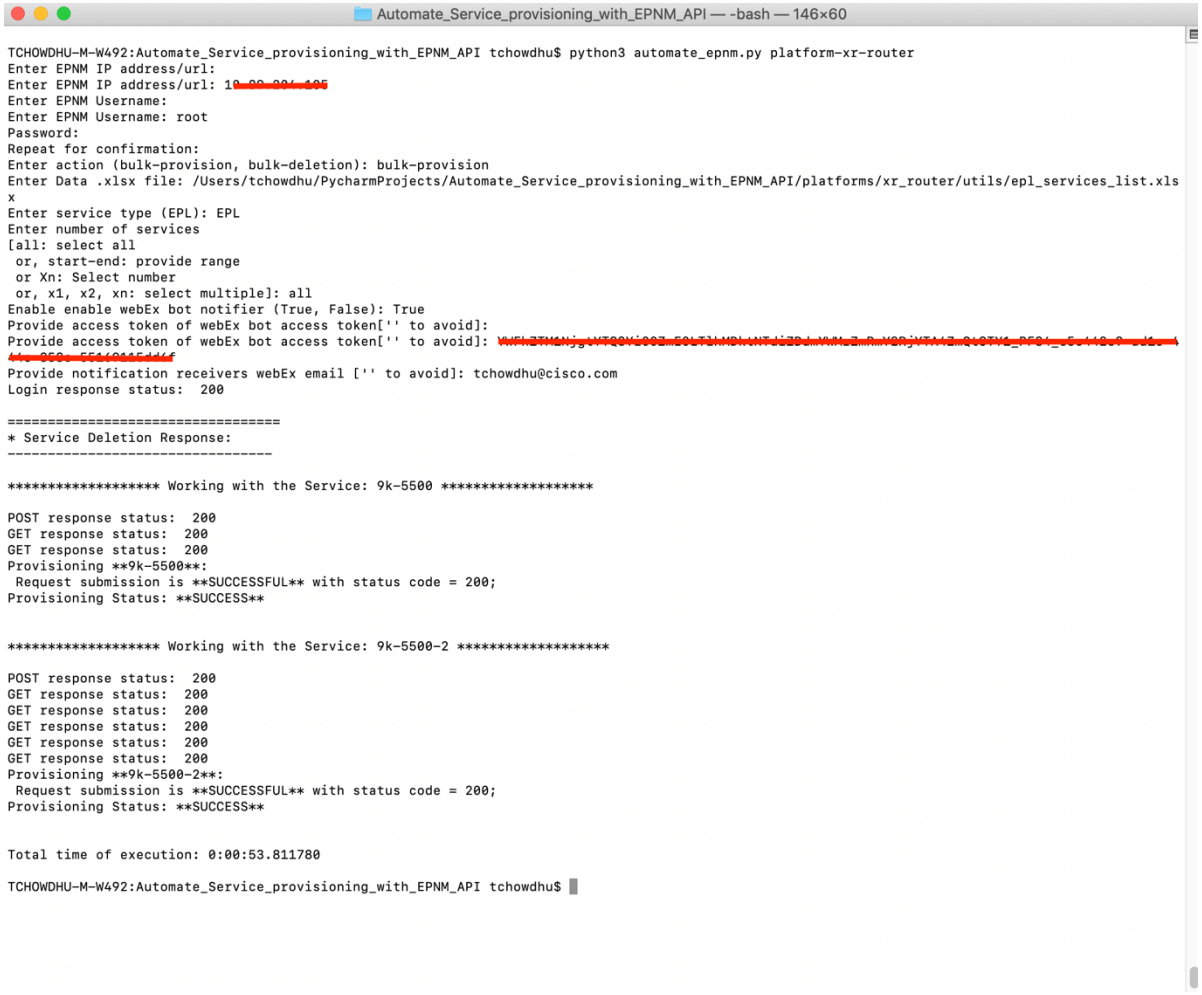
6. (optional) if user wish to add this module in the main section of the project, the main function can be called from **automate_epnm.py** by creating command line user environment with 'Click' library.

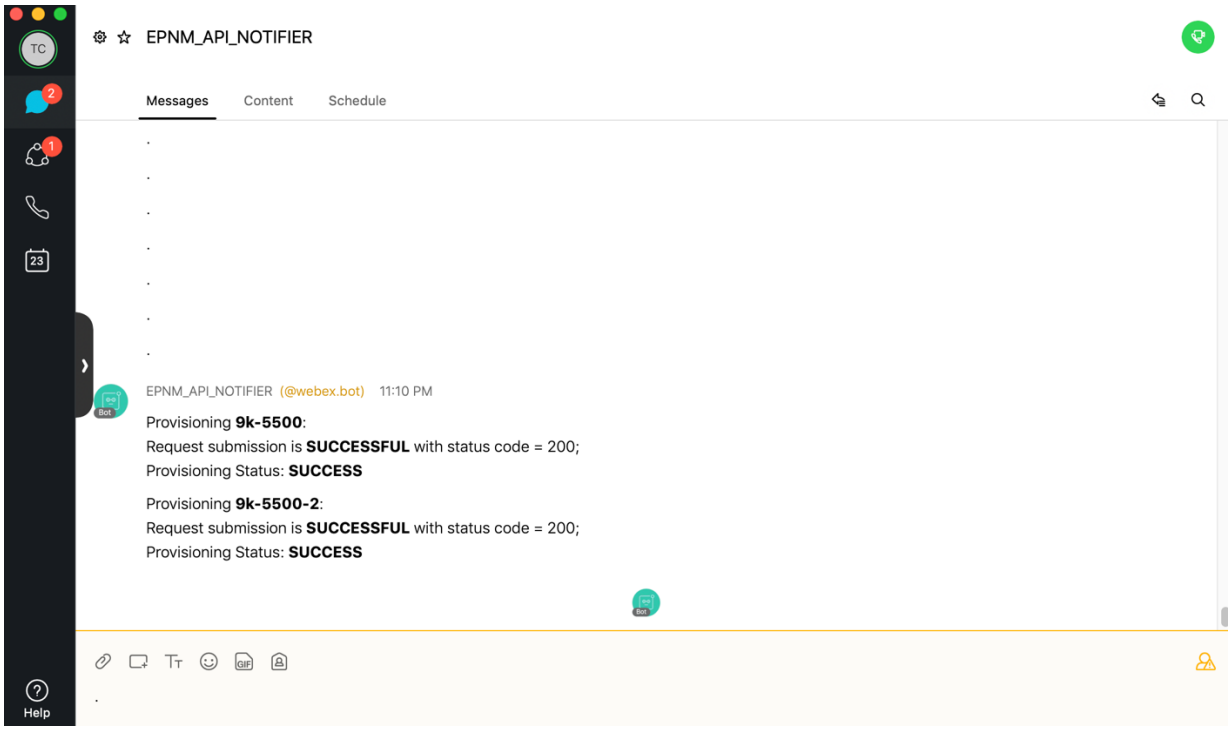
Test and run the command with result for EPL services for Cisco IOS-XR router platforms:

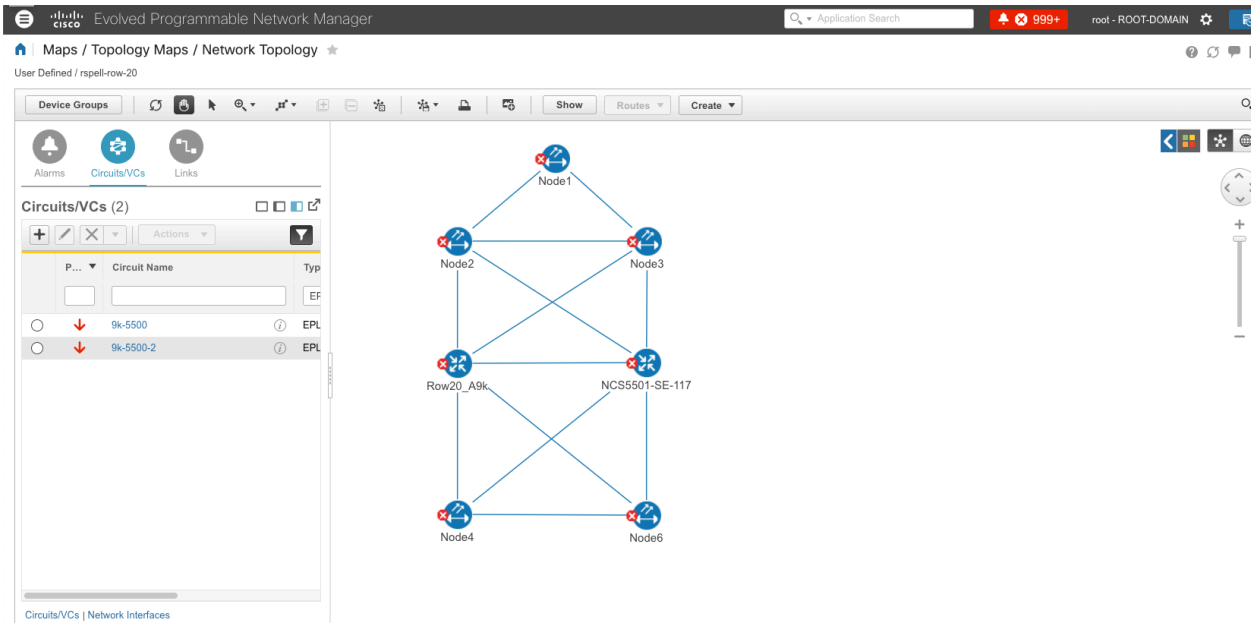
1. Check the command options:

```
Automate_Service_provisioning_with_EPNM_API $   python3 automate_epnm.py
Usage: automate_epnm.py [OPTIONS] COMMAND [ARGS]...

Options:
  --help  Show this message and exit.

Commands:
  platform-ncs4200
  platform-xr-router
  sync-nodes-from-file
Automate_Service_provisioning_with_EPNM_API $   python3 automate_epnm.py platform-xr-router --help
Usage: automate_epnm.py platform-xr-router [OPTIONS]

Options:
  --epnmip TEXT                   Enter EPNM IP address/url
  --username TEXT                 Enter EPNM Username
  --password TEXT
  --action [bulk-provision|bulk-deletion]
                                  Enter action
  --filename PATH                 Enter Data .xlsx file
  --servicetype [EPL]             Enter service type
  --service TEXT                  Enter number of services
                                  [all: select all
                                  or, start-end: provide range
                                   or Xn: Select
                                  number
                                   or, x1, x2, xn: select multiple]
  --enablewebexbot [True|False]   Enable enable webEx bot notifier
  --accesstoken TEXT              Provide access token of webEx bot access
                                  token['' to avoid]
  --toemail TEXT                  Provide notification receivers webEx
                                  email['' to avoid]
  --help                          Show this message and exit.
```

2. Run and Result:







The purpose of developing the project in a modular way so that users can use the section or modules of the codes in various ways they wish to use,

a. either just running as a separate python script (Step 5 or step 4) or,
b. using the sections to create GUI tool (as we did with NCS4200) or,
c. using sections to put it under the umbrella of a bigger project (as we did with step 6).

This can be even better both in terms of the way it was developed and also by adding more content. The purpose is to show EPNM's Northbound API capability to solve various scenarios that will help our customer understand cisco products and feel comfortable with team for their network.

Understand the EPNM in General:

It is important to understand how EPNM works first, what it supports and what it does not.

Key points to understand about EPNM regardless using web client or APIs:

1. Users can use EPNM for the network platforms that EPNM supports.
2. Users can work with the features that is supported in EPNM for that particular platforms. EPNM may not have all the features of a platform developed in the current EPNM.
3. Thus, users can utilize Northbound API resources those were developed for the supported features of different platforms supported in EPNM. There may have some small task API resources for which is not yet developed, so users may need to rely on Web Client.

Reference:

- This is a very good site to start with: https://developer.cisco.com/site/epnm/
- Cisco.com general resource:
  https://www.cisco.com/c/en/us/td/docs/net_mgmt/epn_manager/3_0/user/guide/bk_CiscoEPNManager_3_0_UserAndAdministratorGuide/bk_CiscoEPNManager_3_0_UserAndAdministratorGuide_appendix_0100001.html
- Programming guide at the link below: provides a set of examples.
  https://www.cisco.com/c/en/us/support/cloud-systems-management/evolved-programmable-network-manager-3-1/model.html
- https://<epnm-server-address>/nbi_help/index.html
- EPNM 4.0 is out, see cisco.com links for that version too.
  https://www.cisco.com/c/en/us/td/docs/net_mgmt/epn_manager/4_0_0/user/guide/bk_CiscoEPNManager_4_0_UserAndAdministratorGuide/bk_CiscoEPNManager_4_0_UserAndAdministratorGuide_appendix_0100000.html

  https://www.cisco.com/c/en/us/support/cloud-systems-management/evolved-programmable-network-manager-4-0/model.html
- For WebEx teams bot: https://developer.webex.com/my-apps/new/bot
- Google app password: https://myaccount.google.com/apppasswords