IOS-XE router with a Webex Teams BOT

INTRODUCTION

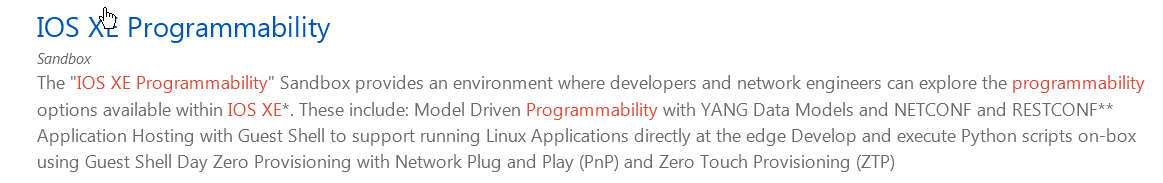
In this POC you will be shown how to configure the following items:

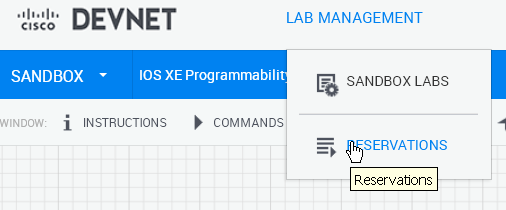
1. How to create a Webex Teams BOT and test the communication between the router and a Webex Teams space
2. How to enable IOx on a CSR1000v (running IOS-XE 16.8) to create a Guest Shell container instance
3. How to add a python script to the Guest Shell that triggers the Webex Teams BOT notification.
4. How to add an EEM script to trigger the Guest Shell python script
5. Review how users can integrate an IOS-XE router or a Catalyst 9K switch running Guest Shell with a Webex Teams BOT that could be used in operations Webex Teams group Chat, for improving problem resolution.

EQUIPMENT AND LAB LINKS-

Webex Teams Bot: <https://developer.webex.com/>

IOS-XE sandbox: <https://developer.cisco.com/site/sandbox/> Search :IOS XE Programmability NETCONF-RESTCONF-YANG





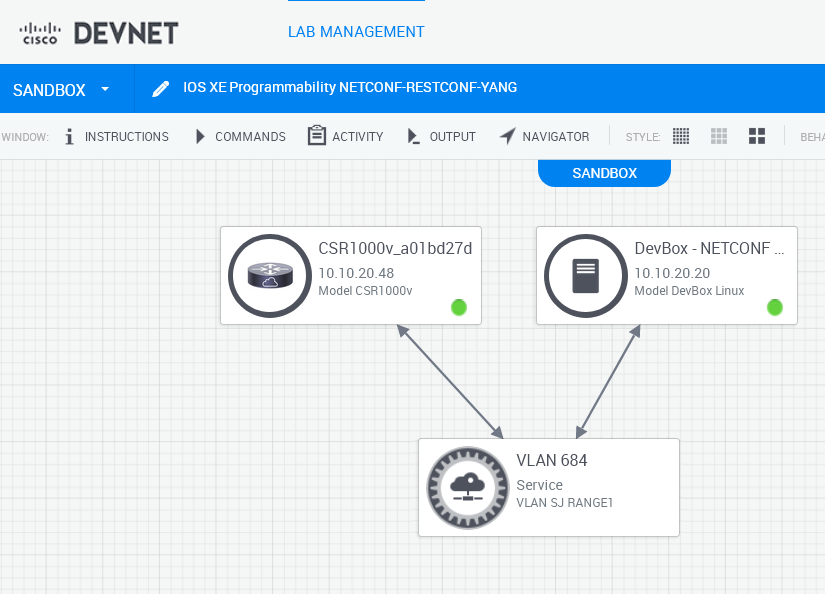
Schedule the a session with this lab or any of the sandbox labs that have an IOS-XE device, router or switch

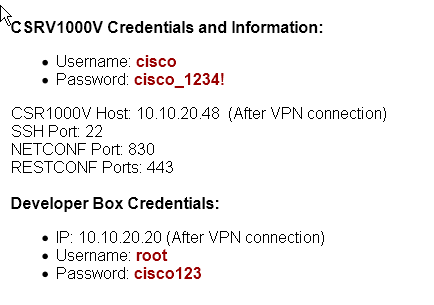
CSR1000v Guestshell Only Lab: https://learninglabs.cisco.com/modules/net\_app\_hosting/intro-guestshell/step/1

DESIGN AND TOPOLOGY DIAGRAM:

<https://devnetsandbox.cisco.com/RM/Diagram/Index/da3261a1-a3ce-4fb3-8730-9d51398513ef>

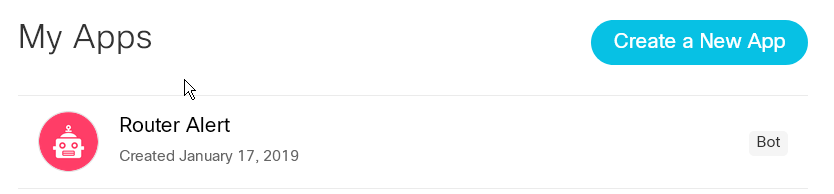
or <https://devnetsandbox.cisco.com/RM/Diagram/Index/1b83c4bf-f63e-4e4b-9119-9b385751f1b6?diagramType=Topology>



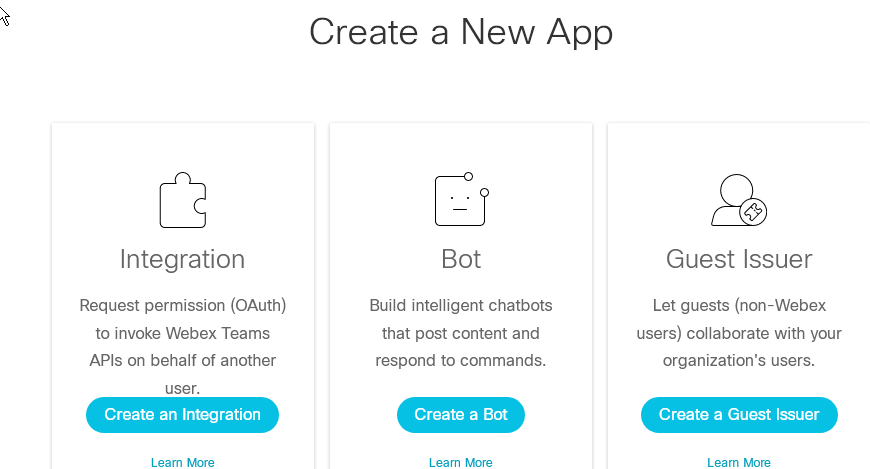


STARTING THE LAB:

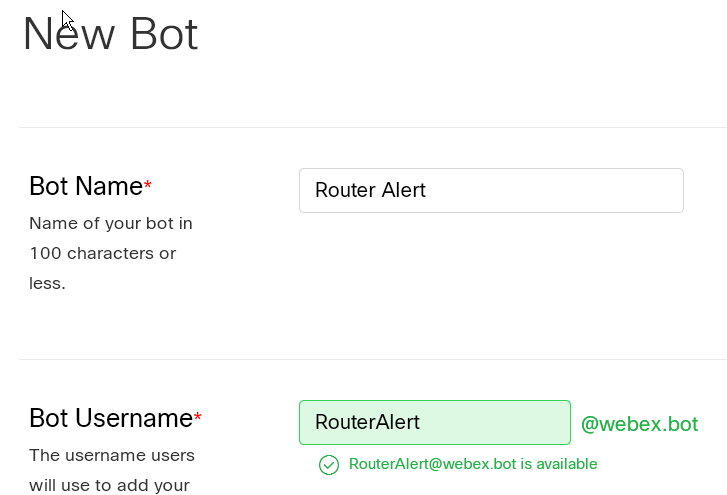
1. Reserve a lab in the DevNet developers sandbox.
2. Create a Webex Teams Bot by going to the developer.Webex.com portal and “Creat a new App”

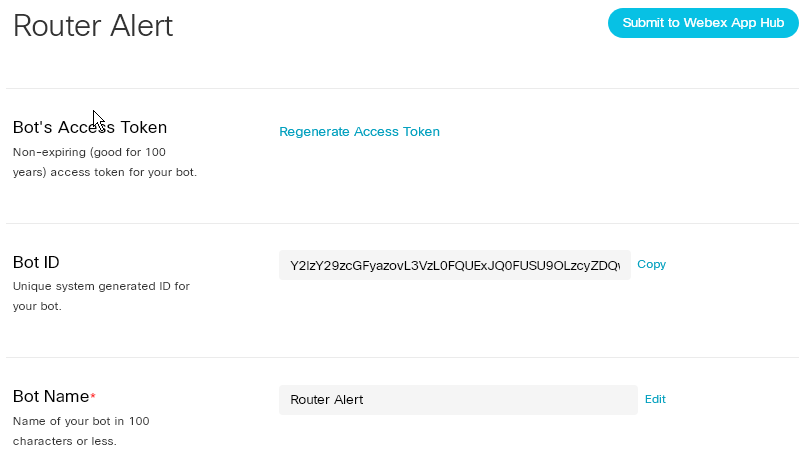


3) “Now select Creat a Bot”



1. Fill out the Bot name, Bot username, select the Icon, and create a description for the Bot





1. Then copy the Bot’s Access Token , and Bot ID that gets generated. Save it for later use.
2. Now log into the csr1000v router and enable IOx

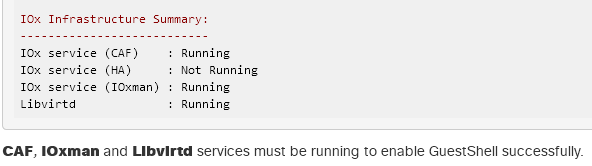
csr1000v#conf t

Enter configuration commands, one per line. End with CNTL/Z.

csr1000v(config)#iox

csr1000v(config)#end

1. Now do a “show iox-service” and make sure the IOx service is “running”



1. Now we need to create a way for the IOS-XE container l to talk to the host router. Create a new interface called **VirtualPortGroup**, which will be in the same broadcast domain as the Guest Shell container.  
   To configure it, use the following commands from IOS XE privileged EXEC mode:

csr1000v# conf t

interface VirtualPortGroup0

ip address 192.168.1.1 255.255.255.0

no shut

exit

1. Now we have to put an IP address on the container with the following command

csr1000v#conf t

app-hosting appid guestshell

vnic gateway1 virtualportgroup 0 guest-interface 0 guest-ipaddress 192.168.1.2 netmask 255.255.255.0 gateway 192.168.1.1 name-server 8.8.8.8

end

1. Now we need to configure NAT on the VirtualPortGroup0 and the GigabitEthernet1 interfaces

conf t

interface VirtualPortGroup0

ip nat inside

!

interface GigabitEthernet1

ip nat outside

!

ip access-list extended NAT-ACL

permit ip 192.168.1.0 0.0.0.255 any

!

ip nat inside source list NAT-ACL interface GigabitEthernet1 overload

end

1. Now we need to enable the Guest Shell, use the following command from the privileged EXEC mode:

csr1000v#guestshell enable

Interface will be selected if configured in app-hosting

Please wait for completion

guestshell installed successfully

Current state is: DEPLOYED

guestshell activated successfully

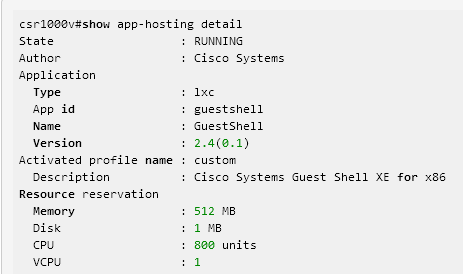
Current state is: ACTIVATED

guestshell started successfully

Current state is: RUNNING

Guestshell enabled successful

1. Now let’s verify the status of the Guest Shell, use the following command:

csr1000v#show app-hosting detail 

State should be “RUNNING”

1. After enabling the Guest Shell, it can be accessed using guestshell command from the privileged EXEC mode:

csr1000v# guestshell

[guestshell@guestshell ~]$

1. Now we need to update repositories, libraries and needed tools. Make sure you type "yes / y" when asked and allow the installation of needed packages. Be patient as it may take a while.

[guestshell@guestshell ~]$ sudo yum update

[...]

Complete!

[guestshell@guestshell ~]$ sudo pip install --upgrade pip

[...]

Successfully installed pip-10.0.0

[guestshell@guestshell ~]$ sudo pip install -U setuptools

[...]

Successfully installed setuptools-39.0.1

1. Now lets install the Webex Teams API, aka ciscosparkapi

[guestshell@guestshell ~]$ sudo -E pip install ciscosparkapi

[...]

Successfully installed certifi-2018.1.18 chardet-3.0.4 ciscosparkapi-0.10 future-0.16.0 idna-2.6 requests-2.18.4 requests-toolbelt-0.8.0 urllib3-1.22

1. Now lets add a python script to the guestshell: We need to create a .py file with the code below in order to be executed by the guestshell and reach to Webex Teams. In this example we will create "teamsbotscript.py"

1) COPY the script text below: select the text, right click copy

2) Create new file, for example teamsbotscript.py Let's use here for example the vi editor:

[guestshell@guestshell ~]$ sudo vi teamsbotscript.py

3) Once you enter vi, press "i" to get into insert mode, right click into terminal, click paste.

4) Press key ESC and type :wq! then press ENTER to save all the document.

If you need some VI command help click here: <https://www.cs.colostate.edu/helpdocs/vi.html>

1. Copy and paste. Note the first 2 lines are not indented.

1. from ciscosparkapi import CiscoSparkAPI

2.

3. if \_\_name\_\_=='\_\_main\_\_':

4. # Use ArgParse to retrieve command line parameters.

5. from argparse import ArgumentParser

6. parser = ArgumentParser("Spark Check In")

7. # Retrieve the Spark Token and Destination Email

8. parser.add\_argument(

9. "-t", "--token", help="Spark Authentication Token", required=True

10. )

11. # Retrieve the Spark Token and Destination Email

12. parser.add\_argument(

13. "-e", "--email", help="Email to Send to", required=True

14. )

15. args = parser.parse\_args()

16. token = args.token

17. email = args.email

18. message = "\*\*Alert:\*\* Router in Config Mode"

19. api = CiscoSparkAPI(access\_token=token)

20. api.messages.create(toPersonEmail=email, markdown=message)

1. Now lets check to see if the script was created and saved:

[guestshell@guestshell ~]$ ls

teamsbotscript.py

[guestshell@guestshell ~]$ cat teamsbotscript.py

from ciscosparkapi import CiscoSparkAPI

if \_\_name\_\_=='\_\_main\_\_':

# Use ArgParse to retrieve command line parameters.

from argparse import ArgumentParser

parser = ArgumentParser("Spark Check In")

# Retrieve the Spark Token and Destination Email

parser.add\_argument(

"-t", "--token", help="Spark Authentication Token", required=True

)

# Retrieve the Spark Token and Destination Email

parser.add\_argument(

"-e", "--email", help="Email to Send to", required=True

)

args = parser.parse\_args()

token = args.token

email = args.email

message = "\*\*Alert:\*\* Router in Config Mode"

api = CiscoSparkAPI(access\_token=token)

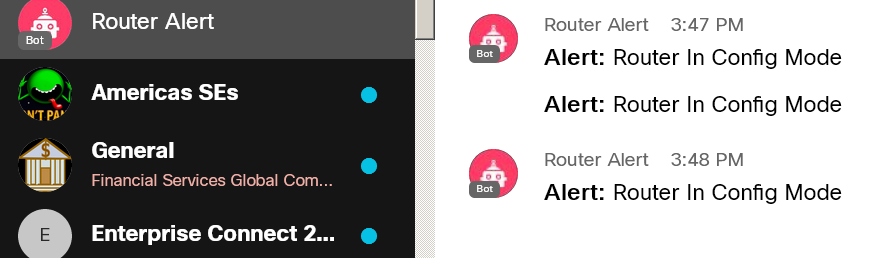
api.messages.create(toPersonEmail=email, markdown=message)

1. Now that the script is in the Guest Shell, it is time to run it and verify that is working. Open the Webex Teams application on your PC and log into Teams. Run the following command and you should see a notification or message if you have the BOT open in Webex Teams.

sudo python teamsbotscript.py -t "Bot's access token from step 5." -e "your email where Teams is enabled"

For example:

[guestshell@guestshell ~]$ sudo python teamsbotscript.py -t YWFhMmItY2Y2OS00YTM2LWJiMDEtOGIwZmE2ZTIyYTIwZGIwNDNhMDUtODAy\_PF84\_1eb65fdf-9643-417f-9974-ad72cae0e10f -e [rmihalcin@cisco.com](mailto:rmihalcin@cisco.com)



1. Now we need to add the EEM script to monitor the Syslog messages from the router to trigger the “teamsbotscript.py”

[guestshell@guestshell ~]$ exit

csr1000v#config t

csr1000v(config)#event manager applet testapplet

csr1000v(config-applet)#event syslog pattern "%SYS-5-CONFIG\_I: Configured from”

csr1000v(config-applet)#action 0.0 cli command "en"

csr1000v(config-applet)#action 1.0 cli command "guestshell run python teamsbotscript.py -t YWFhMmItY2Y2OS00YTM2LWJiMDEtOGIwZmE2ZTIyYTIwZGIwNDNhMDUtODAy\_PF84\_1eb65fdf-9643-417f-9974-ad72cae0e10f -e [rmihalcin@cisco.com](mailto:rmihalcin@cisco.com)

csr1000v(config-applet)#exit

1. Now when the command “conf t” is typed in the router, the syslog message gets detected by the EEM that triggers the python script in the Guest Shell. We then get a Bot notification over Webex Teams.
2. There are many use cases, such as watching interface counters, or watching when a port changes state on a switch. Here are some links for some EEM scripts:

<https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/eem/configuration/xe-16/eem-xe-16-book/eem-overview.html>

<https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/eem/command/eem-cr-book/eem-cr-e2.html>

REFERENCE LINKS:

<https://devnetsandbox.cisco.com/RM/Diagram/Index/18115a77-67d8-47f7-a0e0-b829ea75ff82?diagramType=Topology>

<https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/eem/configuration/xe-16/eem-xe-16-book/eem-overview.html>

<https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/eem/command/eem-cr-book/eem-cr-e2.html>

<https://community.cisco.com/t5/devnet-sandbox-documents/poc-catalyst-9000-sandbox-integration-with-spark/ta-p/3648713>

[https://developer.cisco.com/docs/ios-xe/#](https://developer.cisco.com/docs/ios-xe/)

<https://developer.cisco.com/site/ydk/>

<https://devnetsandbox.cisco.com/RM/Diagram/Index/18115a77-67d8-47f7-a0e0-b829ea75ff82?diagramType=Topology>

<https://github.com/CiscoDevNet/python_code_samples_network>