Collect IOS-XE Routing Protocol Flapping Logs with Python

Contents

Introduction

This document describes how to configure Python Scripts to collect OSPF, EIGRP and IS-IS logs when the protocols flap.

Prerequisites

Requirements

Cisco recommends that you are familiar with the topics listed:

- App-hosting configuration
- OSPF
- EIGRP
- IS-IS
- vi editor

Components Used

The information in this document is based on Cisco IOS XE software version 17.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Note: This document does not dive into the apphosting details. Further information can be found in the referenced links.

Configure

Configurations

When you open a TAC case, it is very important to collect relevant information to save time. Sometimes, the clue for a failure is within some basic outputs that you can gather from the device. In this document, you have examples of how to leverage Python Scripts to get this data. Three protocols are considered, OSPF, EIGRP and IS-IS.

Step 1. The first thing you need to do is to configure and enable guestshell.

```
Router(config)#iox
Router(config)#interface VirtualPortGroup 0
Router(config-if)#ip address 192.0.2.1 255.255.255.252
Router(config-if)#exit
Router(config)#
Router(config)#app-hosting appid guestshell
Router(config-app-hosting)#app-vnic gateway0 virtualportgroup 0 guest-interface 0
Router(config-app-hosting-gateway0)#guest-ipaddress 192.0.2.2 netmask 255.255.255.252
Router(config-app-hosting)#app-default-gateway 192.0.2.1 guest-interface 0
Router(config)#end
```

In this configuration, there are three important steps:

1. Enable IOX service. This is required to enable guestshell.

2. Configure the VirtualPortGroup that acts as the default gateway for the guestshell default gateway.

3. Configure app-hosting for the guestshell. You can tell from the configurations where the VirtualPortGroup comes into play.

Step 2. Next, you need to enable guestshell from privilege mode.

```
Router#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 successfully

Router#
*Jun 15 21:31:31.499: %IM-6-IOX_INST_INFO: R0/0: ioxman: IOX SERVICE guestshell LOG: Guestshell is up a
```

If everything is properly configured, you have to see the log in the preceding example.

Step 3. Now, you are ready to configure the python scripts. Run the command **guestshell** in privilege mode. You see the prompt as in the subsequent example:

```
Router#guestshell
[guestshell@guestshell ~]$
```

Step 4. Create a file with vi editor and configure the scripts based on the protocols you have enabled.

```
[guestshell@guestshell ~]$ vi ospf.py
```

This window shows up

```
~
~
~
~
~
~
~
"ospf.py" 0L, 0C
```

Step 5**. Press** "i" in order to insert text. **Paste** the script then **press** "esc", then **enter** the characters **:wq**

```
~
from cli import cli
from time import sleep

cli("enable")
cli("debug ip ospf hello")
cli("debug ip ospf adj")
cli("show ip ospf interface | append bootflash:Router-ospf-logs.txt")
cli("show ip ospf neighbor | append bootflash:Router-ospf-logs.txt")
cli("show interfaces | append bootflash:Router-ospf-logs.txt")
cli("show logging | append bootflash:Router-ospf-logs.txt")
```

```
cli("show tech | append bootflash:Router-showtech.txt")
sleep(30)
cli("undebug all")
~
~
~
~
"ospf.py" [New] 14L, 458C written
[guestshell@guestshell ~]$
```

Exit the guestshell with **exit** command.

Verify

Test the script. Exit from the guestshell with **exit** command. Then run **guestshell run python3 ospf.py**

```
F340.20.09-8500-1#guestshell run python3 ospf.py
```

Here are the scripts for all three protocols; OSPF, EIGRP and IS-IS.

OSPF

```
from cli import cli
from time import sleep

cli("enable")
cli("debug ip ospf hello")
cli("debug ip ospf adj")
cli("show ip ospf interface | append bootflash:Router-ospf-logs.txt")
cli("show ip ospf neighbor | append bootflash:Router-ospf-logs.txt")
cli("show interfaces | append bootflash:Router-ospf-logs.txt")
cli("show logging | append bootflash:Router-ospf-logs.txt")
cli("show tech | append bootflash:Router-showtech.txt")
sleep(30)
cli("undebug all")
```

EIGRP

```
from cli import cli
from time import sleep

cli("enable")
cli("debug eigrp packet")
cli("show ip eigrp neighbor | append bootflash:Router-eigrp-logs.txt")
cli("show ip eigrp interface | append bootflash:Router-eigrp-logs.txt")
cli("show interfaces | append bootflash:Router-eigrp-logs.txt")
```

```
cli("show logging | append bootflash:Router-eigrp-logs.txt")
cli("show tech | append bootflash:Router-showtech.txt")
sleep(30)
cli("undebug all")
```

IS-IS

```
from cli import cli
from time import sleep

cli("enable")
cli("debug isis adj-packet")
cli("show isis neighbor detail | append bootflash:Router-isis-logs.txt")
cli("show clns neighbor detail | append bootflash:Router-isis-logs.txt")
cli("show clns interface | append bootflash:Router-isis-logs.txt")
cli("show interfaces | append bootflash:Router-isis-logs.txt")
cli("show logging | append bootflash:Router-isis-logs.txt")
cli("show tech | append bootflash:Router-showtech.txt")
sleep(30)
cli("undebug all")
```

You can automate log collection with EEM scripts that run the Python scripts after syslog patterns are observed. In the next section, you have the EEM scripts that you can configure along with the python scripts to accomplish this task.

OSPF

```
event manager applet ospf-flap authorization bypass
event syslog pattern "%OSPF-5-ADJCHG:.*from FULL to DOWN" maxrun 120 ratelimit 120
action 010 cli command "enable"
action 020 cli command "guestshell run python3 ospf.py"
action 030 exit
```

EIGRP

```
event manager applet eirgp-flap authorization bypass
event syslog pattern "%DUAL-5-NBRCHANGE: EIGRP.*Neighbor.*is down" maxrun 120 ratelimit 120
action 010 cli command "enable"
action 020 cli command "guestshell run python3 eigrp.py"
action 030 exit
```

IS-IS

```
event manager applet isis-flap authorization bypass
```

```
event syslog pattern "%CLNS-5-ADJCHANGE: ISIS: Adjacency to.*Down" maxrun 120 ratelimit 120
action 010 cli command "enable"
action 020 cli command "guestshell run python3 isis.py"
action 030 exit
```

Note: The commands collected in these script provide basic initial information. When you open a TAC case, more information can be requested by TAC engineers to investigate further if needed.

Reference links

- Guestshell
- Python API