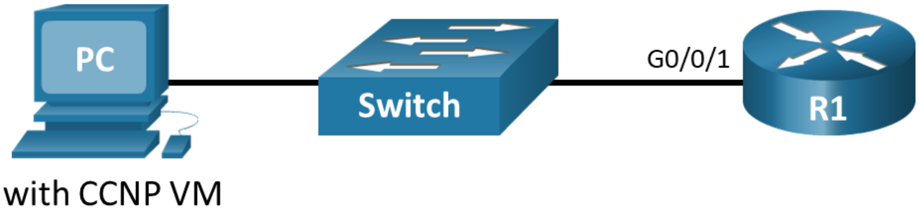
Lab - Construct an EEM Applet

Topology



Addressing Table

Device	Interface	IP Address	Subnet Mask
R1	G0/0/1	192.168.1.1	255.255.255.0
	Lo1	192.168.10.1	255.255.255.0
	Lo2	192.168.20.1	255.255.255.0
PC	NIC	DHCP	DHCP

Objectives

In this lab, you will construct EEM applet to automate configuration and troubleshooting of a network.

Part 1: Build the Network and Verify Connectivity

Part 2: Implement a Syslog Detector EEM Applet

Part 3: Implement a CLI Detector EEM Applet

Background / Scenario

Cisco IOS Embedded Event Manager (EEM) is a subsystem of Cisco IOS that enables you to create EEM applets.

EEM applets consist of statements that detect an event that occurs on an IOS device or connected network, and actions to be executed when the event is detected. A number of event detectors are available in IOS, although this may vary between IOS versions. EEM actions include executing IOS commands, generating an SNMP trap, executing other EEM policies, or sending an email notification to an address that has been specified in the policy. In addition, there are a number of other detectors and actions that have not been described here.

In Part 1 of this lab, you will create an EEM applet that will detect when the status of an interface becomes line protocol down. When the line protocol of the interface goes down, that interface is assumed to have a problem. The policy shuts down the interface and activates a backup interface.

In Part 2 of this lab, you will create an EEM applet that detects when the command to save the running configuration to the startup configuration has been issued. Prior to the configuration being saved, the current startup configuration is sent to a TFTP server as a backup of the last configuration.

Required Resources

- 1 Router with IOS version 12.2 or higher.
- 1 Switch (Optional: any switch available for connecting R1 and the PC)
- 1 PC (Choice of operating system with Cisco Networking Academy CCNP VM running in a virtual machine client and terminal emulation program)
- Ethernet cables as shown in the topology

Instructions

Part 1: Build the Network and Verify Connectivity

In Part 1, you will cable the devices, start the CCNP VM, and configure R1. You will then verify connectivity between the CCNP VM and R1 and test an SSH connection to R1.

Step 1: Cable the network as shown in the topology.

Attach the devices as shown in the topology diagram, and cable as necessary.

Step 2: Start the CCNP VM.

Note: If you have not completed **Lab - Install the CCNP Virtual Machine**, do so now before continuing with this lab.

a. Open VirtualBox. Start the **CCNP VM** virtual machine.

b. Enter the password **StudentPass** to log into the VM as necessary.

Step 3: Configure R1.

Connect to the CLI of R1 and paste in the following configuration for basic router settings.

```
hostname R1
no ip domain lookup
line con 0
 exec-t 0 0
 logg sync
line vty 0 15
 exec-t 0 0
 logg sync
 login local
 transport input ssh
ip domain name example.netacad.com
crypto key generate rsa modulus 2048
username cisco priv 15 password cisco123!
interface GigabitEthernet0/0/1
 description Link to PC
 ip address 192.168.1.1 255.255.255.0
 no shutdown
interface Loopback1
 description Primary Interface
 ip address 192.168.10.1 255.255.255.0
 no shutdown
interface Loopback2
 description Backup Interface
 ip address 192.168.20.1 255.255.255.0
 shutdown
ip dhcp excluded-address 192.168.1.1 192.168.1.10
ip dhcp pool LAN
 network 192.168.1.0 /24
 default-router 192.168.1.1
 domain-name example.netacad.com
end
```

Step 4: Verify connectivity from the PC to R1

a. From the PC, ping R1 to verify that the PC has received an IP address from the DHCP server.

b. Connect to R1 by using an SSH client. Use the **cisco** username and **cisco123!** password.

Note: This lab assumes that configuration will occur over the network to the vty lines of R1. The configuration script configures SSH for transport and local login on the vty lines. Other means of access, such as a direct console connection, are also possible.

Part 2: Implement a Syslog Detector EEM Applet

In Part 2, you will create an EEM applet that detects a specific syslog status message that has been issued by a router. The syslog message is notification of a change in the status of an interface. When this syslog message is detected, a series of IOS commands will be executed and an email will be sent to an account that is monitored by network administrators.

Scenario: It has been found that an interface occasionally becomes unstable and the line protocol goes down for unknown reasons. Staff are investigating the problem to determine the root cause. Meanwhile, the decision has been made to implement an EEM applet, which will detect the syslog message that is issued when the line protocol goes down on the interface. When this event is detected, a series of IOS commands will be executed that shut down the interface and activate a backup interface. For this lab, loopbacks will be used to simulate the faulty primary interface and backup interface.

Step 1: Register the EEM applet.

To register an EEM applet, use the **event manager applet** command followed by the applet name. We will name the applet **BACKUP_INTERFACE**. This will begin the process of registering the applet and moving to the configuration mode for the applet. However, the applet is not fully registered until a triggering event has been specified.

```
R1(config)# event manager applet BACKUP_INTERFACE
```

Step 2: Configure the event that will trigger the applet.

Configure the syslog event detector with the **event syslog** command followed by the **pattern** keyword. This keyword indicates that a text pattern should be matched in the syslog event in order for the applet to run. Regular expressions can be used to provide flexibility regarding detection of the message to be matched. In this case, we will provide the pattern verbatim. The message should be enclosed in quotes because it includes spaces.

```
R1(config-applet)# event syslog pattern "Line protocol on Interface Loopback1,
changed state to down"
```

Step 3: Configure the action commands that will occur if the event is triggered.

The applet will execute a series of actions that consist of IOS commands and a special command that will send an email notification to network staff. The actions use the **action** command, followed by a tag, which serves as a sequence number, and the type of action to be taken. Finally, the full IOS command that will be executed is specified in quotes. The series of actions begin from User EXEC mode. Therefore, include the commands that move between required modes.

The following actions specify commands to shut down the faulty interface and activate a backup interface. In the next step, you will configure the applet to send an email notification.

```
R1(config-applet)# action 1.0 cli command "enable"
R1(config-applet)# action 1.5 cli command "configure terminal"
R1(config-applet)# action 2.0 cli command "interface loopback1"
R1(config-applet)# action 2.5 cli command "shutdown"
R1(config-applet)# action 3.0 cli command "interface loopback2"
R1(config-applet)# action 3.5 cli command "no shutdown"
R1(config-applet)# action 4.0 cli command "end"
```

Step 4: Enter the email action.

The final action sends an email from R1. The command is **action** *tag* **mail server** followed by parameters such as the name of the email server, the **to:** and **from:** addresses, the email subject, and the text to be sent in the body of the message. The command ends with the built-in variable **$_cli_result** which contains the output from the last CLI command that was executed. Because the router does not have access to the internet, we will use example email information in the command.

```
R1(config-applet)# action 5.0 mail server "mailserver.example.com" to
"net_admin@example.com" from "R1_EEM@example.com" subject "Primary interface
down." body "Backup interface active $_cli_result"
```

Step 5: Verify the applet configuration.

Use the **show event manager policy registered** command to verify the EEM policies that are registered on R1.

```
R1(config-applet)# end
R1# show event manager policy registered
No.  Class     Type    Event Type          Trap  Time Registered           Name
1    applet    user    syslog              Off   Fri Feb 21 21:45:31 2020
backup_interface
 pattern {Line protocol on Interface Loopback1, changed state to down}
 maxrun 20.000
 action 1.0 cli command "enable"
 action 1.5 cli command "config t"
 action 2.0 cli command "interface lo1"
 action 2.5 cli command "shutdown"
 action 3.0 cli command "interface lo2"
 action 3.5 cli command "no shutdown"
 action 4.0 cli command "end"
 action 5.0 mail server "mailserver@example.net" to "net_admin@example.com" from
"R1_EEM@example.com" subject "Primary interface down" body "Backup interface active
$_cli_result"
```

Step 6: Test the applet to verify that it works.

To test this applet, you must cause the router to generate the syslog message that is configured as the event trigger.

a. Verify the status of the interfaces with the **show ip interface brief** command to ensure that Loopback1 is up and Loopback2 is administratively down.

```
R1# show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet0/0/0   unassigned      YES unset  administratively down down
GigabitEthernet0/0/1   192.168.1.1     YES manual up                    up
Serial0/1/0            unassigned      YES unset  administratively down down
Serial0/1/1            unassigned      YES unset  administratively down down
GigabitEthernet0       unassigned      YES unset  administratively down down
Loopback1              192.168.10.1    YES manual up                    up
Loopback2              192.168.20.1    YES manual administratively down down
```

b. Activate debugging of the EEM CLI action with the **debug event manager action cli** command.

```
R1# debug event manager action cli
```

c. Use the **send log** command to simulate a syslog message. To trigger the EEM applet, the message must exactly match the message that was configured as the event trigger. This will simulate the event trigger. The debug command will display messages related to the execution of the applet.

```
R1# send log "Line protocol on Interface Loopback1, changed state to down"
R1#
*Feb 21 22:00:46.194: %SYS-7-USERLOG_DEBUG: Message from tty0(user id: ):
"Lineprotocol on Interface Loopback1, changed state to down"
*Feb 21 22:00:46.197: %HA_EM-6-LOG: BACKUP_INTERFACE : DEBUG(cli_lib) : : CTL
:cli_open called.
*Feb 21 22:00:46.197: %HA_EM-6-LOG: BACKUP_INTERFACE : DEBUG(cli_lib) : : OUT :R1>
*Feb 21 22:00:46.197: %HA_EM-6-LOG: BACKUP_INTERFACE : DEBUG(cli_lib) : : IN
:R1>enable
*Feb 21 22:00:46.208: %HA_EM-6-LOG: BACKUP_INTERFACE : DEBUG(cli_lib) : : OUT :R1#
*Feb 21 22:00:46.208: %HA_EM-6-LOG: BACKUP_INTERFACE : DEBUG(cli_lib) : : IN
:R1#config t
*Feb 21 22:00:46.320: %HA_EM-6-LOG: BACKUP_INTERFACE : DEBUG(cli_lib) : : OUT :Enter
configuration commands, one per line.  End with CNTL/Z.
*Feb 21 22:00:46.320: %HA_EM-6-LOG: BACKUP_INTERFACE : DEBUG(cli_lib) : : OUT
:R1(config)#
*Feb 21 22:00:46.320: %HA_EM-6-LOG: BACKUP_INTERFACE : DEBUG(cli_lib) : : IN
:R1(config)#interface lo1
*Feb 21 22:00:46.431: %HA_EM-6-LOG: BACKUP_INTERFACE : DEBUG(cli_lib) : : OUT
:R1(config-if)#
*Feb 21 22:00:46.431: %HA_EM-6-LOG: BACKUP_INTERFACE : DEBUG(cli_lib) : : IN
:R1(config-if)#shutdown
*Feb 21 22:00:46.542: %HA_EM-6-LOG: BACKUP_INTERFACE : DEBUG(cli_lib) : : OUT
:R1(config-if)#
*Feb 21 22:00:46.543: %HA_EM-6-LOG: BACKUP_INTERFACE : DEBUG(cli_lib) : : IN
:R1(config-if)#interface lo2
*Feb 21 22:00:46.654: %HA_EM-6-LOG: BACKUP_INTERFACE : DEBUG(cli_lib) : : OUT
:R1(config-if)#
*Feb 21 22:00:46.654: %HA_EM-6-LOG: BACKUP_INTERFACE : DEBUG(cli_lib) : : IN
:R1(config-if)#no shutdown
```

```
*Feb 21 22:00:46.766: %HA_EM-6-LOG: BACKUP_INTERFACE : DEBUG(cli_lib) : : OUT
:R1(config-if)#
*Feb 21 22:00:46.766: %HA_EM-6-LOG: BACKUP_INTERFACE : DEBUG(cli_lib) : : IN
:R1(config-if)#end
*Feb 21 22:00:46.778: %SYS-5-CONFIG_I: Configured from console by  on vty0
(EEM:BACKUP_INTERFACE)
<output omitted>
```

Note: After about 30 seconds, you will receive an SMTP error message similar to the one below. This will verify that R1 attempted to send an email message. However, recall that the configuration was just an example. R1 requires internet access or access to an email server to send an email message.

```
*Feb 21 22:01:17.594: %HA_EM-3-FMPD_SMTP: Error occurred when sending mail to SMTP
server: mailserver.example.com : timeout error
```

d. Issue the **undebug all** command to stop debugging the EEM applet activity.

e. Use the **show ip interface brief** command to verify that interface Loopback1 is administratively down and the backup interface Loopback2 is up.

```
R1# show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet0/0/0   unassigned      YES unset  administratively down down
GigabitEthernet0/0/1   192.168.1.1     YES manual up                    up
Serial0/1/0            unassigned      YES unset  administratively down down
Serial0/1/1            unassigned      YES unset  administratively down down
GigabitEthernet0       unassigned      YES unset  administratively down down
Loopback1              192.168.10.1    YES manual administratively down down
Loopback2              192.168.20.1    YES manual up                    up
```

Part 3: Implement a CLI Detector EEM Applet

In Part 3, you will construct an EEM applet that uses the CLI event detector. The CLI detector is triggered when a specified command is entered at the CLI. Like the syslog event detector, a regular expression pattern can be specified to match the desired CLI command. In this applet, you detect the **copy running-config startup-config** command. Before the command is allowed to execute, the current startup configuration will be backed up to flash memory.

Step 1: Configure the EEM environment variables.

Environment variables can simplify creation of EEM applets by allowing variables to be used in action commands.

a. Determine and make note of the IP address of the NIC for the PC that is connected to the R1 network.

b. Configure the EEM environment variable for the path to which the backup configuration file will be saved. The first argument is the name of the variable, and the second argument is the value of the variable. Replace the expression in brackets, including the brackets, with the IP address of your PC.

```
R1(config)# event manager environment TFTP_SERVER tftp://[your PC IP
address]/
```

c. Configure the EEM environment variable for the name of the configuration file that will be sent to the TFTP server. Name the file **R1_last_config.cfg**.

```
R1(config)# event manager environment filename R1_last_config.cfg
```

Step 2: Register the EEM applet.

Register the new applet. Name the applet **LAST_CONFIG**.

```
R1(config)# event manager applet LAST_CONFIG
```

Step 3: Configure the event that will trigger the applet.

Configure the CLI event trigger for the applet. The pattern in the command uses a regular expression to match the possible CLI shorthand variations of the **show running-config startup-config** command. In this regular expression, the "**.**" represents any character, and the "*****" represents zero or more instances of those characters. By using this simple regular expression, the many possible valid variations of the command that someone might enter will be detected. The **sync** parameter in the **event** command indicates that the policy should be executed synchronously before the CLI command executes. This allows the old configuration to be sent to the server before the new configuration is saved.

```
R1(config-applet)# event cli pattern "cop.* ru.* st.*" sync yes
```

Step 4: Configure the action commands that will happen if the event is triggered.

The following actions will send the startup configuration to the TFTP server. The **file prompt quiet** command suppresses prompts that require user verification. This is essential for automating some IOS commands.

```
R1(config-applet)# action 1.0 cli command "enable"
R1(config-applet)# action 2.0 cli command "configure terminal"
R1(config-applet)# action 3.0 cli command "file prompt quiet"
R1(config-applet)# action 4.0 cli command "end"
R1(config-applet)# action 5.0 cli command "copy start $TFTP_SERVER$filename"
R1(config-applet)# action 6.0 cli command "configure terminal"
R1(config-applet)# action 7.0 cli command "no file prompt quiet"
R1(config-applet)# action 8.0 syslog priority information msg "Configuration
file backed up to TFTP."
R1(config-applet)# end
```

Step 5: Verify the applet configuration.

Verify that the EEM applet is registered and that its contents are correct.

```
R1# show event manager policy registered
No.  Class     Type    Event Type          Trap  Time Registered           Name
1    applet    user    syslog              Off   Wed Feb 26 15:57:09 2020
BACKUP_INTERFACE
<output omitted>

2    applet    user    cli                 Off   Wed Feb 26 17:06:18 2020  LAST_CONFIG
 pattern {cop.* ru.* st.*} sync yes
 maxrun 20.000
 action 1.0 cli command "enable"
 action 2.0 cli command "configure terminal"
 action 3.0 cli command "file prompt quiet"
 action 4.0 cli command "end"
 action 5.0 cli command "copy start $TFTP_SERVER$filename"
 action 6.0 cli command "configure terminal"
 action 7.0 cli command "no file prompt quiet"
 action 8.0 syslog priority informational msg "Configuration file backed up to TFTP."
```

Step 6: Test the applet to verify that it works.

a. In the Python VM, list the contents of the TFTP server directory. The directory should be empty.

```
student@CCNP:/$ ls /tftpboot/
student@CCNP:/$
```

b. Return to R1 and activate debugging of the EEM CLI action with the **debug event manager action cli** command.

```
R1# debug event manager action cli
```

c. Copy the running-config to the startup-config. Any valid version of the command should trigger the EEM applet actions.

```
R1# copy run start
R1#
*Feb 24 23:17:02.931: %HA_EM-6-LOG: LAST_CONFIG : DEBUG(cli_lib) : : CTL : cli_open
called.
*Feb 24 23:17:02.931: %HA_EM-6-LOG: LAST_CONFIG : DEBUG(cli_lib) : : OUT : R1>
*Feb 24 23:17:02.931: %HA_EM-6-LOG: LAST_CONFIG : DEBUG(cli_lib) : : IN  : R1>enable
*Feb 24 23:17:03.043: %HA_EM-6-LOG: LAST_CONFIG : DEBUG(cli_lib) : : OUT : R1#
*Feb 24 23:17:03.043: %HA_EM-6-LOG: LAST_CONFIG : DEBUG(cli_lib) : : IN  :
R1#configure terminal
*Feb 24 23:17:03.154: %HA_EM-6-LOG: LAST_CONFIG : DEBUG(cli_lib) : : OUT : Enter
configuration commands, one per line.  End with CNTL/Z.
*Feb 24 23:17:03.154: %HA_EM-6-LOG: LAST_CONFIG : DEBUG(cli_lib) : : OUT : R1(config)#
*Feb 24 23:17:03.154: %HA_EM-6-LOG: LAST_CONFIG : DEBUG(cli_lib) : : IN  :
R1(config)#file prompt quiet
*Feb 24 23:17:03.264: %HA_EM-6-LOG: LAST_CONFIG : DEBUG(cli_lib) : : OUT : R1(config)#
*Feb 24 23:17:03.264: %HA_EM-6-LOG: LAST_CONFIG : DEBUG(cli_lib) : : IN  :
R1(config)#end
*Feb 24 23:17:03.270: %SYS-5-CONFIG_I: Configured from console by  on vty0
(EEM:LAST_CONFIG)
*Feb 24 23:17:03.275: %HA_EM-6-LOG: LAST_CONFIG : DEBUG(cli_lib) : : OUT : R1#
*Feb 24 23:17:03.275: %HA_EM-6-LOG: LAST_CONFIG : DEBUG(cli_lib) : : IN  : R1#copy
start tftp://192.168.1.13/R1_last_config.cfg
<output omitted>
```

d. Return to the Python VM. List the contents of the TFTP server directory again. You should see that the R1_last_config.cfg file is now present on the TFTP server.

```
student@CCNP:/$ ls /tftpboot/
R1_last_config.cfg
student@CCNP:/$
```

e. Use the **more** command to view the configuration stored on the TFTP server.

```
student@CCNP:/$ more /tftpboot/R1_last_config.cfg

!
! Last configuration change at 17:26:03 UTC Wed Feb 26 2020
!
version 16.9
service timestamps debug datetime msec
service timestamps log datetime msec
platform qfp utilization monitor load 80
no platform punt-keepalive disable-kernel-core
!
hostname R1
!
boot-start-marker
boot-end-marker
!
!
vrf definition Mgmt-intf
 !
 address-family ipv4
 exit-address-family
 !
 address-family ipv6
 exit-address-family
--More--(11%)
```

f. Issue the **undebug all** command to stop debugging the EEM applet activity.

Router Interface Summary Table

Router Model	Ethernet Interface #1	Ethernet Interface #2	Serial Interface #1	Serial Interface #2
1800	Fast Ethernet 0/0 (F0/0)	Fast Ethernet 0/1 (F0/1)	Serial 0/0/0 (S0/0/0)	Serial 0/0/1 (S0/0/1)
1900	Gigabit Ethernet 0/0 (G0/0)	Gigabit Ethernet 0/1 (G0/1)	Serial 0/0/0 (S0/0/0)	Serial 0/0/1 (S0/0/1)
2801	Fast Ethernet 0/0 (F0/0)	Fast Ethernet 0/1 (F0/1)	Serial 0/1/0 (S0/1/0)	Serial 0/1/1 (S0/1/1)
2811	Fast Ethernet 0/0 (F0/0)	Fast Ethernet 0/1 (F0/1)	Serial 0/0/0 (S0/0/0)	Serial 0/0/1 (S0/0/1)
2900	Gigabit Ethernet 0/0 (G0/0)	Gigabit Ethernet 0/1 (G0/1)	Serial 0/0/0 (S0/0/0)	Serial 0/0/1 (S0/0/1)
4221	Gigabit Ethernet 0/0/0 (G0/0/0)	Gigabit Ethernet 0/0/1 (G0/0/1)	Serial 0/1/0 (S0/1/0)	Serial 0/1/1 (S0/1/1)
4300	Gigabit Ethernet 0/0/0 (G0/0/0)	Gigabit Ethernet 0/0/1 (G0/0/1)	Serial 0/1/0 (S0/1/0)	Serial 0/1/1 (S0/1/1)

Note: To find out how the router is configured, look at the interfaces to identify the type of router and how many interfaces the router has. There is no way to effectively list all the combinations of configurations for each router class. This table includes identifiers for the possible combinations of Ethernet and Serial interfaces in the device. The table does not include any other type of interface, even though a specific router may contain one. An example of this might be an ISDN BRI interface. The string in parenthesis is the legal abbreviation that can be used in Cisco IOS commands to represent the interface.