The bridge to possible

Automation in data collection with advanced Embedded Event Manager

Yuanhua Luo
Technical Consulting Engineer
Xiaofei Sun
Technical Consulting Engineer

BRKATO-2105

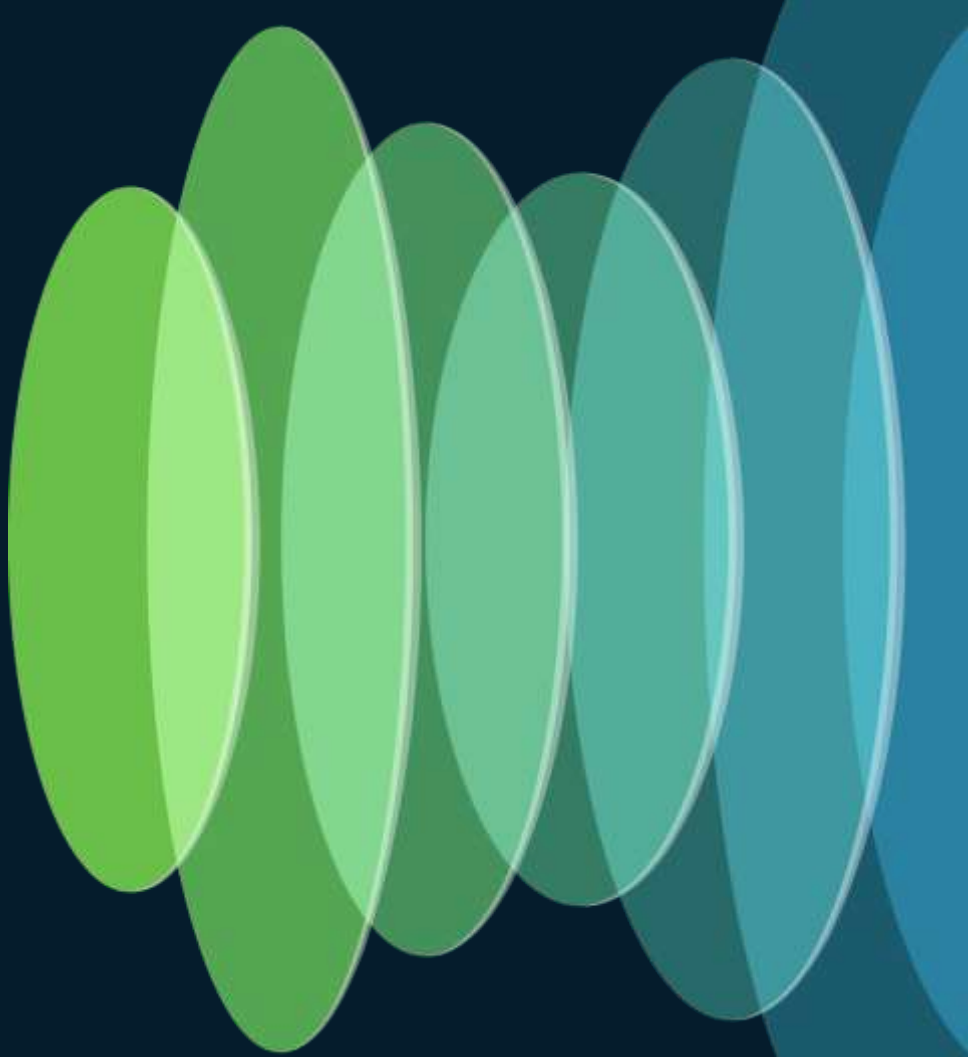
cisco Live!

#CiscoLive

Agenda

- Introduction
- Extended Log File Retention
- Trigger-Based Event Log Auto-Collection
- Integration with Python scripts
- Use Cases of running Python scripts
- References

Introduction



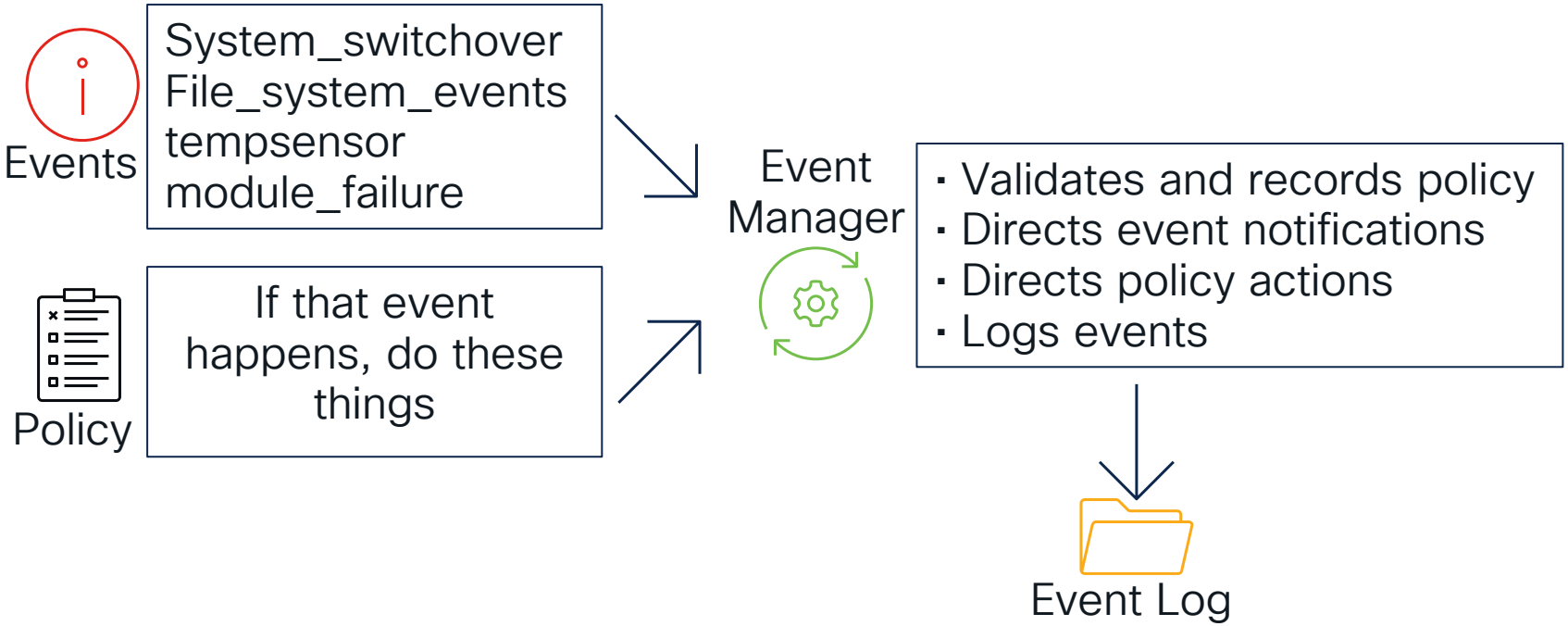
Real Life Problems

- A roll-over of the buffer overwrite incident logs.
- Investigation stalled due to delayed log collection.
- Log collection takes time and labor.

- Save more logs before roll-over.
- Log auto-collect on issues.

EEM Overview

Embedded Event Manager (EEM) monitors events and takes action to recover or troubleshoot these events.



EEM Policies

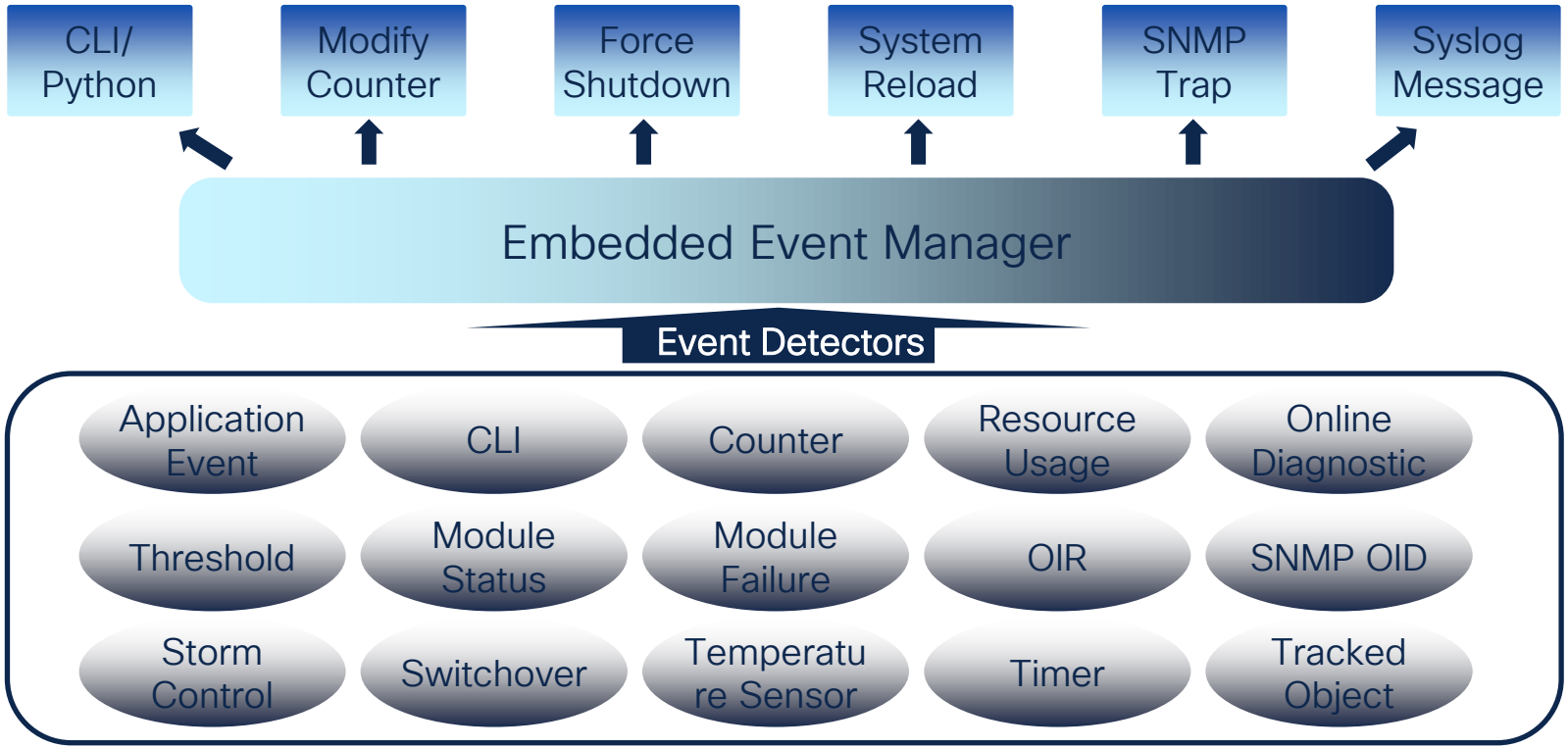
Predefined System Policies on Cisco Nexus 9000 switches

show event manager system-policy

__L2ACLRedirect	Do CallHome, log error and disable further HM testing on affected ports after 10 consecutive failures of GOLD "L2ACLRedirect" test
__PortLoopback	Do CallHome, log error and disable further HM testing on affected ports after 10 consecutive failures of GOLD "PortLoopback" test
__ethpm_link_flap	More than 30 link flaps in 420 seconds interval. Action: Error Disable the port
__ethpm_reinit_no_flap	More than 30 reinits in 420 seconds interval. Action: Error Disable the port
__syslog_trigger_default	Default policy for trigger based logging

EEM Policies

User-defined Policies



EEM Policies

User-defined Policies

Monitoring Module Powerdown

```
event manager applet monitorPoweroff
  description "Monitors module power down."
  event cli match "conf t; poweroff *"
  action 1.0 cli show module
```

EEM Overview

Integration with Python

EEM policies support Python scripts.

An EEM applet can include a Python script with an action command.

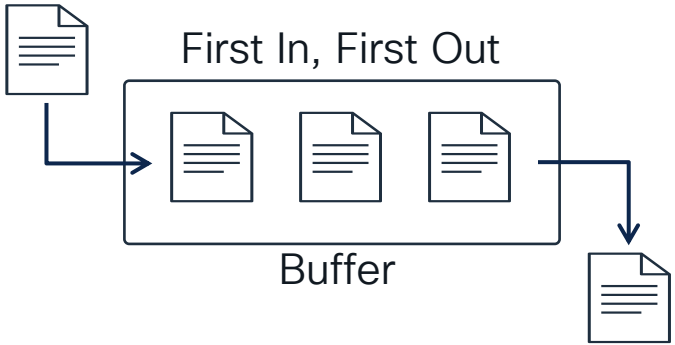
```
event manager applet a1
  event cli match "show clock"
  action 1 cli python bootflash:pydate.py
```

EEM Overview

Auto-Collection and Backup

Buffer Roll-over

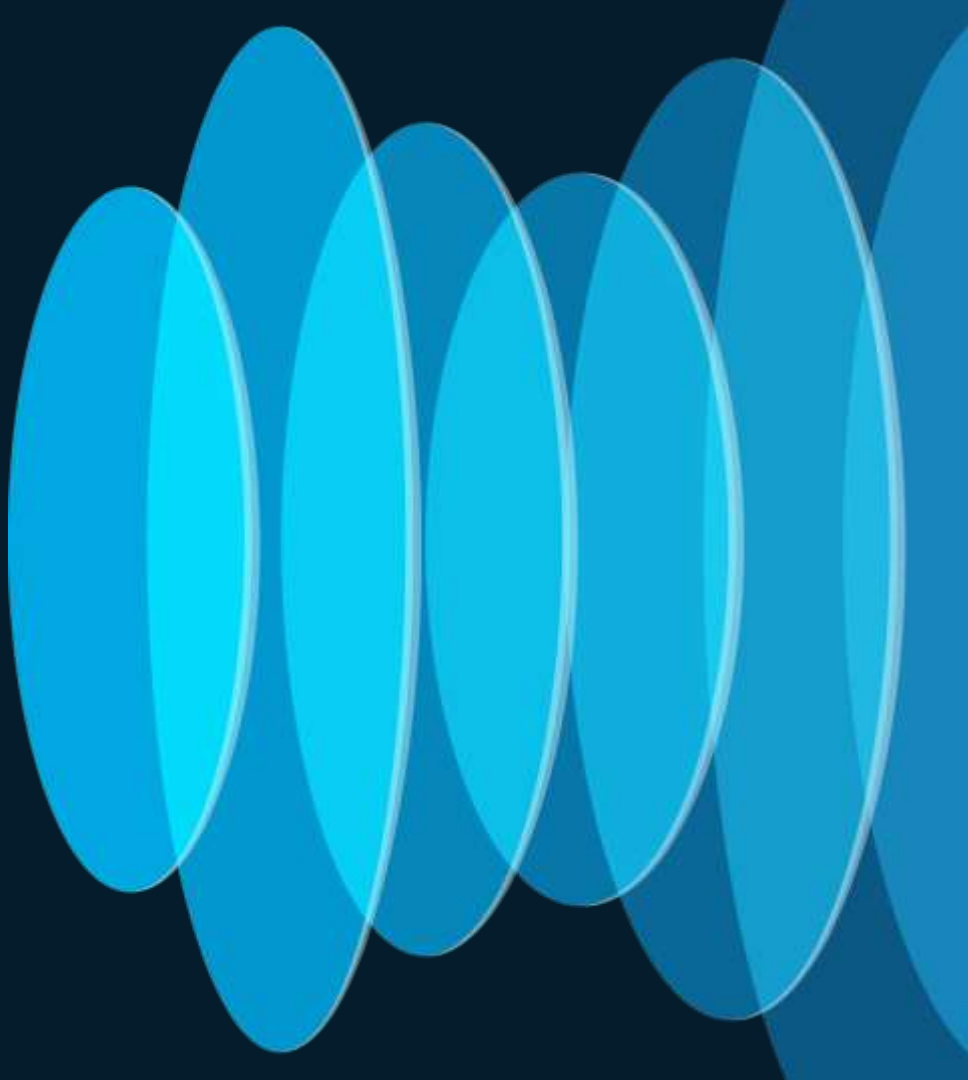
- Temporary buffer stores log files for a fixed amount of time
- The roll-over uses a first-in-first-out method



Methods of Auto-Collection and Backup:

- Extended Log File Retention
- Trigger-Based Event Log Auto-Collection

Extended Log File Retention



Extended Log File Retention

Overview

- Enabled by default for all services
- 15 minutes to several hours of event logs
- Depending on size, scale and component activity
- Can only enable specific services you need

Extended Log File Retention

Default setting

Extended Log File Retention is enabled by default for all services running on a switch.

```
N9K# show system internal bloggerd info log-dump-info
------------------------------------------------------------

Log Dump config is READY
Log Dump is ENABLED for ALL application services in the switch
Exceptions to the above rule (if any) are as follows:
------------------------------------------------------------
Module      | VDC         | SAP                          | Enabled?
------------------------------------------------------------
------------------------------------------------------------
Log Dump Srvlog config: source: default, mode: Normal, size: 80%
------------------------------------------------------------
------------------------------------------------------------
Log Dump Throttle Switch-Wide Config:
------------------------------------
Log Dump Throttle                                  : ENABLED
Minimum buffer rollover count (before throttling)  : 5
Maximum allowed rollover count per minute          : 5
------------------------------------------------------------
--More--
```

```
N9K# show system internal sysmgr service all | egrep -v "\[NA\]"
Name                 UUID       PID    SAP     state  Start count
-------------------- ---------- ------ ------- -------- -----------
aaa                  0x000000B5 14534  111     s0009              1
cert_enroll          0x0000012B 14533  169     s0009              1
ExceptionLog         0x00000050 14695  92      s0009              1
psshelper_gsvc       0x0000021A 14449  398     s0009              1
platform             0x00000018 14327  39      s0009              1
radius               0x000000B7 15136  113     s0009              1
securityd            0x0000002A 14532  55      s0009              1
acllog               0x0000023C 17949  425     s0009              1
aclmgr               0x00000182 14612  351     s0009              1
aclqos               0x0000016E 18031  190     s0009              1
adbm                 0x000004BA 17948  1250    s0009              1
adjmgr               0x00000108 14901  251     s0009              1
arp                  0x0000010C 15026  278     s0009              1
ascii-cfg            0x00000051 14531  76      s0009              1
bloggerd             0x00000474 14448  856     s0009              1
--More--
```

Extended Log File Retention

Log files and Statistics

- Event log files are stored in the debug:log-dump directory
- Statistics of the number of times a log being repeated

```
N9K# dir debug:log-dump/
    4864000     Apr 12 11:54:13 2024  2024041211485
6450713_evtlog_archive.tar
      51200     Apr 12 11:59:13 2024  2024041211541
3217955_evtlog_archive.tar
     204800     Apr 12 12:04:42 2024  2024041211591
3235267_evtlog_archive.tar
      51200     Apr 12 12:10:13 2024  2024041212044
2217457_evtlog_archive.tar
```

```
N9K# show system internal sysmgr service name l2fm
Service "l2fm" ("l2fm", 181):
        UUID = 0x19A, PID = 18042, SAP = 221
        State: SRV_STATE_HANDSHAKED (entered at time Fri
        Restart count: 1
        Time of last restart: Fri Apr 12 11:49:23 2024.
        The service never crashed since the last reboot.
        Tag = N/A
        Plugin ID: 1
```

```
N9K# show system internal sdwrap buffers sap 221 dict-stats detailed
Sap received is: 221

SDWrap Format Strings Dictionary stats for sap MTS_SAP_L2FM (221)

UUID: SRVUUID_LIBSDWRAP, Inst Type: 0
Message id  Frequency Message
----------  --------- ------------------------------
         4          1 System is not undergoing ISSU
        78          1 Vlan %d is part of reserved vlan bmp from sdb
       179          1 Vlan %d is not found in L2FM database. Skipping
       306          1 Vlan %d is removed from L2FM database and MTM d
       416          1 mts_drap_get_my_local_swid_only_msg failed with
       496          1 Lookup for backplane mac failed for vdc %d with
       598          1 L2FM - Slot %d SwCardId %d Port %d - %d Fp %d C
       687          1 Initialization done, handling control to fsm en
       753          2 Got l213_mode as %d
       823          1 L2FM - Slot %d Interface %s - %s [%08x-%08x] Fr
       931          1 L2FM - Slot %d Interface %s - %s [%08x-%08x] Br
```

Extended Log File Retention

Log files storage

- Log files are stored locally on flash RAM.
- 250MB memory is reserved for log file storage
- Log files are optimized in tar format

  (one file for every five minutes or 10MB, whichever occurs first)

Extended Log File Retention

Generating a Local Copy

bloggerd log-snapshot [*file-name*] [bootflash: *file-path* | logflash: *file-path* | usb1:] [size *file-size*] [time *minutes*]

```
N9K# bloggerd log-snapshot
Snapshot generated at logflash:evt_log_snapshot/20240413214801_snapshot_bundle.tar Please cleanup once done.
N9K#
N9K# run bash
bash-4.4$
bash-4.4$ tar -tvf /logflash/evt_log_snapshot/20240413214801_snapshot_bundle.tar
-rw-rw-rw- root/root    194560 2024-04-13 21:46 20240413214115771512_evtlog_archive.tar
-rw-rw-rw- root/root    133120 2024-04-13 21:41 20240413213613345263_evtlog_archive.tar
-rw-rw-rw- root/root    573440 2024-04-13 21:36 20240413213046547331_evtlog_archive.tar
-rw-rw-rw- root/root    256000 2024-04-13 21:30 20240413212546537170_evtlog_archive.tar
-rw-rw-rw- root/root    256000 2024-04-13 21:25 20240413212016528508_evtlog_archive.tar
-rw-rw-rw- root/root    419840 2024-04-13 21:20 20240413211511342327_evtlog_archive.tar
-rw-rw-rw- root/root    225280 2024-04-13 21:15 20240413210946504157_evtlog_archive.tar
-rw-rw-rw- root/root    358400 2024-04-13 21:09 20240413210446677059_evtlog_archive.tar
-rw-rw-rw- root/root    153600 2024-04-13 21:04 20240413205923083832_evtlog_archive.tar
-rw-rw-rw- root/root    389120 2024-04-13 20:59 20240413205416458499_evtlog_archive.tar
-rw-rw-rw- root/root    256000 2024-04-13 20:54 20240413204914429266_evtlog_archive.tar
-rw-rw-rw- root/root    112640 2024-04-13 20:49 20240413204346414816_evtlog_archive.tar
```

Extended Log File Retention

Parsing the Log tar Files

```
N9K# show system internal event-logs parse debug:log-dump/20240412195648961424_evtlog_archive.tar
--------FILE:/tmp/BLOGGERD0.5664771229546606/1-240412195106-240412195720-570340-Csensor-U422-M1-V1-IO-0-P18790
-messages.gz--------
2024-04-12T19:57:20.570331000+00:00 [M 1] [sensor] E_MTS_RX [18790] [REQ] Opc:MTS_OPC_PFM_GET_TEMPERATURE_INFO
(1009), Id:0X00019B59, Ret:SUCCESS(0), Src:0x00000101/39, Dst:0xFF000005/915, Flags:None, HA_SEQNO:0X00000000,
 RRtoken:0x00019B59, Sync:None, Payload Size: 4, Payload: 00 00 00
2024-04-12T19:57:05.569902000+00:00 [M 1] [sensor] E_MTS_RX [18790] [REQ] Opc:MTS_OPC_PFM_GET_TEMPERATURE_INFO
(1009), Id:0X0001791C, Ret:SUCCESS(0), Src:0x00000101/39, Dst:0xFF000005/915, Flags:None, HA_SEQNO:0X00000000,
 RRtoken:0x0001791C, Sync:None, Payload Size: 4, Payload: 00 00 00
2024-04-12T19:56:50.569342000+00:00 [M 1] [sensor] E_MTS_RX [18790] [REQ] Opc:MTS_OPC_PFM_GET_TEMPERATURE_INFO
(1009), Id:0X0001737F, Ret:SUCCESS(0), Src:0x00000101/39, Dst:0xFF000005/915, Flags:None, HA_SEQNO:0X00000000,
 RRtoken:0x0001737F, Sync:None, Payload Size: 4, Payload: 00 00 00
```

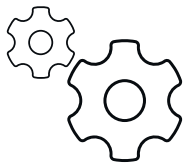
Additional keywords for parsing log tar file

component	Decode logs belonging to the component identified by process name
from-datetime	Decode logs from a specific date and time in yy[mm[dd[HH[MM[SS]]]]] format
instance	List of SDWRAP buffer instances to be decoded (comma separated)
module	Decode logs from modules such as SUP and LC (using module IDs)
to-datetime	Decode logs up to a specific date and time in yy[mm[dd[HH[MM[SS]]]]] format

Extended Log File Retention

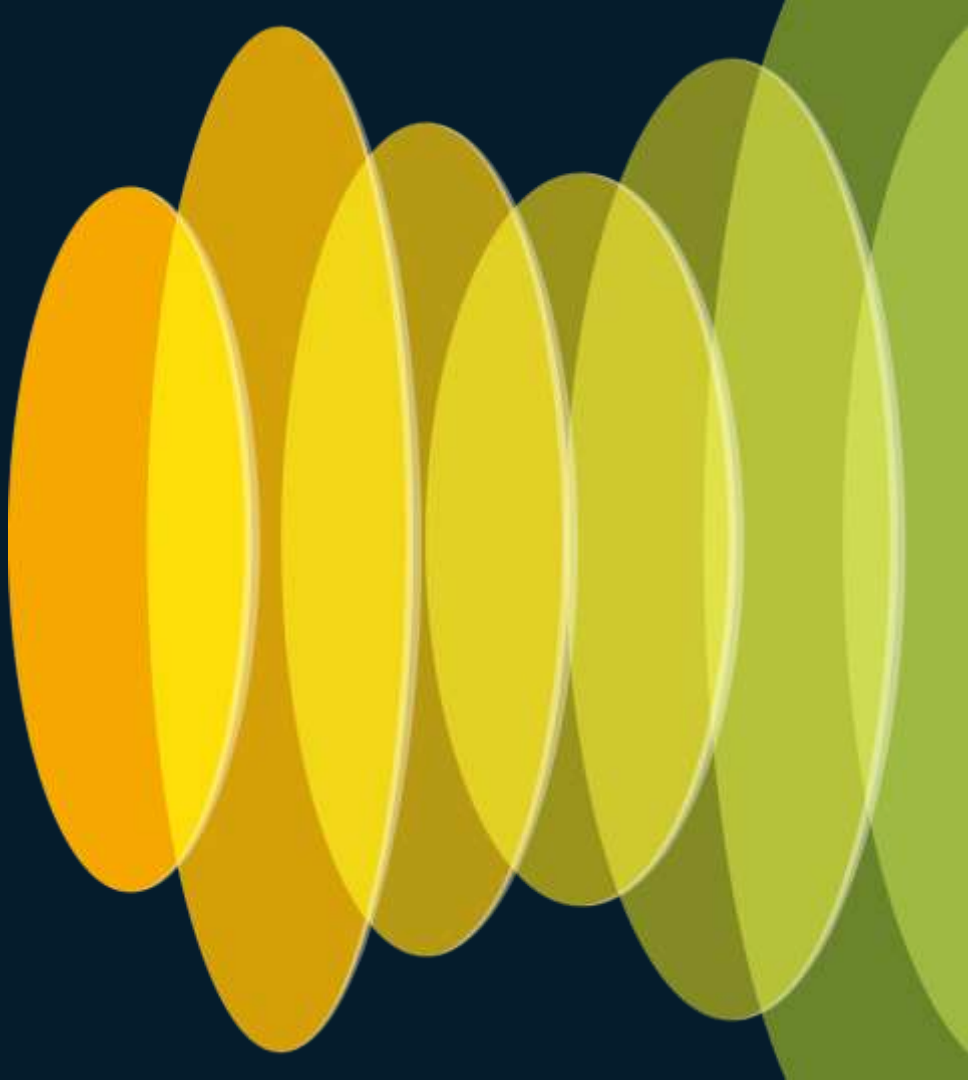
Enabling For a Single Service

To be able to collect relevant logs that span a longer period, only enable event log retention for the specific services/features you need



```
no bloggerd log-dump all
 //Disables the log file retention feature for all services
bloggerd log-dump sap 351
 //Enables the feature for the sap 351 (aclmgr service)
```

Trigger-Based Event Log Auto-Collection



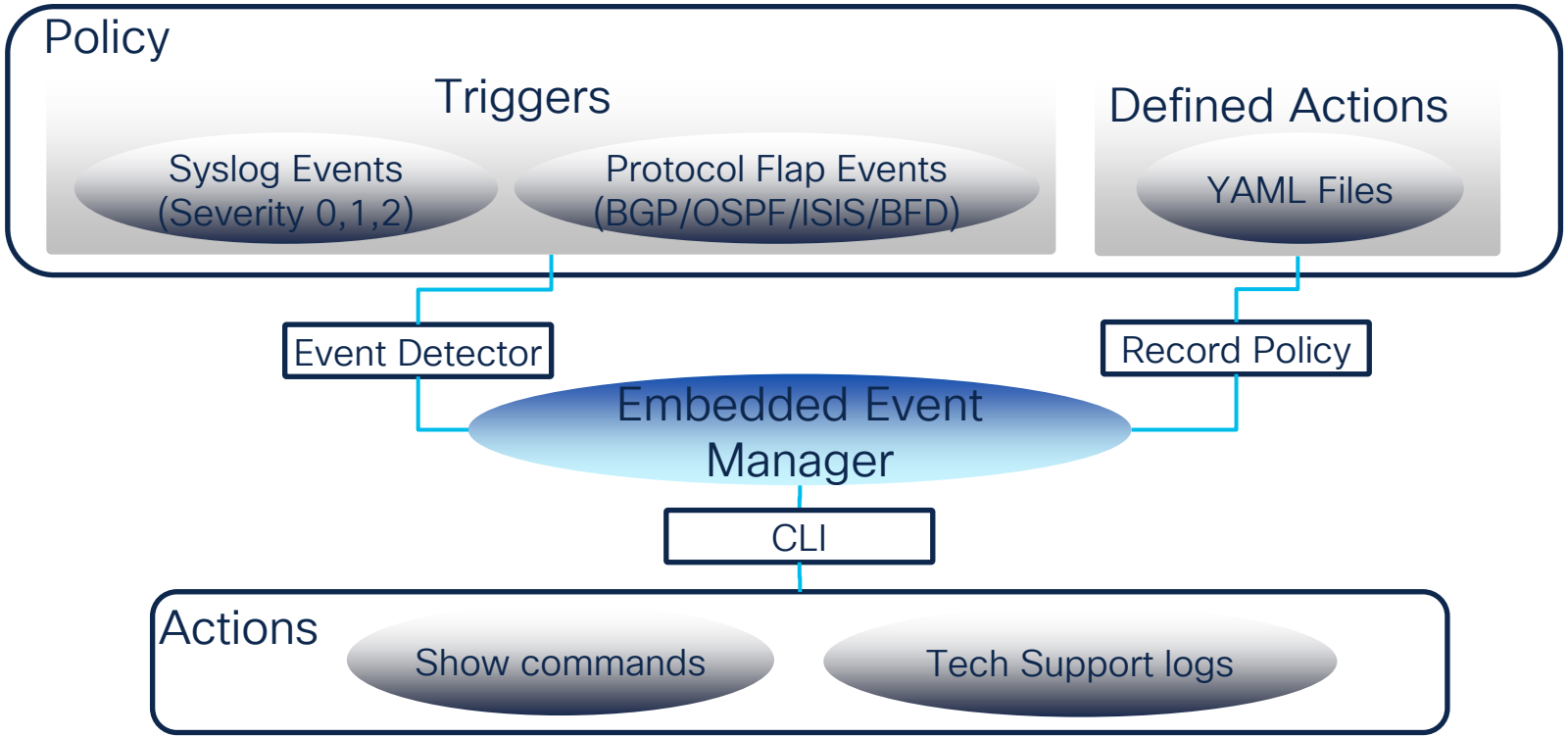
Trigger-Based Event Log Auto-Collection

Feature capabilities

- Automatically collect relevant data when issues occur
- Store logs locally on the switch or remotely on an external server
- Support severity 0, 1, and 2 syslogs
- Support unexpected protocol events such as BGP, BFD, OSPF

Trigger-Based Event Log Auto-Collection

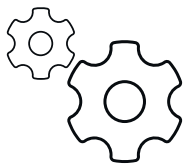
Triggering Mechanism



Trigger-Based Event Log Auto-Collection

Auto-Collection YAML File

The Auto-Collection YAML file is specified in the action command in the EEM function



```
manager applet test_1 override __syslog_trigger_default
  action 1.0 collect test.yaml $_syslog_msg
```

Default YAML file	Located in the switch directory: /bootflash/scripts
Component-specific YAML files	Located in the switch directory: /bootflash/scripts/default-autocollect

Trigger-Based Event Log Auto-Collection

Example YAML File

```
N9K# show file bootflash:scripts/test.yaml
version: 1
components:
   securityd:
        default:
            tech-sup: port
            commands: show module
   platform:
        default:
            tech-sup: port
            commands: show module
```

Key: Value	Description
version: 1	Set to 1. Any other number creates an incompatibility for the auto collect script
components:	Keyword specifying that what follows are switch components
securityd:	Name of the syslog component (securityd is a facility name in syslog)
default:	Identifies all messages belonging to the component
tech-sup: port	Collect tech support of the port module for the securityd syslog component
commands: show module	Collect show module command output for the securityd syslog component

Trigger-Based Event Log Auto-Collection

Example YAML File

Associate auto-collect metadata only for a specific log

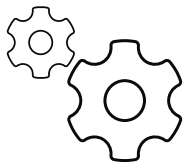
```
N9K# show file bootflash:scripts/test2.yaml
version: 1
components:
  securityd:
    feature_enable_disable:
      tech-sup: security
      commands: show module
```

Key: Value	Description
securityd:	Name of the syslog component (securityd is a facility name in syslog)
feature_enable_disable:	Message ID of the syslog message
tech-sup: security	Collect tech support of the security module for the securityd syslog component
commands: show module	Collect show module command output for the securityd syslog component

Trigger-Based Event Log Auto-Collection

Example YAML File

Use semicolons to separate multiple show commands and tech support key values



```
version: 1
components:
  securityd:
    default:
      commands: show module;show version
      tech-sup: port;lldp
```

Trigger-Based Event Log Auto-Collection

Enable auto-collect

Create an override policy for the __syslog_trigger_default system policy with a custom YAML file and define the specific logs for which information will be collected

```
N9K# show file bootflash:scripts/test.yaml
version: 1
components:
   securityd:
        default:
            tech-sup: port
            commands: show module
```

```
N9K# show running-config eem | egrep -v "\!"


version 10.3(2) Bios:version 05.47
event manager applet test_1 override __syslog_trigger_default
  action 1.0 collect test.yaml $_syslog_msg
```

Trigger-Based Event Log Auto-Collection

YAML folder

The default YAML file can be replaced with a folder inside which more than one YAML files can be present. All the YAML files in the folder must follow the ComponentName.yaml naming convention.

```
event manager applet logging2 override __syslog_trigger_default
   action 1.0 collect test_folder rate-limt 30 $_syslog_msg


ls /bootflash/scripts/test_folder

bgp.yaml ppm.yaml
```

Trigger-Based Event Log Auto-Collection

Enable auto-collect component

Default enabled auto-collect components

```
N9K# dir bootflash:scripts/default-autocollect
        522    Apr 13 08:46:04 2024  aclmgr.yaml
       8819    Apr 13 09:24:48 2024  arp.yaml
        499    Apr 13 08:46:04 2024  bgp.yaml
        490    Apr 13 08:46:04 2024  cdp.yaml
       2501    Apr 13 08:46:04 2024  clis.yaml
        494    Apr 13 08:46:04 2024  dhcp_snoop.yaml
        515    Apr 13 08:46:04 2024  eigrp.yaml
       1275    Apr 13 08:46:04 2024  fex.yaml
```

Enable auto-collection for a single or set of components based on the requirement

```
N9K# bloggerd auto-collect component l2fm enable
Component l2fm auto-collect successfully enabled.
l2fm.yaml file copied from /bootflash/scripts/backup-staging to /bootflash/scripts/default-autocollect
N9K#
N9K# dir bootflash:scripts/default-autocollect/l2fm.yaml
        431    Apr 13 09:31:56 2024  scripts/default-autocollect/l2fm.yaml
```

Trigger-Based Event Log Auto-Collection

Enable auto-collect component

Sample collection of supported logs for a few components

Component: IPQoSMgr

Supported logs:
QOSMGR_MTS_FAILURE
QOSMGR_NETWORK_QOS_POLICY_CHANGE
QOSMGR_LLFC_APPLY_FAILURE
QOSMGR_FCOE_POLICY_NOT_REMOVED

Component: ACLQOS

Supported logs:
ACLQOS_UNEXPECTED_MCAST_FRAMES
ACLQOS_UNEXPECTED_PFC_FRAMES
PPF_SUBSCRIPTION_FAILED
ACLQOS_QOS_NO_DROP_CLASSIFICATION_UNSUPPORTED
ACLQOS_QUEUE_LIMIT_IGNORED_ON_FEX
ACLQOS_BUFFER_DRAIN_FAILURE
ACLQOS_BURST_DETECT_FPGA_INCOMPATIBLE
ACLQOS_BURST_DETECT_OVER_THRESHOLD
ACLQOS_FAILED
PPF_FAILED

Trigger-Based Event Log Auto-Collection

Auto-collect history

Trigger: support severity 0, 1, and 2 syslogs

```
2024 Apr 12 21:58:25 N9K %SECURITYD-2-FEATURE_ENABLE_DISABLE: User  has disabled the feature bash-shell
```

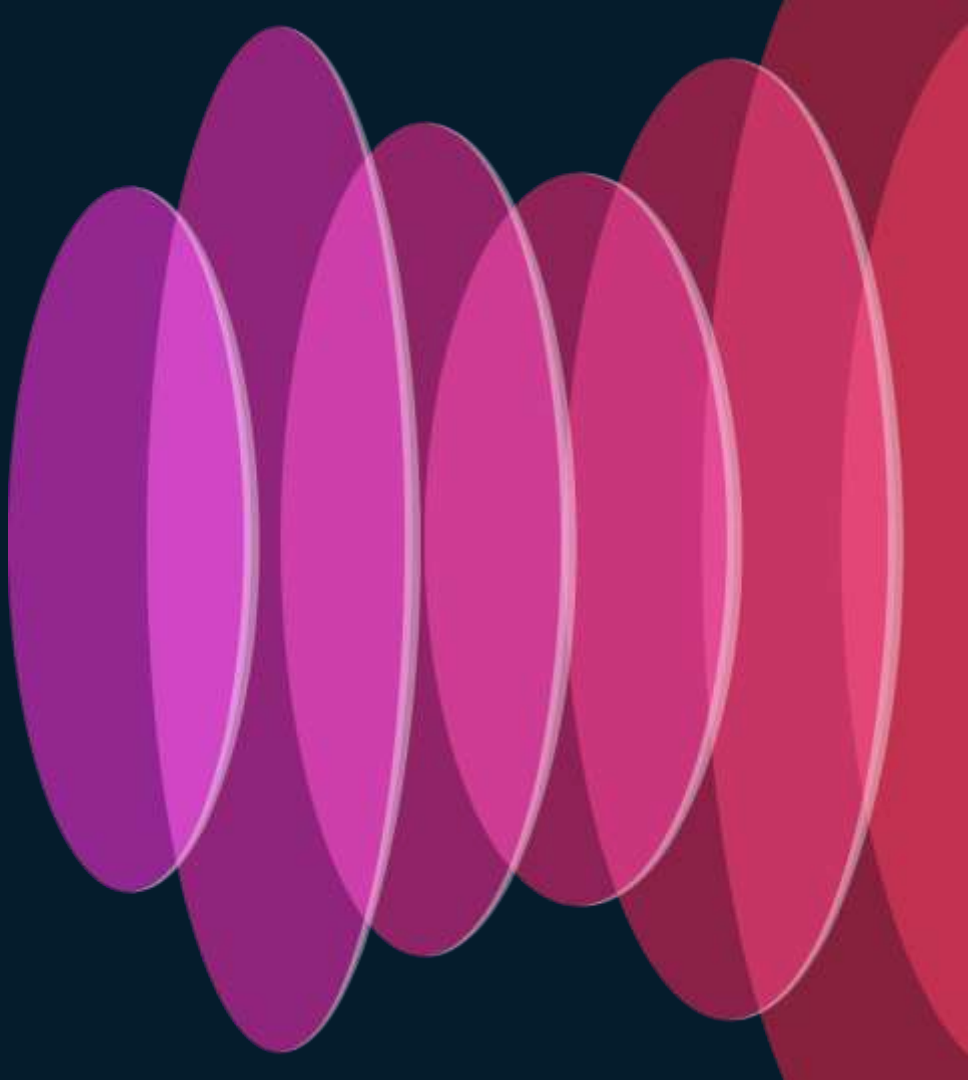
Auto-collect history

```
N9K(config)# show system internal event-logs auto-collect history
DateTime              Snapshot ID  Syslog                                  Status/Secs/Logsize(Bytes)
2024-Apr-12 21:58:34  136390980    SECURITYD-2-FEATURE_ENABLE_DISABLE      PROCESSED:8:5084769
2024-Apr-12 21:58:26  136390980    SECURITYD-2-FEATURE_ENABLE_DISABLE      PROCESSING
```

Auto-collected log files

```
N9K# dir debug:///log-snapshot-auto/
    5084769    Apr 12 21:58:34 2024  20240412215834_136390980_SECURITYD_2_FEATURE_ENABLE_DISABLE_eem_snapshot.tar.gz
```

Integration with Python scripts



Integration with Python scripts

Why Python?

- Complex Logic Handling
- Readability and Maintainability
- Powerful Standard Library
- Better Error Handling

Integration with Python scripts

Cisco Python Package

How to display the details of the Cisco Python package

```
N9K(config)# python3
Python 3.8.18 (default, Oct 30 2023, 01:27:05)
[GCC 5.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cisco
>>> help(cisco)
```

```
NAME
    cisco

PACKAGE CONTENTS
    acl
    bgp
    buffer_depth_monitor
    check_port_discards
    cisco_secret
    feature
    history
    interface
    ipaddress
    key
    line_parser
```

Integration with Python scripts

Python API

API	Description
cli() Example: string = cli (“cli-command”)	Returns the raw output of CLI commands, including control or special characters.
clid() Example: json_string = clid (“cli-command”)	Returns JSON output for cli-command, if XML support exists for the command, otherwise an exception is thrown.
clip() Example: clip (“cli-command”)	Prints the output of the CLI command directly to stdout and returns nothing to Python.

Integration with Python scripts

Example of python API

cli() API

```
>>> from cli import *
>>> cli("conf t ; interface eth1/6 ; no sh")
```

```
N9K# show accounting log
user=admin:cmd=configure terminal ; interface Ethernet1/6 ; no shutdown (REDIRECT)
user=admin:cmd=configure terminal ; interface Ethernet1/6 ; no shutdown (SUCCESS)
```

Integration with Python scripts

Example of python API

clid() API

This example is used to filter interface with up status.

```
>>> from cli import *
>>> import json
>>> intflist=json.loads(clid('show interface brief'))
>>> i=0
>>> while i < len(intflist['TABLE_interface']['ROW_interface']):
...   intf=intflist['TABLE_interface']['ROW_interface'][i]
...   i=i+1
...   if intf['state'] == 'up':
...     print(intf['interface'])
mgmt0
Ethernet1/1
Ethernet1/2
```

Note: check JSON structure

```
string = clid('show interface brief')
print(string)
```

Integration with Python scripts

Example of python API

clip() API

This example is used to print the output of the CLI command

```
>>> from cli import *
>>> clip('show user-account')
user:admin
        this user account has no expiry date
        roles:network-admin
user:N93K
        this user account has no expiry date
        roles:dev-ops
```

Integration with Python scripts

EEM with python

- Script location

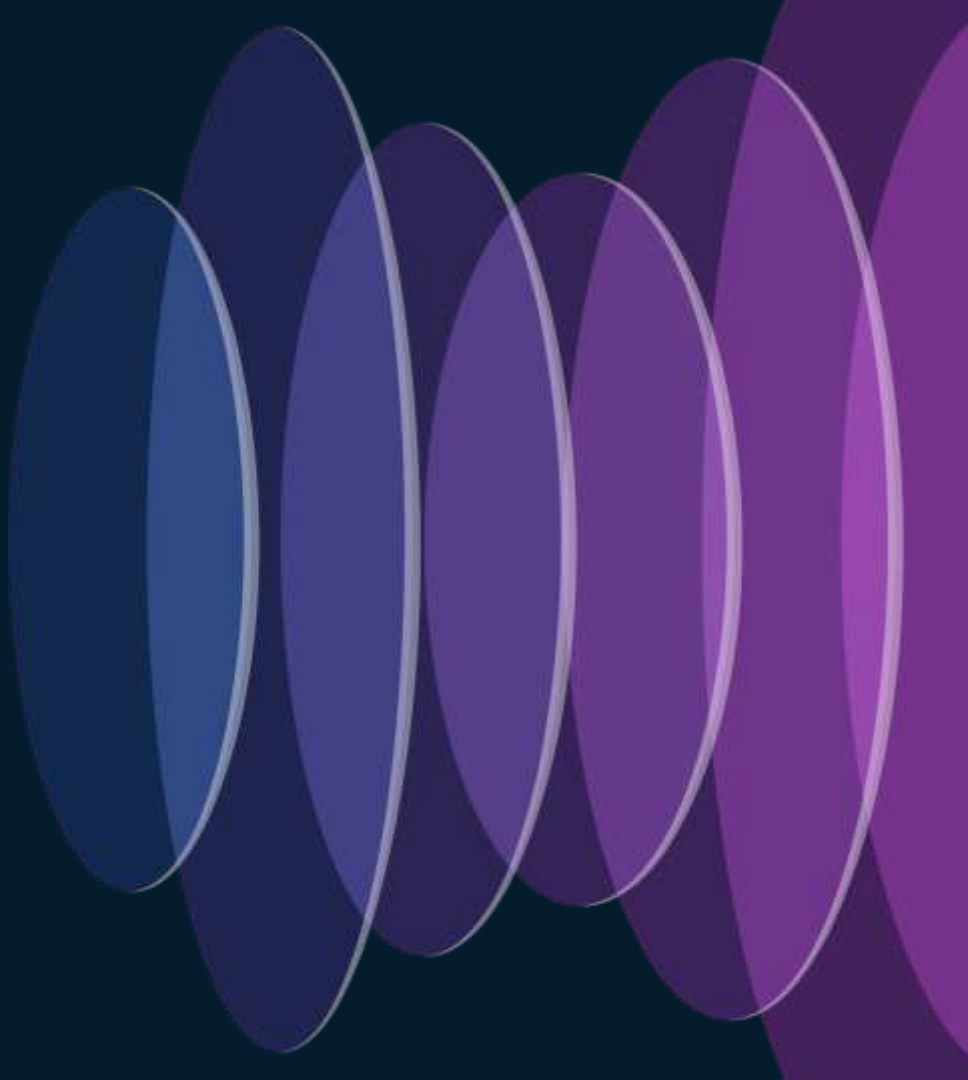
```
N9K# dir bootflash:scripts/pbr_monitor.py
       1297    Apr 11 07:09:00 2024  scripts/pbr_monitor.py
```

- EEM configuration and event log

```
N9K# show run eem

!Command: show running-config eem
!Running configuration last done at: Fri Apr 12 10:27:42 2024
!Time: Fri Apr 12 10:39:00 2024

version 10.4(2) Bios:version 05.51
event manager applet pbr_monitor
  event timer cron cron-entry "30 10 * * *"
  action 1 cli python bootflash:scripts/pbr_monitor.py
  action 2 syslog priority alerts msg eem triggered
```

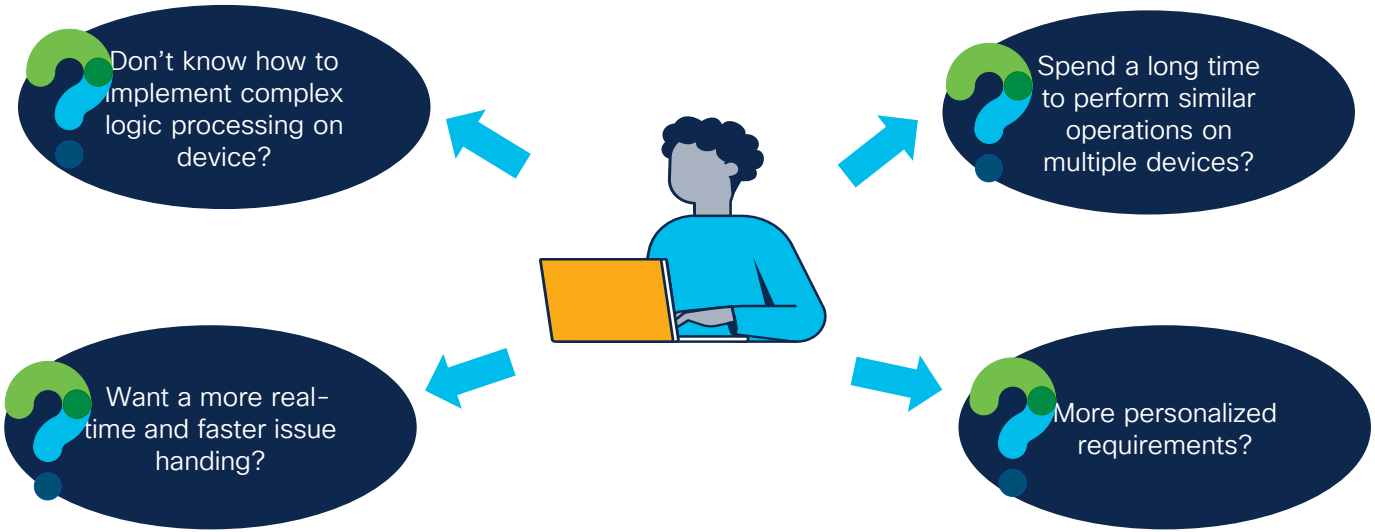
```
N9K# show file logflash:vdc_1/event_archive_22 | last 20
Executing the following commands succeeded:
          python bootflash:scripts/pbr_monitor.py

      syslog priority alerts msg eem triggered
Completed executing policy pbr_monitor
Event Id:54 event type:317700 handling completed
```

Use Cases of running Python scripts



Use Cases of running Python scripts

- Real Life Problems



Use Cases of running Python scripts

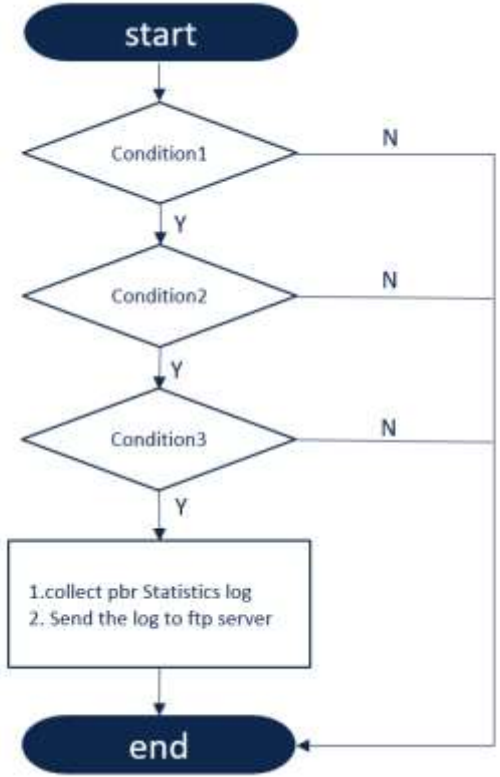
Background:

▶The customer needs to monitor N9K devices that meet specific conditions.

condition1：The version is 10.4(x)
condition2：System uptime is less than 30 days
condition3: Has PBR（route-map A） configurations
……
other necessary conditions

▶ For devices that meet the above conditions, PBR statistics need to be sent to the specified ftp server at a fixed time every day .

*filename is datetime+hostname+pbr-statistics.log



Use Cases of running Python scripts

N9K# show run eem

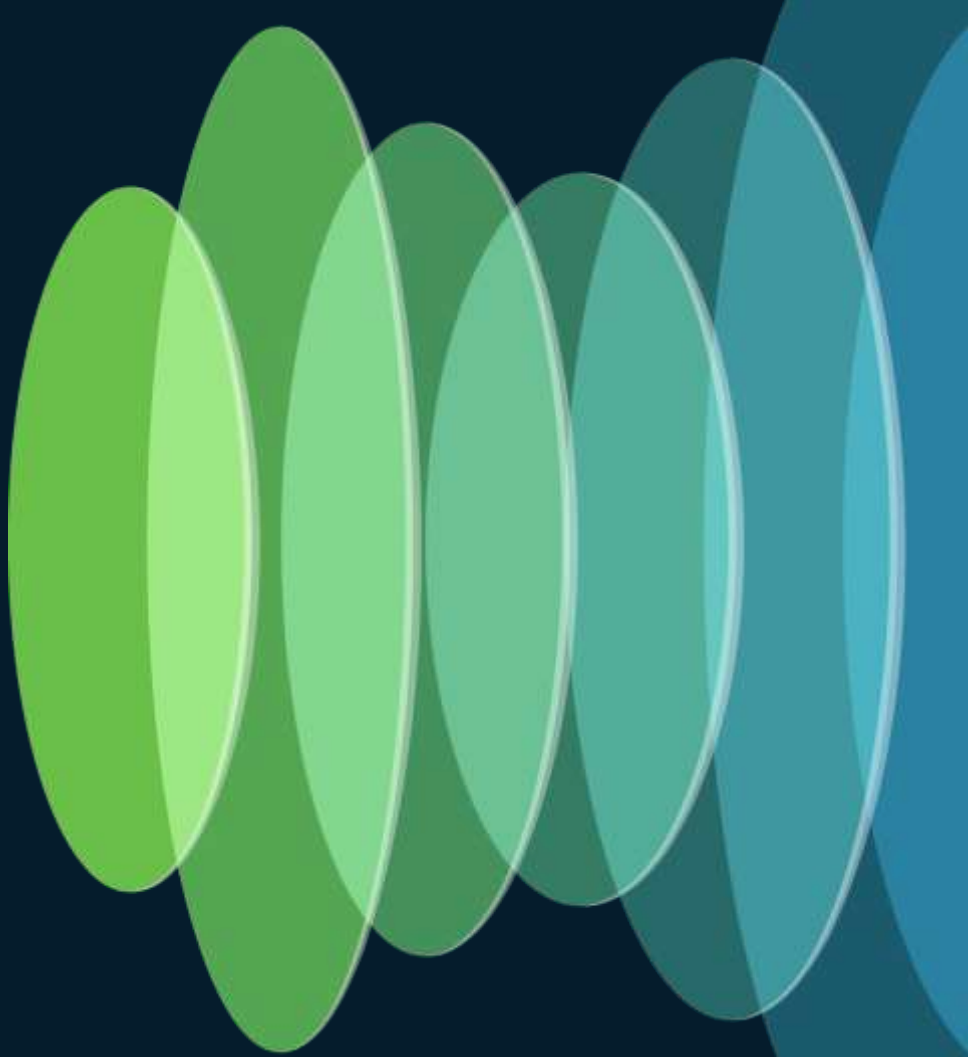
```
event manager applet pbr_monitor
event timer cron cron-entry "30 10 * * *"
action 1 cli python bootflash:scripts/pbr_monitor.py
action 2 syslog priority alerts msg eem triggered
```

Use Cases of running Python scripts

Code example

```
from cli import *
import json
import re
from ftplib import FTP
from datetime import datetime
from io import BytesIO

#condition1:The version is 10.4(x)
json_version = json.loads(clid("show version"))
pbr_log = None
if json_version:
    nxos_version = json_version["nxos_ver_str"]
    if re.search(r"(10\.4\([A-Za-z0-9]+\))",nxos_version):
        #condition2:System uptime is less than 30 days
        json_time = json.loads(clid("show system uptime"))
        if json_time:
                sys_days = json_time["sys_up_days"]
                if sys_days <= "30":
                    #condition3:Has PBR（route-map A） configurations
                    run_config = cli("show run | include route-map[[:space:]]A")
                    if run_config:
                        pbr_log = cli("show route-map A pbr-statistics")

#send logs to FTP server
#set the filename as datetime+hostname+pbr-statistics.log

if pbr_log != None:
    switchname = cli("show hostname").strip('\n')
    ftp_host = “x . x. x . x "
    ftp_username = “x"
    ftp_password = " x "
    ftp = FTP(ftp_host)
    ftp.login(user=ftp_username,passwd=ftp_password)
    current_datetime = datetime.now().strftime("%Y-%m-
%d %H:%M:%S")
    file_name =(f"{current_datetime}_{switchname}_pbr-
statistics.log")
    pbr_log_bytes = bytes(pbr_log,encoding="utf-8")
    pbr_log_stream = BytesIO(pbr_log_bytes)
    ftp.storbinary(f'STOR {file_name}', pbr_log_stream)
    pbr_log_stream.close()
    ftp.quit()
```

References



References

Cisco Nexus 9000 Series NX-OS System Management Configuration Guide, Release 10.4(x)

Cisco Nexus 9000 Series NX-OS Programmability Guide, Release 10.4(x)

Key notes/Take away

- Capabilities of Advanced EEM:

  Extended Log File Retention

  Trigger-Based Event Log Auto-Collection

  Integration with Python scripts

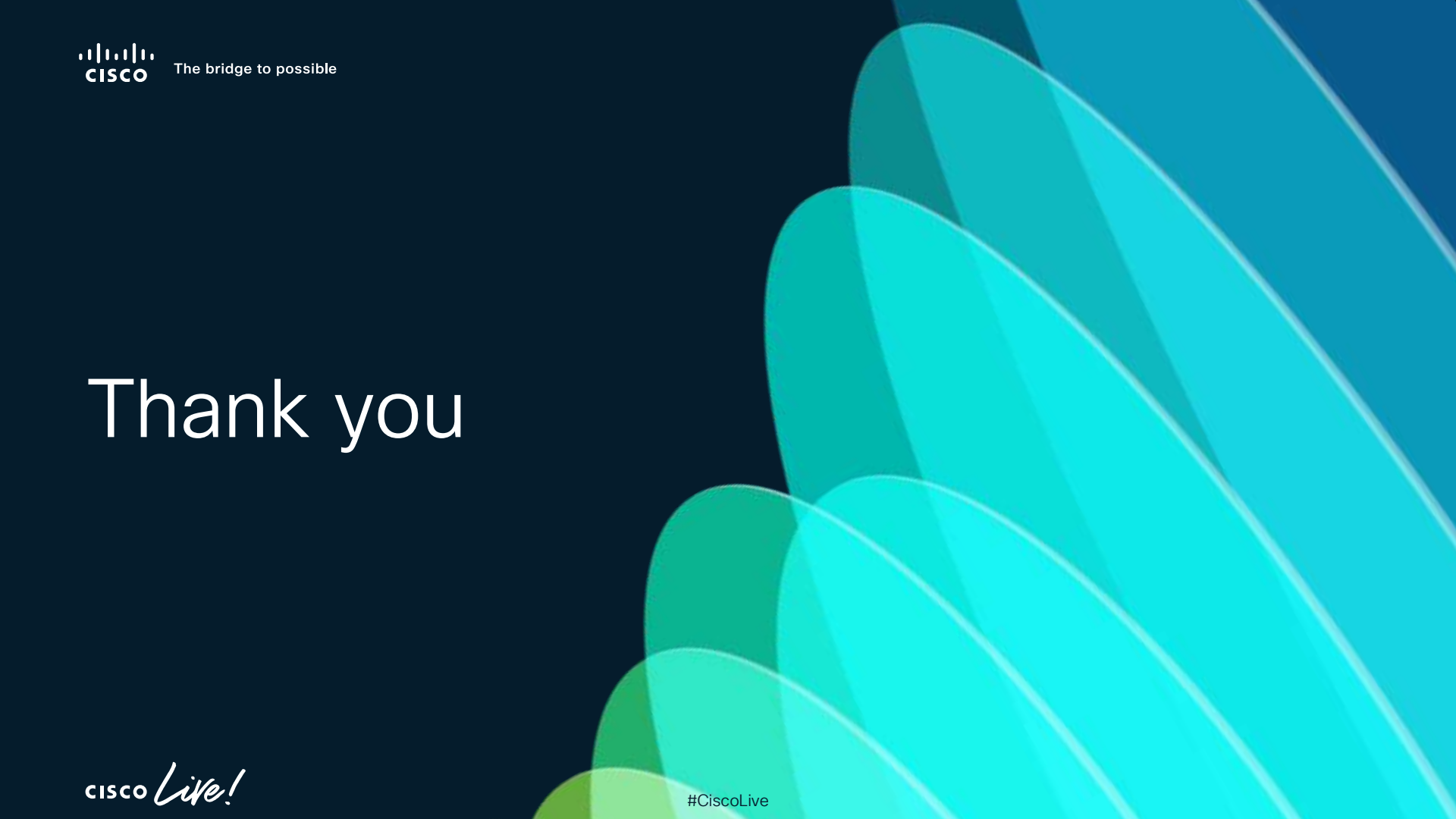
- Python and YAML Integration with EEM:

  How Python and YAML can be used to create EEM policies.

  Examples of Python scripts and YAML configurations for EEM.

- Use Cases:

Real-world examples of EEM Python automation.


The bridge to possible
Thank you
cisco Live!
#CiscoLive

cisco Live!

#CiscoLive