



You make **possible**



Are You Utilizing your Campus Network to its Full Potential?

Shashank Singh
Technical Leader, Cisco

BRKRST-2600

CISCO *Live!*

Barcelona | January 27-31, 2020



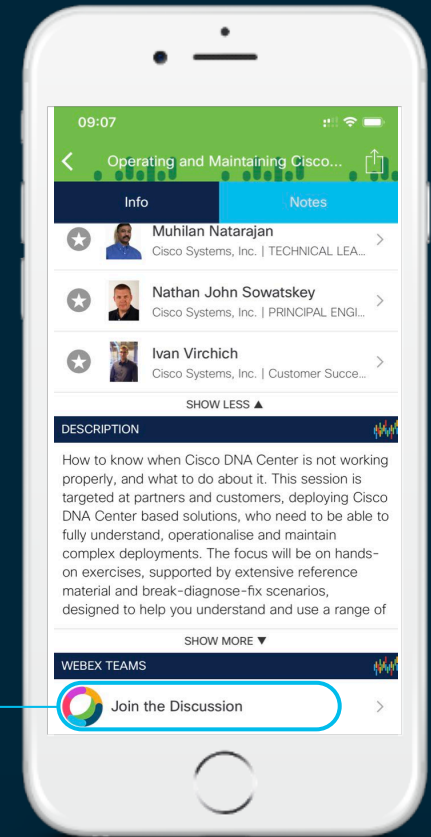
Cisco Webex Teams

Questions?

Use Cisco Webex Teams to chat with the speaker after the session

How

- 1 Find this session in the Cisco Events Mobile App
- 2 Click “Join the Discussion”
- 3 Install Webex Teams or go directly to the team space
- 4 Enter messages/questions in the team space



Agenda

- Introduction
- Advanced platform visibility
- Zero Touch Provisioning (ZTP)
- PoE Innovations & StackPower
- Manageability Innovations
- Support Packages
- Appendix (Script Examples for Reference)

Your Instructor Today...

Shashank Singh

Technical Leader, Cisco Services

Email: shashasi@cisco.com

Twitter: [@shashankcisco](https://twitter.com/shashankcisco)

Shashank is a Technical Leader with Routing and Switching Technical Leadership team in San Jose, CA and has extensive experience in troubleshooting Catalyst line of products.

Shashank works as an escalation point for Cisco Customer Experience teams and partners with engineering teams to solve some of the most complex customer problems pertaining to Cisco switches.

Prior to this role, Shashank has worked as a TAC engineer for over five years, troubleshooting switching products and technologies. Shashank has a software development background from his previous role as a software developer in General Electric.

Introduction

Cisco Catalyst 9000 switches at a glance

Enabling a new era of intent-based networking



Secure

- Encrypted Traffic Analytics
- MACsec link encryption
- Trustworthy solutions
- Group-based policy
- Full Flexible NetFlow



IoT Convergence

- Constrained Application Protocol (CoAP)
- Cisco DNA Service for Bonjour
- Perpetual PoE
- IEEE 1588 Audio Video Bridging (AVB)



Mobility

- Fabric-enabled wireless
- Embedded Cisco Catalyst 9800 Series wireless controller
- Unified control and policy
- Wired and wireless guest access

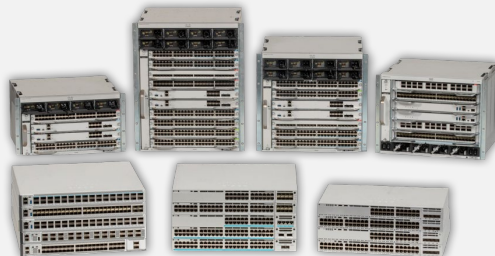




Cloud

- DevOps toolkit
- NETCONF/YANG models
- Streaming telemetry
- Patching and Graceful Insertion and Removal (GIR)
- Application hosting

Software and hardware Innovations

Built on a modern modular OS (Cisco IOS XE) and programmable ASIC



- **Cisco Catalyst 9600 Series switches** 
Lead modular core
- **Cisco Catalyst 9500 Series switches**
Lead fixed core
- **Cisco Catalyst 9400 Series switches**
Lead modular access
- **Cisco Catalyst 9300 Series switches** 
Lead fixed access
- **Cisco Catalyst 9200 Series switches**
Entry-level fixed access

Cisco Catalyst 9000 innovations

→ Converged ASIC



Open Cisco IOS XE

→ Single image



UADP ASIC

→ Common licensing



Benefits

Flexibility, operational simplicity, and optimized cost

CISCO *Live!*

Cisco DNA Center™



Segmentation



Automation



Assurance



Analytics

- Streaming telemetry
- Network monitoring



On-box app hosting

- Monitoring security IoT



DevOps toolkit

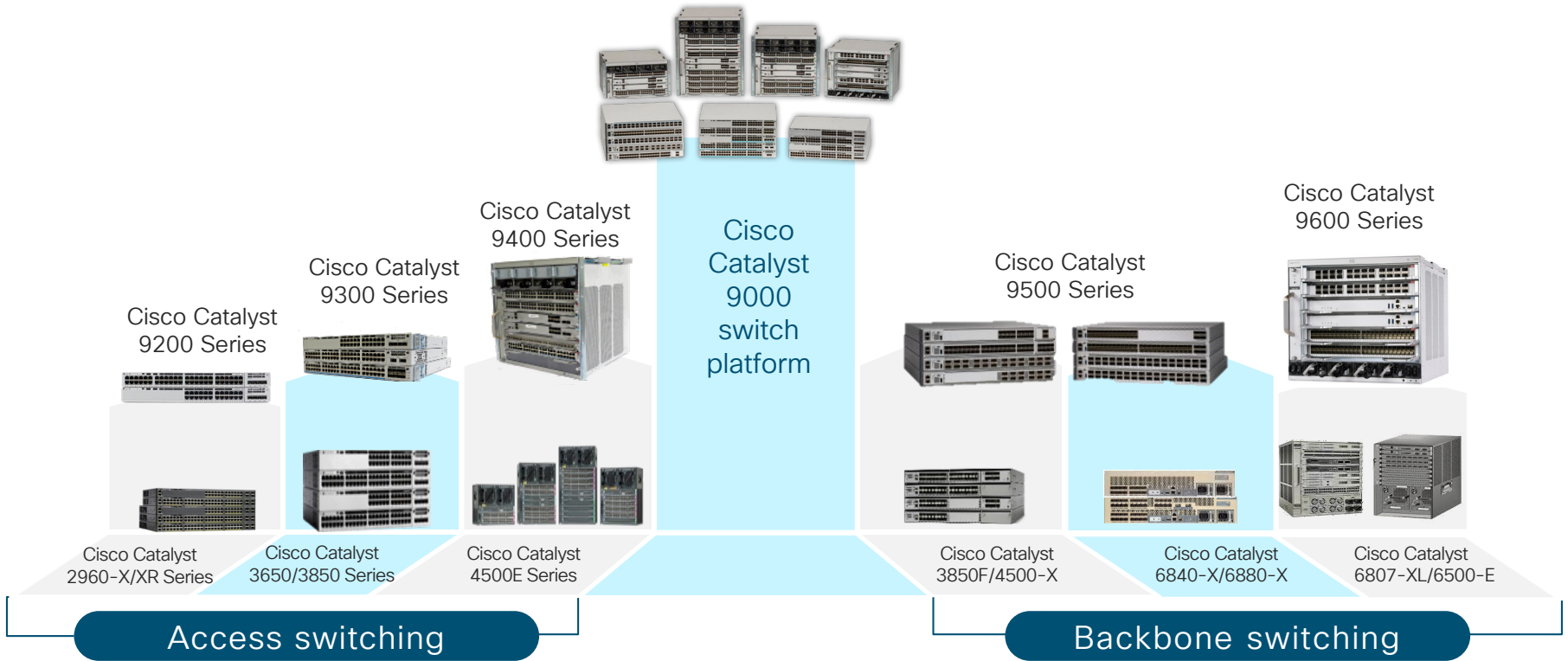
- Zero-touch provisioning
- Model-driven programmability
- Server management tools



High availability

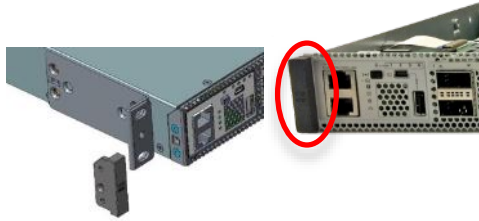
- Patching, GIR, In-Service Software Upgrades (ISSU)

Campus switching portfolio

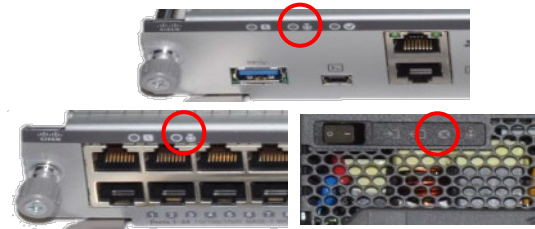


Simplified operations and serviceability with Cisco Catalyst 9000

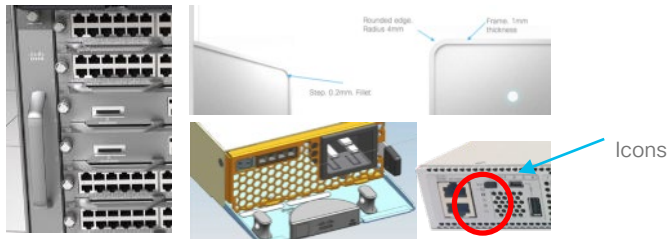
Inventory management efficiency with built-in RFID



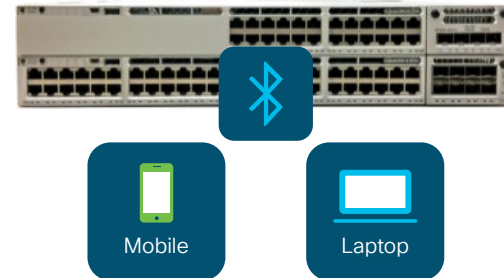
Ease of serviceability with blue beacons on each component



Ergonomic design with industry-standard icons



Wireless console access with Bluetooth



Advanced Platform Visibility Guestshell

Real life problems and tools



Event driven actions

Periodic monitoring

Day 0/1 deployment

Network-wide configuration



Application Hosting

Embedded Event Manager (EEM)

Model driven programmability
(RESTCONF etc)

TCL

Cisco Guestshell

Cisco DNAC/PnP



On the box Python

Guestshell overview

- 64 bit application environment running on IOS XE and NX OS platforms
- Isolated user space – Fault isolation, Resource isolation
- Access to bootflash.
- Linux Commands – Integrate into existing Linux workflows
- Bundled with Python- Cisco cli python library for CLI operations and automated output collection.
- Default access only permitted through Mgmt VRF. Additional options to integrate with device data plane.

Open Shell is an embedded Linux environment that allows customers (DevOps) to develop and run custom python applications for automated control and management of the IOSXE family of switches.

IOSXE CLI interface

- Access the management Shell from IOSXE CLI
- Access IOSXE CLI from within the management Shell

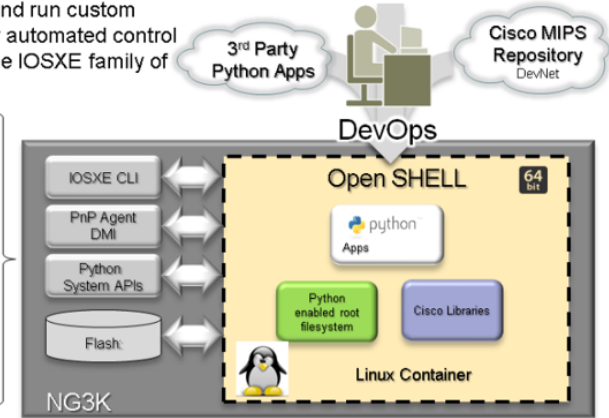
PnP Agent / DMI

Python System APIs

- Cisco cli and socket libraries

Flash

- Read/Write access to the IOSXE flash (bootflash / harddrive on routers)



DMI = Data Model Interface = Netconf/Yang interface
PnP = Plug N Play = Zero Touch provisioning

Also Supported...

- **ZTP** – Zero Touch Provisioning can retrieve a Python script via DHCP at boot time
- **EEM** – Use Embedded Event Manager to trigger a Python script in response to an event

Guestshell

Preparation

```
Cat9K#conf t
Cat9K(config)#iox
```

Start IOx

```
Cat9K#show iox-service
IOx Infrastructure Summary:
```

```
-----
IOx service (CAF) :      Running
IOx service (HA)  :      Running
IOx service (IOxman) :   Running
Libvirtd         :      Running
```

Ensure service is running

```
Cat9K#conf t
Cat9K(config)#app-hosting appid guestshell
Cat9K(config-app-hosting)#app-vnic management guest-interface 0
```

Configuration required before activation
Configuration includes VPG interface for Guestshell
Activation is via simplified exec command

```
Cat9K#guestshell enable
```

```
Cat9K#show app-hosting detail | b Network
Network interfaces
```

```
-----
eth0: MAC address      :      52:54:dd:42:ee:b3
IPv4 address          :      192.168.30.2
```

IP address configured on eth0 in guestshell will
always be 192.168.30.2 even though specific IP
address could be given in IOS-XE configuration



Guestshell

Verification

```
Cat9K#show app-hosting list
```

```
App id          State
```

```
-----  
guestshell     RUNNING
```

Ensure guestshell is up and running

Double-check that Linux inherited mgmt configuration

```
Cat9K#guestshell run sudo ifconfig eth0
```

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
  inet 192.168.30.2 netmask 255.255.255.248 broadcast 192.168.30.7  
  inet6 fe80::5054:ddff:fe20:e54c prefixlen 64 scopeid 0x20<link>  
  ether 52:54:dd:20:e5:4c txqueuelen 1000 (Ethernet)  
  RX packets 8 bytes 648 (648.0 B)  
  RX errors 0 dropped 0 overruns 0 frame 0  
  TX packets 8 bytes 648 (648.0 B)  
  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
Cat9K#guestshell destroy
```

```
Guestshell destroyed successfully
```

```
Cat9K(config)#no iox
```

```
Cat9K(config)#iox
```

```
Cat9K(config)#exit
```

```
Cat9K#guestshell enable
```

In case of any issues you may restart guestshell / IOX service

Guestshell

Using cli library

```
Cat9K#guestshell run python  
Python 2.7.5 (default, Aug 4 2017, 00:39:18)  
[GCC 4.8.5 20150623 (Red Hat 4.8.5-16)] on linux2  
Type "help", "copyright", "credits" or "license" for more information.
```

Run python and execute cisco CLI.

```
>> import cli  
>>> interface_cmd = 'show run interface gi1/0/1'  
>>> interface_cfg = cli.cli(interface_cmd)  
>>> print (interface_cfg)
```

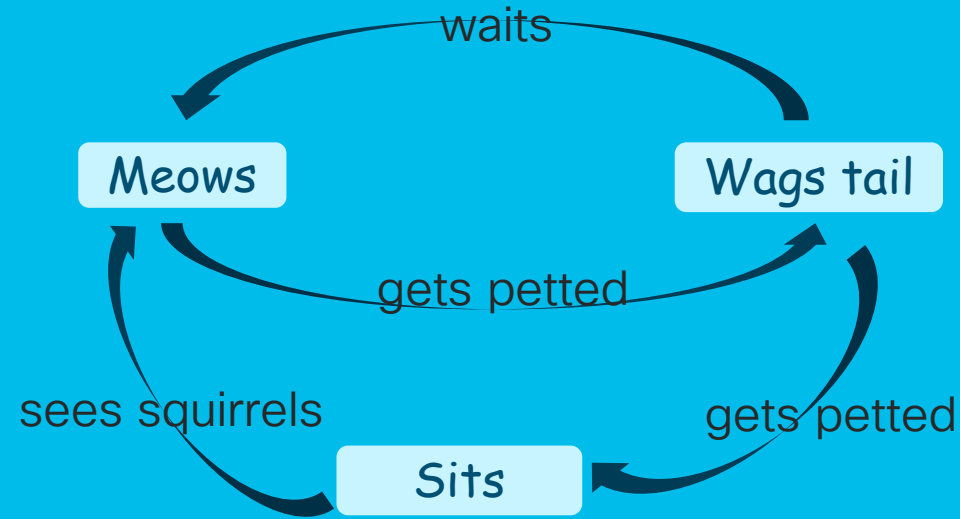
Run basic script to run commands

```
Building configuration...  
Current configuration : 38 bytes  
!  
interface GigabitEthernet1/0/1  
end
```

Change configuration (i.e. go to interface gi1/0/1 and set description and IP address)

```
>> import cli  
>>> config_cmd = '''interface GigabitEthernet1/0/1\ndescription Configured by  
Python\nno switchport\nip address 10.1.1.1 255.255.255.0'''  
>>> config_result = cli.configure(config_cmd)  
>>> print(config_result)  
[ConfigResult(success=True, command='interface GigabitEthernet1/0/1', line=1, output='', notes=None), ConfigResult(success=True, command='description Configured by Python', line=2, output='', notes=None), ConfigResult(success=True, command='no switchport', line=3, output='', notes=None), ConfigResult(success=True, command='ip address 10.1.1.1 255.255.255.0', line=4, output='', notes=None)]
```

Events?



✓ Syslog

✓ Value/string in an output

✓ SNMP result

✓ Configuration change

✓ Custom Triggers

Regular Expressions



Regex Cheatsheets & Activities

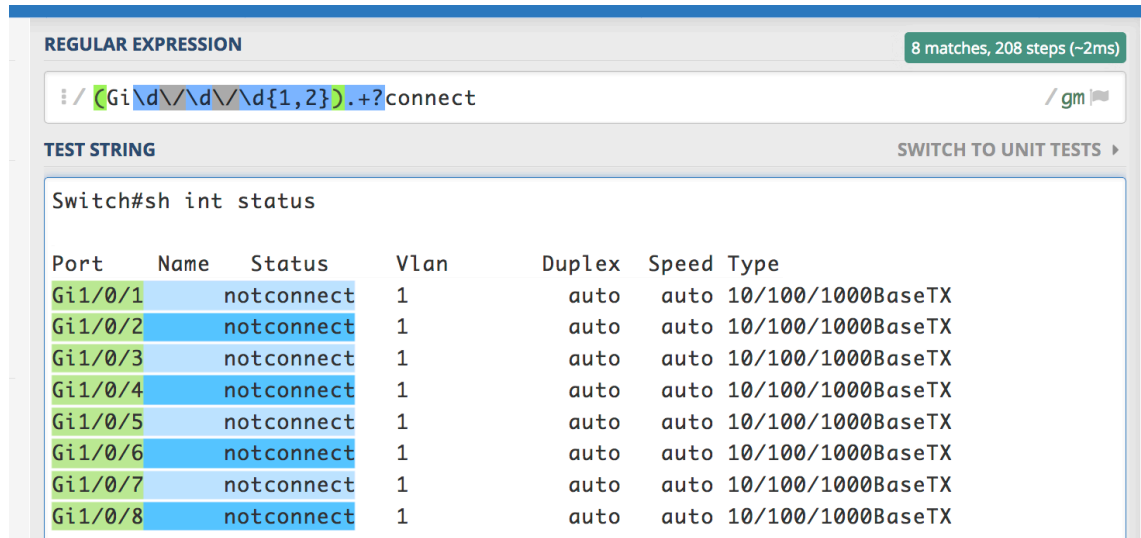
 <https://regex101.com/> - Test your expressions against a string, very useful tool.

<http://www.rexegg.com/regex-quickstart.html>

<https://alf.nu/RegexGolf>

<https://regexone.com/>

<http://regexpr.com/>



REGULAR EXPRESSION 8 matches, 208 steps (~2ms)

`/ [Gi\d\\d\\d{1,2}].+?connect /gm`

TEST STRING SWITCH TO UNIT TESTS ▶

```
Switch#sh int status
```

Port	Name	Status	Vlan	Duplex	Speed	Type
Gi1/0/1		notconnect	1	auto	auto	10/100/1000BaseTX
Gi1/0/2		notconnect	1	auto	auto	10/100/1000BaseTX
Gi1/0/3		notconnect	1	auto	auto	10/100/1000BaseTX
Gi1/0/4		notconnect	1	auto	auto	10/100/1000BaseTX
Gi1/0/5		notconnect	1	auto	auto	10/100/1000BaseTX
Gi1/0/6		notconnect	1	auto	auto	10/100/1000BaseTX
Gi1/0/7		notconnect	1	auto	auto	10/100/1000BaseTX
Gi1/0/8		notconnect	1	auto	auto	10/100/1000BaseTX

Trigger off an interface statistic

EEM example

```
event manager applet test_applet
event interface name GigabitEthernet1/0/1 parameter output_errors entry-op ge
  entry-val 0 poll-interval 1 entry-type value
action 1.0 syslog msg "Found errors on interface GigabitEthernet1/0/1"
```

```
Cat9K(config-applet)#event interface name Gil/0/1 parameter ?
input_errors          Number of damaged packets received
input_errors_crc      Number of packets received with CRC errors
input_errors_frame    Number of framing ERR packets received
input_errors_overrun  Number of overruns and resource errors
input_packets_dropped Number of packets dropped from input Q
interface_resets      Number of times an interface has been reset
output_buffer_failures Number of failed buffers
output_buffer_swappedout Number of packets swapped to DRAM
output_errors         Number of packets errored on output
output_errors_underrun Number of underruns on output
output_packets_dropped Number of packets dropped from output Q
receive_broadcasts    Number of broadcast packets received
receive_giants        Number of too large packets received
receive_rate_bps      Interface receive rate in bits/sec
receive_rate_pps      Interface receive rate in pkts/sec
receive_runts         Number of too small packets received
receive_throttle      Number of times the receiver was disabled
reliability           Interface reliability as a fraction of 255
rxload               Receive rate as a fraction of 255
transmit_rate_bps     Interface transmit rate in bits/sec
transmit_rate_pps     Interface transmit rate in pkts/sec
txload               Transmit rate as a fraction of 255
```

Event Type	Description
event snmp oid	Look for the data of a specific SNMP OID. The data in these come in many data types, strings, ints, and booleans. A logical operator will be needed to trigger the event.
event syslog	Perform an action based off of the state of an interface.

Other common event types

CISCO *Live!*

Other interface parameters

Packet capture

EEM + Python – Automated data collection

```
event manager applet detect_high_interface_rate
event interface name GigabitEthernet1/0/1 parameter receive_rate_bps entry-op ge
  entry-val 1000 poll-interval 1 entry-type value
action 0.01 syslog msg "High Rx rate detected, running automated packet capture."
action 0.02 cli command "enable"
action 0.03 cli command "guestshell run python /flash/g_s_script/packet_capture.py Gi1/0/1"
```

Trigger is receive rate on Gi1/0/1

Variables can be passed to python script at runtime



packet_capturet.py

```
import cli,time,sys

t = time.localtime()
timestamp = time.strftime('%b-%d-%Y_%H%M%S', t)
PATH_NAME = ("flash:/g_s_script/" + timestamp + ".pcap")
INTERFACE_NAME = str(sys.argv[1])

cli.execute("enable")
cli.execute("mon cap pack_cap int %s in file loc %s size 10 match any" % (INTERFACE_NAME, PATH_NAME))
cli.execute("monitor capture pack_cap start")
cli.execute("send log Capture running on %s for 10 sec" %INTERFACE_NAME)
cli.executep("show monitor capture pack_cap")
time.sleep(10)
cli.execute("monitor capture pack_cap stop")
cli.execute("send log Capture saved in %s" %PATH_NAME)
```

Unique filename using timestamp



Interface name received from EEM as variable

Periodic Events



At certain time/day



Every N sec

Common trigger mechanisms

Watchdog timers:

- event timer watchdog time 120 maxrun 100

Time before script runs in seconds

Time before system kills script in seconds

Cron Timers:

- event timer cron cron-entry "1 2 3 4 5"

"minute hour day-of-month month day-of-week"

Cron Timer Examples:

- "15 9 * * *" 9:15 every morning
- "00 10 * * 1-5" 10:00 every Mon. through Fri.

Shut down a port at 6pm every Thursday.

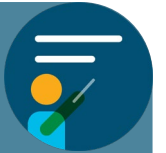
EEM example

```
event manager applet shut_port authorization
bypass
event timer cron cron-entry "0 18 * * 4"
action 1.0 syslog msg "shutting port"
action 1.2 cli command "enable"
action 1.3 cli command "config t"
action 1.4 cli command "interface Te2/1/15"
action 1.5 cli command "shutdown"
action 1.6 cli command "end"
```

"minute hour day-of-month month day-of-week"

And cron lets you do a lot more...

17 8 * * *	--run daily at 8:17 am
17 20 * * *	--run daily at 8:17 pm
00 4 * * 0	--run at 4 am every Sunday
42 4 1 * *	--run 4:42 am every 1st of the month
01 * 19 07 *	--run hourly on the 19th of July
59 11 * * 1-5	--run at 11:59 Monday, Tuesday, Wednesday, Thursday and Friday
*/1 * * * *	--Run every minute
* */2 * * *	-- Run every two hours



ASIC level drops and exceptions

```
show platform hardware fed switch active fwd-asic drop exceptions
```

```
****EXCEPTION STATS ASIC 0****
```

```
NO_EXCEPTION 0
IPV4_CHECKSUM_ERROR 0
ROUTED_AND_IP_OPTIONS_EXCEPTION 0
CTS_FILTERED_EXCEPTION 0
SIA_TTL_ZERO 0
ALLOW_NATIVE_EXCEPTION_COUNT 0
ALLOW_DOT1Q_EXCEPTION_COUNT 0
ALLOW_PRIORITY_TAGGED_EXCEPTION_COUNT 0
ALLOW_UNKNOWN_ETHER_TYPE_EXCEPTION 0
IP_SOURCE_GUARD_VIOLATION 0
SECURE_L3IF_LEARNING_VIOLATION 0
AUTH_DRIVEN_DROP 0
VLAN_LOADBALANCE_GROUP_DENY 0
RPF_UNICAST_FAIL 0
RPF_UNICAST_FAIL_SUPPRESS 0
RPF_UNICAST_CHECK_INCOMPLETE 0
RPF_MULTICAST_FAIL 0
PKT_DROP_COUNT 0
SOURCE_ROUTE_EXCEPTION 0
IGR_MISC_FATAL_ERROR 0
BLOCK_FORWARD 0
POLICER_DROP 0
DENY_ROUTE 0
DENY_BRIDGE 0
STATIC_MAC_VIOLATION 0
STATIC_IP_VIOLATION 0
FPM_DROP_PACKET 0
IGR_EXCEPTION_L4_ERROR 0
IGR_EXCEPTION_L5_ERROR 0
IGR_EXCEPTION_HARDWARE_PARSE_EXCEPTION 0
IGR_EXCEPTION_INVALID_VLAN_DROP 0
IGR_EXCEPTION_31 0
FRAGMENTING_IPV4_WITH_OPTIONS 0
FRAGMENTING_IPV6_WITH_EXTENSIONS 0
ICMP_REDIRECT 0
MTU_FAIL_PUNT_TO_CPU_NO_IP_UNREACHABLE 0
LINK_LOCAL_CHECK_FAIL_NO_IP_UNREACHABLE 0
IP_UNICAST_TTL_REACHED_ZERO 0
MISC_FATAL_ERROR 0
STP_OR_FLEXLINK_DROP 0
PROTECTED_PORT_DROP 0
PVLAN_ISOLATED_CHECK_FAILED 0
PVLAN_COMMUNITY_CHECK_FAILED 0
DEJA_VU_CHECK_FAILED 0
NOT_VLAN_LOAD_BALANCE_GROUP_ALLOWED 0
RSPAN_DROP 85575
SPLIT_HORIZON_DROP 0
SYSTEM_TTL_DROP 0
PRUNED 0
DENY_NO_IP_UNREACHABLE 0
IP_MULTICAST_TTL_REACHED_ZERO 0
MTU_FAIL_DROP_BRIDGED 0
MTU_FAIL_DROP_BRIDGED_IP_ROUTED 0
MTU_FAIL_ERSPAN 0
LINK_LOCAL_CHECK_FAIL_L3M_VALID 0
DENY_NOT_NO_IP_UNREACHABLE 0
MTU_FAIL_PUNT_TO_CPU_NOT_NO_IP_UNREACHABLE 0
LINK_LOCAL_CHECK_FAIL_NOT_NO_IP_UNREACHABLE 0
SGT_CACHE_FULL 0
EGR_L3_ERROR 0
EGR_L4_ERROR 0
EGR_L5_ERROR 0
EGR_HARDWARE_PARSE_EXCEPTION 0
EGR_SHOW_FORWARD_DROP 0
```

Run command multiple times
to check for incrementing
counts

Monitoring Cat9K forwarding ASIC drops

Guestshell example

```
import re,time,cli
import sys

#Get output of- show platform hard fed sw active fwd-asic drop exceptions

sh_drop_exceptions = cli.execute('show platform hard fed sw active fwd-asic drop exceptions')

#Check for non zero value against any field in delta column.
#Ignore rows NO_EXCEPTION PKT_DROP_COUNT BLOCK_FORWARD these rows as they are seen to increment without any
issue too

non_zero_values_delta = re.findall(r"\d+?\s+?\d+?\s+(\S+)\s+?\d+?\s+?\d+?\s+?([1-9]\d*?)\s",
sh_drop_exceptions)
non_zero = 0
if non_zero_values_delta:
    for name, non_zero_delta in non_zero_values_delta:
        if str(name) != "NO_EXCEPTION" and str(name) != "PKT_DROP_COUNT" and str(name) != "BLOCK_FORWARD":
            non_zero =1
            cli.execute("send log" + "Non zero delta value found found %s, for %s. Check 'show platform hard
fed sw active fwd-asic drop exceptions'" % (non_zero_delta, name))

#If not non zero delta values found generating no problem found alert.
if not non_zero:
    cli.execute("send log" + "No problem found")
```

Raw data

Regex to match
desired pattern



Demo

Forwarding ASIC drops



Troubleshooting Bot (Tbot) Scripts

Troubleshooting bot scripts – IOS-XE 17.1.X

Bundled python bots



python



- Python scripts bundled with IOS-XE software that can be run from guestshell in order to confirm hardware programming for Layer 2 and 3 entries.
- Collects the dumps of different components and analyzes them on the box.
- Integrated in the build image, scripts will be copied to the /flash/Tbot directory.

```
Cat9K#dir flash:/Tbot  
Directory of flash:/Tbot/
```

```
286724 -rwx      62196 Nov 21 2019 11:36:28 +00:00 HealthCheckDebugInfra.py  
286725 -rw-      73168 Nov 21 2019 11:36:28 +00:00 L3ParseOutputs.py  
286727 -rw-      64445 Nov 21 2019 11:36:28 +00:00 L3DebugInfraAPIs.py  
286728 -rwx      64395 Nov 21 2019 11:36:28 +00:00 L2DebugInfraAPIs.py  
286729 -rwx      86251 Nov 21 2019 11:36:28 +00:00 L2ParseOutputs.py  
286730 -rwx      72387 Nov 21 2019 11:36:28 +00:00 L2DebugInfraScript.py
```

Tbot scripts on IOS XE 17.1.1

Troubleshooting bot scripts – Demo

Run the `L2DebugInfraAPI.py` script, it will use the other .py files in the Tbot directory.

Running Tbot

```
guestshell run python /flash/Tbot/L2DebugInfraAPIs.py 2182 00a7.428a.93d9
```

```
#####  
#### Layer2 Debugging Tool ####  
#####
```

```
@@@ User Inputs @@@
```

```
-> VLAN ID      : 2182  
-> MAC ADDRESS  : 00a7.428a.93d9
```

```
@@@ L2 Forwarding Programming @@@
```

```
-> PoMembers    : ['Fo6/0/9']  
-> MacAddrIndex : 0  
-> PoMemStats   : ['P']  
-> PoID         : 101  
-> PoNAME       : Port-channel101  
-> PoProtocol   : On
```

```
@@@ Interface to Doppler D Instance Mapping @@@
```

```
-> IntfIfID_List : ['0x92']  
-> IntfAsic_List : ['2']  
-> IntfCore_List : ['0']  
-> IntfInst_List : ['4']
```

Contextual data
gathering and analysis

```
MVID Value and Lead Vlan LookUp Value Matches -- SUCCESS
```

```
@@@ Software Programming @@@
```

```
-> Matm_FPPortsList : 262  
-> Matm_RPPortsList : 262  
-> Matm_RPOmPtr     : OM: 0x3480768298
```

```
Matm RP Active Mac Address Matches -- SUCCESS  
Matm FP Active Table ID Matches -- SUCCESS  
Aom Vlan ID and Mac Address Matches -- SUCCESS  
Aom Status Matches -- SUCCESS
```

```
@@@ Hardware Programming @@@
```

```
-> AsicHexList : []  
-> ECIFIDIntf  : Port-channel101  
-> FedMVID     : 15  
-> ECIFIDGpn   : 2144  
-> AsicVlanList : []  
-> AsicCoreList : ['0']  
-> AsicGpnList  : []  
-> ECIFID      : 0x00000106  
-> MacHandle   : 0x7f34d24dd178  
-> DIInstIDList : ['4']  
-> DIHandle    : 0x7f34d271b758  
-> DIPortIDList : ['20']  
-> DICoreIDList : ['0']  
-> DIAsicIDList : ['2']  
-> SI_PortMapList : 0x00000000 0x00100000  
-> SI_RILList  : ['0x2']  
-> SICoreIDList : ['0']  
-> SIAsicIDList : ['2']  
-> SIHandle    : 0x7f34d2713628  
-> SI_DILList  : ['0x5248']  
-> SIPortIDList : ['20']
```

```
++++ MAC Handle Programming Validation ++++  
Vlan Programming and Mac Handle Programming Matches -- SUCCESS
```

<snip>

CISCO *Live!*

Zero-Touch Provisioning

Zero Touch Provisioning

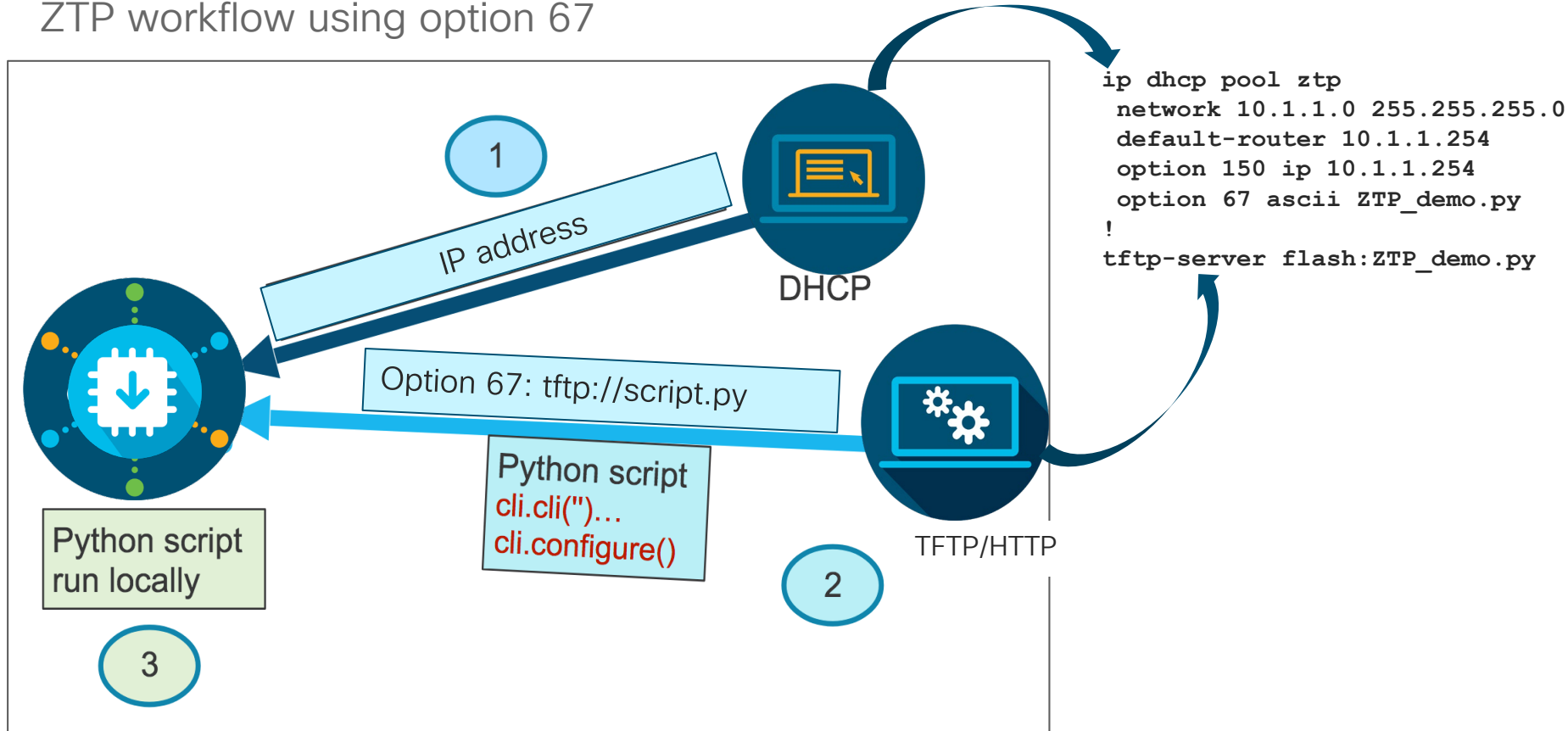
What is this?



Ability to configure a device with no human interacting with device CLI

Zero Touch Provisioning

ZTP workflow using option 67



Zero touch provisioning workflow

1

ZTP device boots up & does not find the startup configuration.

2

Device enters ZTP mode



3

ZTP device searches for DHCP server, gets IP address and enables Guest Shell.

4

ZTP device gets IP address of HTTP/TFTP server, and downloads Python script.

5

Guest Shell executes Python script & applies configuration to the device.

6

After provisioning is complete, Guest Shell gets disabled.

Zero Touch Provisioning

Day 0 configuration

```
print "\n\n *** ZTP Python Script *** \n\n"
```

```
import cli,re
```

```
user = "cisco"
```

```
password = "cisco"
```

```
enable = "cisco"
```

```
print "\n\n *** Configuring hostname *** \n\n"
```

```
cli.configurep(["hostname ZTP-Switch", "end"])
```

Configure hostname

```
print "\n\n *** Configuring credentials *** \n\n"
```

```
cli.configurep(['username {} privilege 15 password {}'.format(user, password)])
```

```
cli.configurep(['enable secret {}'.format(enable)])
```

Configure credentials

```
print "\n\n *** Configuring telnet & ssh *** \n\n"
```

```
cli.configurep(['line vty 0 4', 'login local', 'transport input telnet ssh'])
```

Configure telnet & ssh

Zero Touch Provisioning

Day 1 configuration

```
list = cli.execute('show interface status')
gig_ports = re.findall(r"(Gi\d//\d//\d{1,2}).+?connect", list)
for gig_port in gig_ports:
    print '\n\n Configuring interface {} as access port in vlan 10'.format(gig_port)
    cli.configurep(['int {}'.format(gig_port), 'switchport mode access', 'switchport access vlan 10', 'description configured by python'])

TenG_ports = re.findall(r"(Te\d//\d//\d{1,2}).+?connect", list)
for TenG_port in TenG_ports:
    print '\n\n Configuring interface {} as trunk port'.format(TenG_port)
    cli.configurep(['int {}'.format(TenG_port), 'switchport mode trunk', 'description configured by python'])

print "\n\n *** ZTP Python Script Execution Complete *** \n\n"
```

Get list of all interfaces

Find all 1G interfaces

Configure all 1G interfaces

Find all 10G interfaces

Configure all 10G interfaces

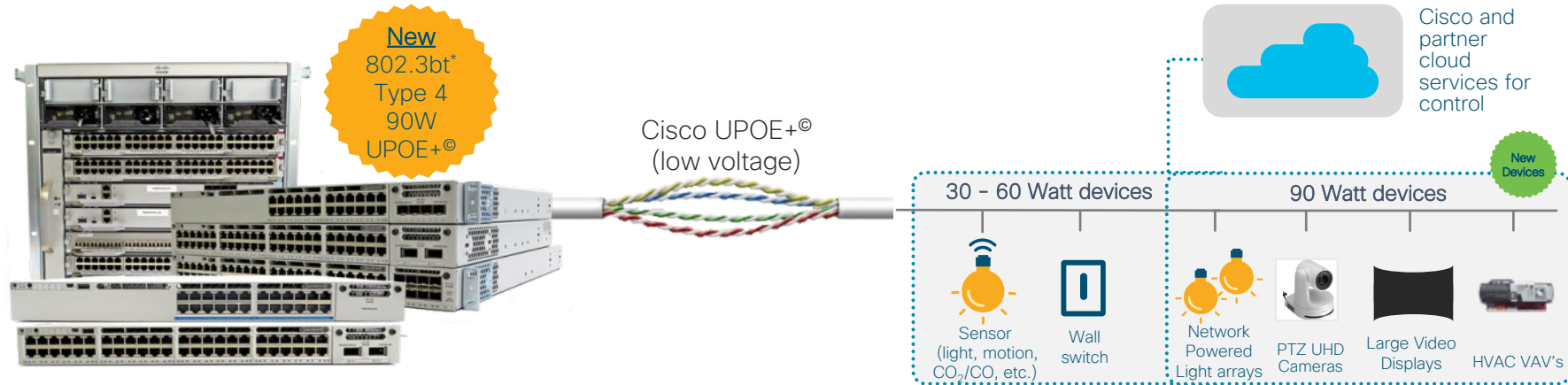
Demo

Zero Touch Provisioning



PoE Innovations & StackPower

UPoE+ on Catalyst 9000

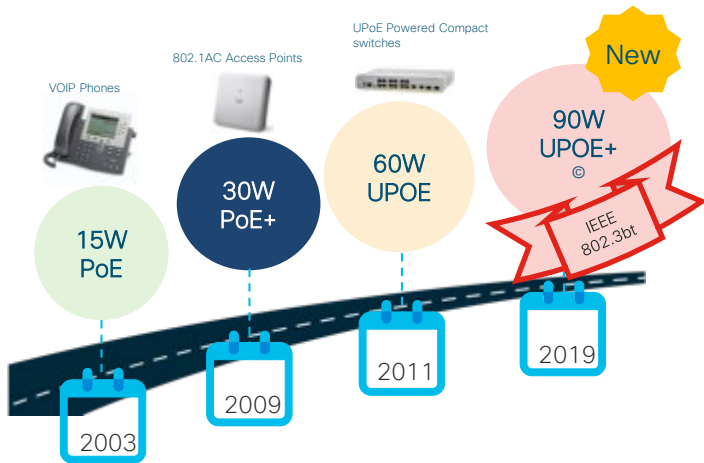


End-to-end solution managed by central IT team that lowers TCO

*Type 4 UPOE+ shipping on Catalyst 9400 and Type 3 shipping on Catalyst 9300. Type 4 UPOE+ on 9300 is on roadmap. Catalyst 9200 is PoE+.

90W UPOE+ standardization is enabling a growing ecosystem of Switches

UPOE+[©] Standardization



Growing ecosystem

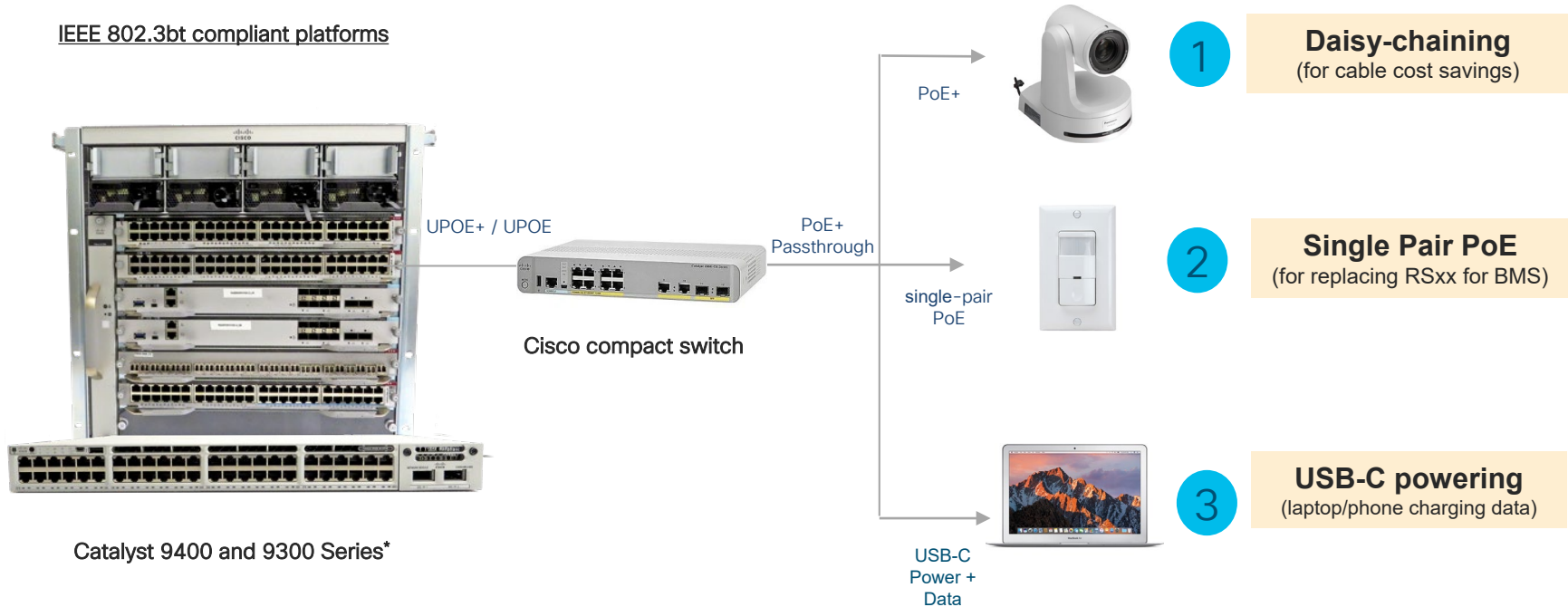


Cisco validated solutions with 90W Switch eco-system

- 802.3bt complements Cisco UPOE by adding 4 new classes of Switches
 - Safety measures ensure up to 90W of power is safely delivered

Push architecture boundaries with UPOE+ 90W

IEEE 802.3bt compliant platforms



Catalyst 9400 and 9300 Series*

*IEEE 802.3bt Type 3 (60W) supported on C9300. Type 4 (90W) in roadmap.

Cisco innovations in PoE deliver a robust low voltage infrastructure

2-event classification

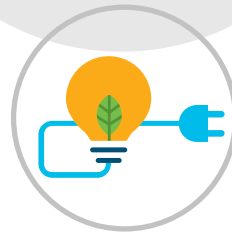
- Fast power negotiation without LLDP
- Physical layer negotiation < 1s

Perpetual PoE

- Uninterrupted PoE power during control plane reboot

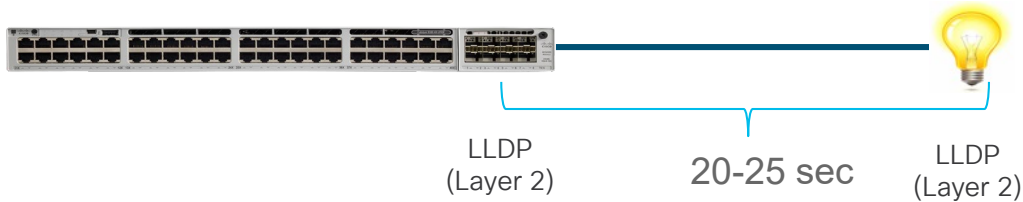
Fast PoE

- Bypasses IOS control plane boot
- Restores power to PD within 30 sec of power resumption

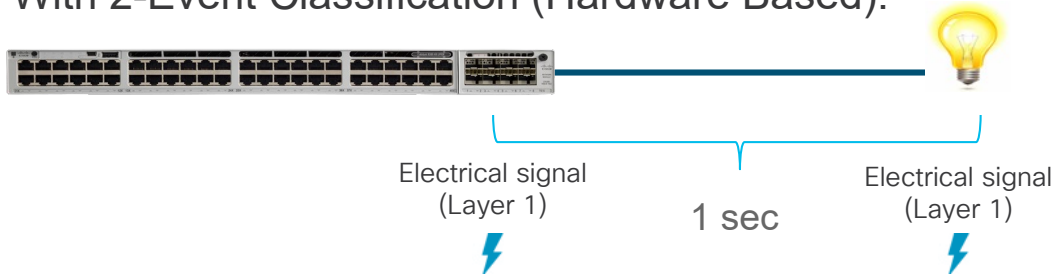


2-Event Classification

Without 2-Event Classification:



With 2-Event Classification (Hardware Based):



- Class 4 PD gets 30W even before the link comes up, otherwise 15.4W
- Type-2 PSE sends classification voltage (2nd Handshake) pulse to Class 4 PD (PoE+, 12.96-25.5 W) confirming it is a high power capable PSE
- 30W of power is allocated for PD (PD draws up to 25.5W)
- Both PSE and PD need to support 2-Event
- No waiting for CDP/LLDP exchange for PoE+ power levels

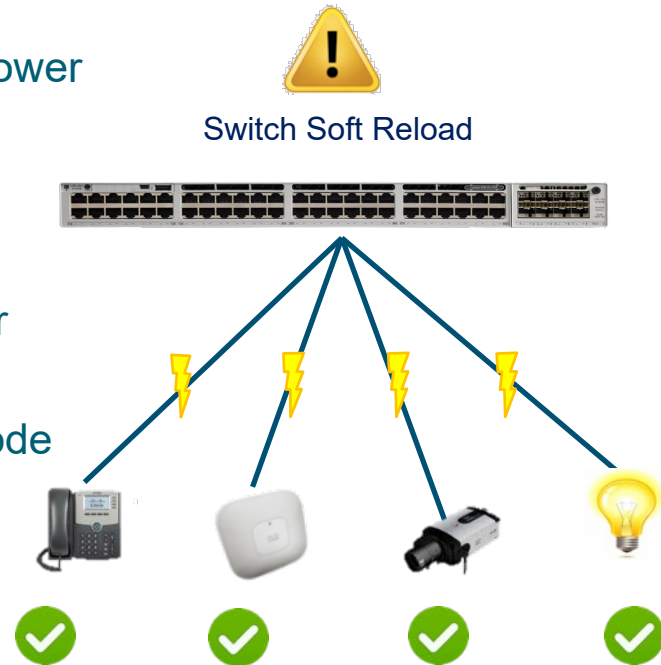
```
Switch> enable
Switch# configure terminal
Switch(config)# interface gigabitethernet2/0/1
Switch(config-if)# power inline port 2-event
Switch(config-if)# end
```

Perpetual PoE/UPOE

PoE devices connected to switch stay powered even on switch reload

- PoE devices continue to get Last Negotiated Power
- Applicable to “Soft” Reload – image upgrade, software crash, manual reboot
- Supported with stacking deployments
- Not applicable during power outage to switch or power supply removal
- Not applicable when switch is in hibernation mode

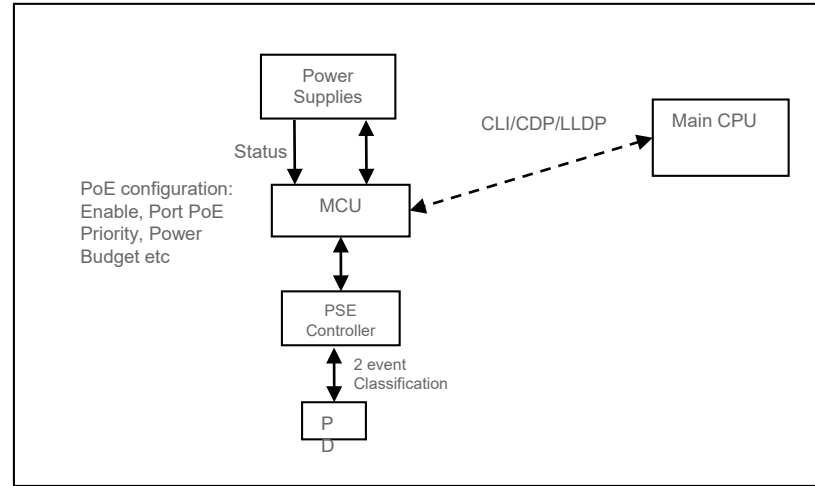
```
Switch> enable
Switch# configure terminal
Switch(config)# interface gigabitethernet2/0/1
Switch(config-if)# power inline port perpetual-poe-ha
Switch(config-if)# end
```



Fast PoE/UPOE

- Remembers the last power drawn from a particular PSE port
- Restores power to PD in less than 30 seconds post restoration of power
- Works even before IOS comes up
- Allocates last power (stored in NVRAM) drawn from PDs
- Works in stacking deployments

[“perpetual-poe-ha” is a prerequisite to “poe-ha”]



* In case of UPOE, since the PD relies on LLDP to get to higher power levels, PD may still need to wait till the IOS comes up and LLDP packet exchanges happen

```
Switch> enable
Switch# configure terminal
Switch(config)# interface gigabitethernet2/0/1
Switch(config-if)# power inline port perpetual-poe-ha
Switch(config-if)# power inline port poe-ha
Switch(config-if)# end
```

StackPower – Overview

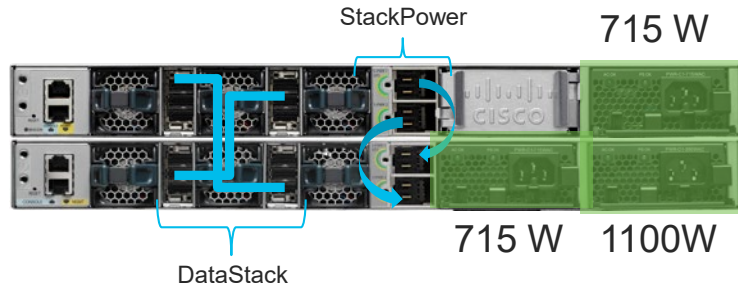
“Zero-Footprint” RPS deployment



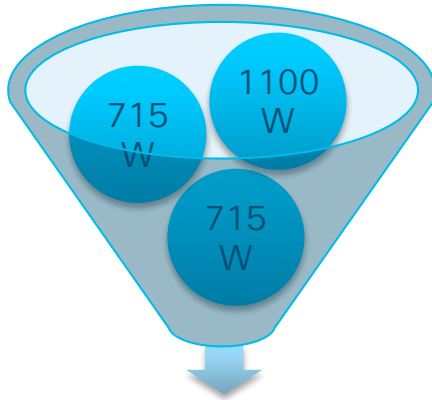
Intelligent power management
Cisco StackPower

- Provides RPS functionality with **Zero RPS Footprint**
- **Pay-as-you-grow** architecture – similar to the Data Stack
- **1+N Redundancy** with Inline Power
- Up to **4 Switches** in a StackPower Ring
- **Multiple StackPower** Possible within one Data Stack
- Up to **9 Switches** in a star topology with XPS

How StackPower Works?

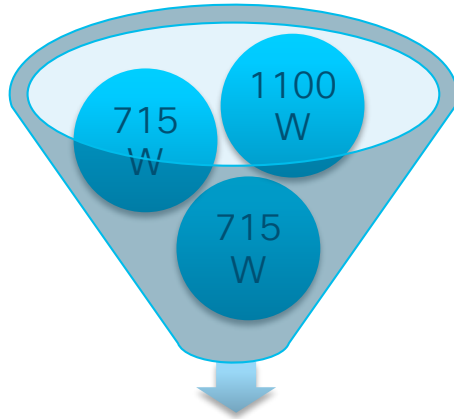


- Pools power from all power supplies
- All switches in StackPower domain share available power in Pool
- Each switch is given their minimum power Budget



Total Input Power = 2530W

Power Budget Modes

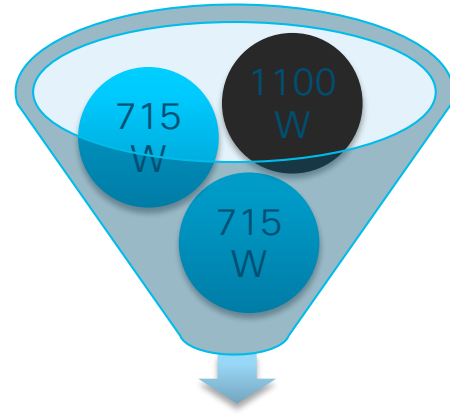


2530W - 30W

Power Sharing Mode

- The Default Mode
- Sum of all power supplies - 30~60W

Load shedding does not occur in redundant mode unless two or more power supplies fail, because the largest power supply is used as a backup power source.



1430W - 30W

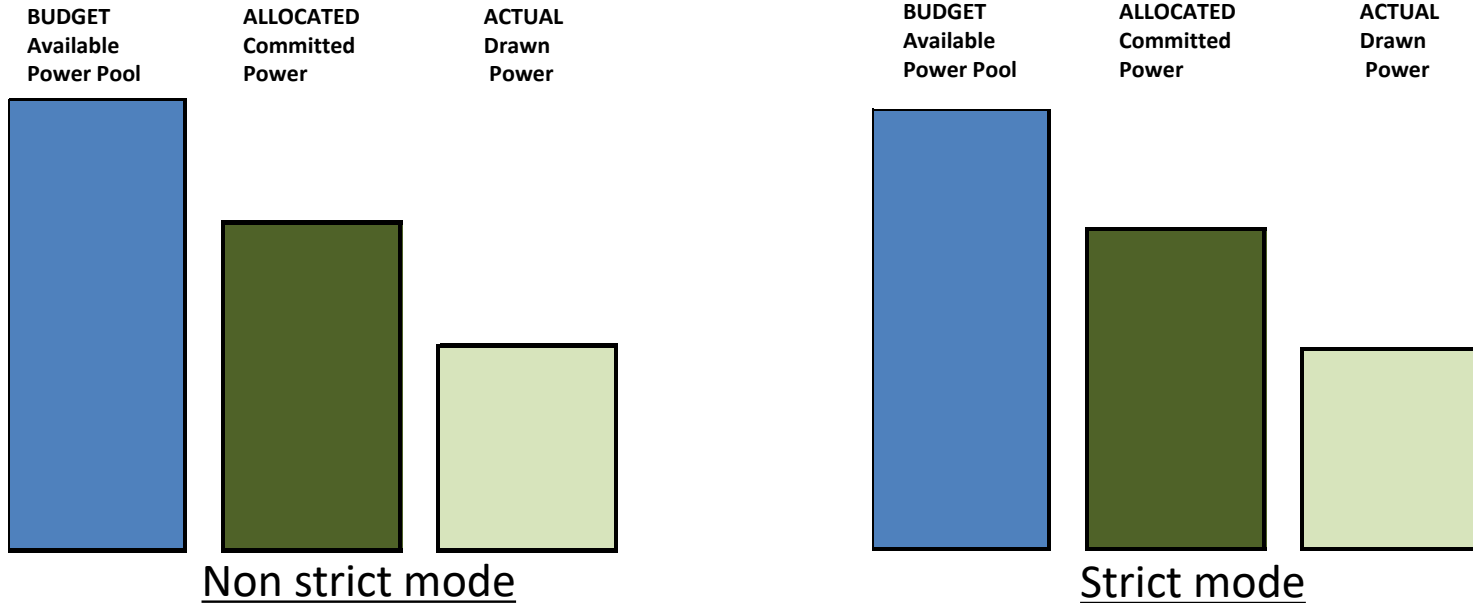
Redundant Mode

- User Configurable
- Sum of all power supplies - Largest PS - 30~60W

```
Switch(config)# stack-power stack power1  
Switch(config-stackpower)# mode redundant  
Switch(config-stackpower)# exit
```


Enforcement Modes

Strict & Loose Modes Control The Behavior of Load Shed



- Non Strict mode allows for a negative power budget
- Strict mode sheds load as soon as the power budget goes below the allocated power level

StackPower CLIs



9300-STACK#sh environment power all

SW	PID	Serial#	Status	Sys Pwr	PoE Pwr	Watts
1A	Not Present					
1B	Not Present					
2A	PWR-C1-1100WAC	LIT21212WAR	OK	Good	Good	1100
2B	PWR-C1-715WAC	LIT211549FX	OK	Good	Good	715
3A	PWR-C1-1100WAC	LIT21212NFY	OK	Good	Good	1100
3B	PWR-C1-1100WAC	DTN2145V53F	OK	Good	Good	1100
4A	PWR-C1-1100WAC-P	ART2216FDQJ	OK	Good	Good	1100
4B	Not Present					

9300-STACK#show stack-power budgeting

Power Stack Name	Stack Mode	Stack Topology	Total Pwr (W)	Rsvd Pwr (W)	Alloc Pwr (W)	Unused Pwr (W)	Num SW	Num PS
Powerstack-1	SP-PS	Ring	5115	35	1180	3900	4	5

SW	Power Stack Name	PS-A (W)	PS-B (W)	Power Budgt (W)	Alloc Power (W)	Avail Pwr (W)	Consumd Sys/PoE (W)	Pwr
1	Powerstack-1	0	0	1200	240	960	129	/0
2	Powerstack-1	1100	715	1230	240	990	131	/0
3	Powerstack-1	1100	1100	1230	240	990	127	/0
4	Powerstack-1	1100	0	1420	460	960	143	/0
Totals:					1180	3900	530	/0

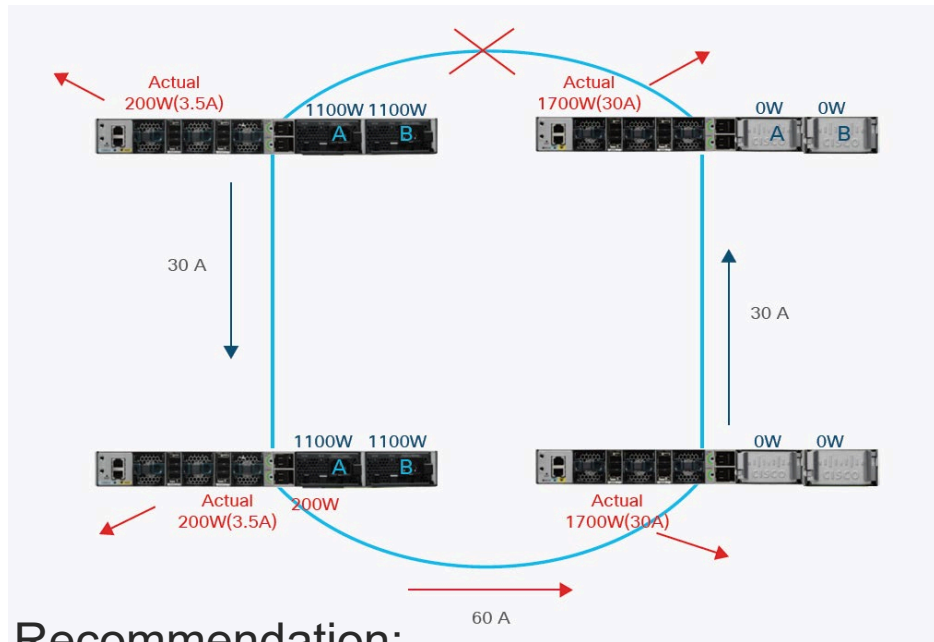
Power supply inputs, PIDs

PowerStack domain

Detailed power accounting

StackPower Best Practice

Balance Power supplies across the stack



Recommendation:

1. Balance PS across all systems
2. Fill up PS slot A on every switch in the stack, before using slot B on any switch!

Total Input Power = 4400W

Total Output Power = 3800W

The right half generates 0A but consumes 60A

Stackpower rated for ~55A

In failure scenario, Stackpower could be oversubscribed; console messages will warn about the condition and Intelligent load shed will occur.

Support Packages

Support Packages – IOS-XE 16.9.3, 16.10.1

Layer3 unicast stream

```
show tech-support platform layer3 unicast [vrf <VRFname>] destIp  
<destinationIP> srcIp <destinationIP>
```

Context aware data
collection

Layer3 multicast stream

```
show tech-support platform layer3 multicast group_ipAdd <mcast dest IP>  
srcIp <source IP>
```

IGMP snooping group

```
show tech-support platf igmp_snooping  
[group_ipAddr <IPv4McastGroupAddress> [vlan <vlan_ID>] ]
```

MLD snooping group

```
show tech-support platform mld_snooping group_ipv6Addr <IPv6 MLD  
Group Address>
```

QoS

```
show tech-support qos switch <id> [interface <interfaceID>]
```

Dot1x

```
show tech-support identity mac <mac address> interface <type_and_#>
```

REP - show tech-support rep [<segment_id>]

SDA (fabric) - show tech-support fabric

CPU - show tech-support qos switch <id> control-plane

PoE - show tech-support poe

Platform - show tech-support platform

ACL - show tech-support acl

Diagnostics - show tech-support diagnostic

Stack - show tech-support stack

Port (Interfaces) - show tech-support port

Serviceability Innovations

CPU Punt packet rate per interface

```
C9300#show platform software fed sw active ifm mappings
```

Interface	IF_ID	Inst	Asic	Core	Port	SubPort	Mac	Cntx	LPN	GPN	Type	Active
GigabitEthernet1/0/1	0x8	1	0	1	0	0	26	6	1	1	NIF	Y

IF_ID mapping

```
C9300#show platform software fed sw active punt rates interfaces 0x8
```

```
Punt Rate on Single Interfaces Statistics
```

```
Interface : GigabitEthernet1/0/1 [if_id: 0x8]
```

Received		Dropped	
-----		-----	
Total	: 3263	Total	: 0
10 sec average	: 4340	10 sec average	: 0
1 min average	: 4340	1 min average	: 0
5 min average	: 450	5 min average	: 0



Also works without specifying interface number

```
Per CPUQ punt stats on the interface (rate averaged over 10s interval)
```

Q no	Queue Name	Recv Total	Recv Rate	Drop Total	Drop Rate
0	CPU_Q_DOT1X_AUTH	0	0	0	0
1	CPU_Q_L2_CONTROL	1582	0	0	0
2	CPU_Q_FORUS_TRAFFIC	0	0	0	0
3	CPU_Q_ICMP_GEN	0	0	0	0
4	CPU_Q_ROUTING_CONTROL	0	0	0	0
5	CPU_Q_FORUS_ADDR_RESOLUTION	482	0	0	0

Interface level stat for punted traffic

Serviceability Innovations

CPU Punt & Inject packet capture with punt reason and platform context

to start the capture

debug platform software fed switch active [punt|inject] packet-capture start

to stop the capture

debug platform software fed switch active [punt|inject] packet-capture stop

to display the captured packets

show platform software fed switch active punt packet-capture [brief |detailed]

```
Punt packet capturing: disabled. Buffer wrapping: disabled
Total captured so far: 19 packets. Capture capacity : 4096 packets
```

```
----- Punt Packet Number: 1, Timestamp: 2019/07/03 20:43:16.087 -----
interface : physical: TenGigabitEthernet1/0/40[if-id: 0x00000030], pal: TenGigabitEthernet1/0/40 [if-id: 0x00000030]
metadata : cause: 96 [Layer2 control protocols], sub-cause: 0, q-no: 15, linktype: MCP_LINK_TYPE_LAYER2 [10]
ether hdr : dest mac: 0180.c200.0000, src mac: 34f8.e795.4b59
ether hdr : length: 39
```

```
----- Punt Packet Number: 2, Timestamp: 2019/07/03 20:43:17.919 -----
interface : physical: TenGigabitEthernet1/0/4[if-id: 0x00000040], pal: TenGigabitEthernet1/0/4 [if-id: 0x00000040]
metadata : cause: 11 [For-us data], sub-cause: 0, q-no: 2, linktype: MCP_LINK_TYPE_IP [1]
ether hdr : dest mac: cc70.edc4.f676, src mac: 700b.4f28.d964
ether hdr : ethertype: 0x0800 (IPv4)
ipv4 hdr : dest ip: 21.1.1.1, src ip: 21.1.1.2
ipv4 hdr : packet len: 100, ttl: 254, protocol: 1 (ICMP)
icmp hdr : icmp type: 0, code: 0
```

Supports display filters

Manageability Innovations

Manageability that fits your network



What can DNA Center do? Take a [Tour](#)

Need to add functionality to DNA Center? [Add applications](#)
Want to learn more about DNA Center? [Watch video](#)

Design

Model your entire network, from sites and buildings to devices and links, both physical and virtual, across campus, branch, WAN and cloud.

- Add site locations on the network
- Designate golden images for device families
- Create wireless profiles of SSIDs

Provision

Provide new services to users with ease, speed and security across your enterprise network, regardless of network size and complexity.

- Discover and provision switches to defined sites
- Provision VLANs and APs to defined sites
- Set up Campus Fabric across switches

Platform BETA

Use DNA-C Platform to unlock the full potential of DNA-C using APIs, integration capabilities and Data services.

- View the API Catalog
- Configure DNA - to - Third Party Integrations
- Schedule and Download - Data Sets and Reports

Policy

Use policies to automate and simplify network management, reducing cost and risk while speeding rollout of new and enhanced services.

- Segment your network as Virtual Networks
- Create scalable groups to describe your critical assets
- Define segmentation policies to meet your policy goals

Assurance

Use proactive monitoring and insights from the network, devices, and applications to predict problems faster and ensure that policy and configuration changes achieve the business intent and the user experience you want.

- Assurance Health
- Assurance Issues

Cisco DNA Center
Part of the Larger Network

WebUI
Small Branch - CPC Migration

Catalyst 9000 switches can be managed multiple ways

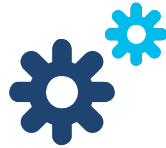


Embedded WebUI

Ease of Access



Build configurations



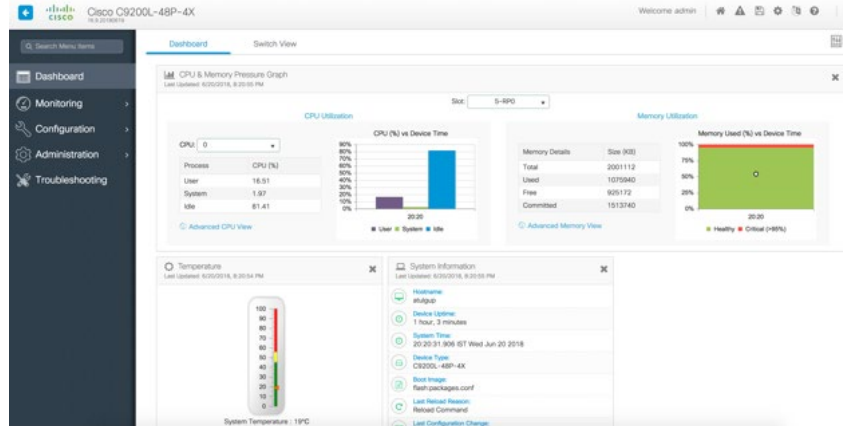
Intuitive Interface



Switch View



Troubleshooting made fun



CISCO *Live!*

WebUI - Switch view

The screenshot displays the Cisco WebUI interface for a switch stack. The top navigation bar includes the Cisco logo, device model 'Cisco C9200L-48P-4X', and user information 'Welcome admin'. The left sidebar contains menu items: Dashboard, Monitoring, Configuration, Administration, and Troubleshooting. The main content area is titled 'Switch View' and shows a stack of four switches:

- Switch No 4:** Active state. Port status grid shows ports 13 and 15 as green (UP). Uplink ports 01-04 are 100%.
- Switch No 5:** Provisioned state. Port status grid shows ports 13 and 25 as green (UP). Uplink ports 01-04 are 100%.
- Switch No 7:** Provisioned state. No port status grid visible.
- Switch No 8:** Standby state. Port status grid shows all ports as grey (DOWN). Uplink ports 01-04 are 100%.

Annotations with blue arrows point to specific elements:

- Active switch:** Points to the 'Active' label for Switch No 4.
- UP Ports:** Points to the green status indicators for ports 13 and 15 on Switch No 4.
- Standby switch:** Points to the 'Standby' label for Switch No 8.
- POE status:** Points to the 'PoE' radio button in the configuration options for Switch No 8.

A large blue bracket on the right side of the switch stack is labeled 'Switch Stack'.

WebUI - Troubleshooting made fun

Check the Core files

Cisco C9200L-48P-4X
16.9.20180619

Welcome admin

Troubleshooting

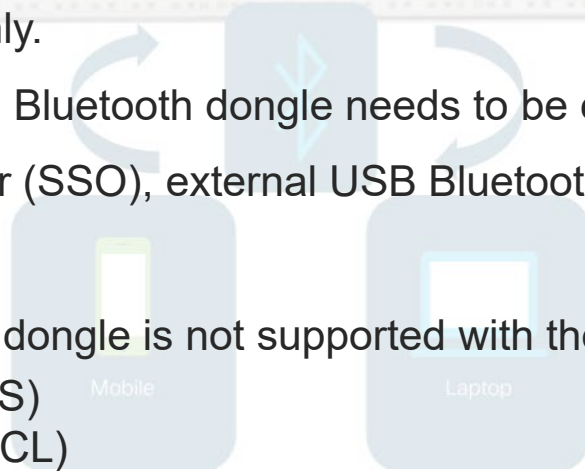
- Audit Support**
Check security , hardening, and the best practices aspect of the device config. Needs internet access
- Core Dump**
View the list of core files captured in the device
- Debug Bundle**
Capture require info like CLI outputs, logs as a single bundle for error reporting and debugging
- Packet Capture**
Capture packets with different filter options to feed into Wireshark for debugging
- Ping and Trace Route**
Check Ping-ability and Trace route info of a target destination through different sources
- Syslog**
Configure and View Syslog
- Web Server log**
View and Download Access and Error info of Web User Interface Logs

Ping and traceroute

View Syslog

Wireless manageability with Bluetooth

- Supported IOS XE version 16.12.1 onwards.
- Supported on all Catalyst 9000 series switches except C9500-32C, C9500-32QC, C9500-48Y4C, and C9500-24Y4C models
- Bluetooth 4.0 version only.
- In a stack, external USB Bluetooth dongle needs to be enabled on active switch.
- After Stateful Switchover (SSO), external USB Bluetooth needs to be re-enabled on the new active switch.
- External USB Bluetooth dongle is not supported with the following configurations:
 - Quality of Service (QoS)
 - Access Control List (ACL)



Bluetooth Dongle for wireless management

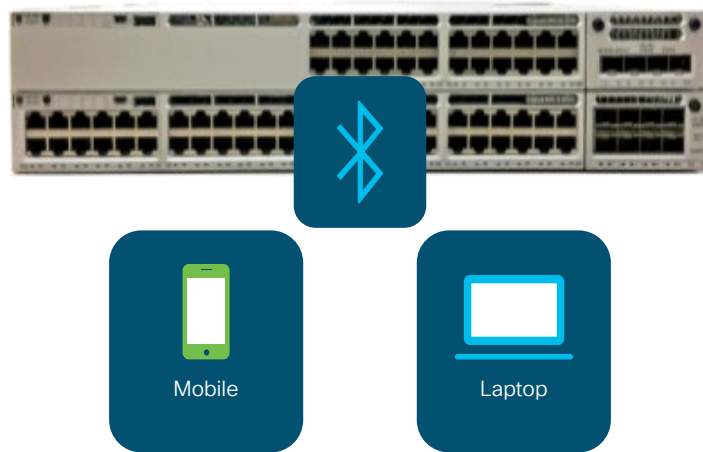
Step 1 Connect an external USB Bluetooth dongle to the USB Type A port on the Switch.

Step 2

```
Switch# configure terminal
Switch(config)# interface bluetooth 0/4
Switch(config-if)# enable
Switch(config-if)# no shut
```

Step 3

```
Switch(config-if)#bluetooth pin 1111
OR
Switch(config-if)#exit
Switch(config)#bluetooth pin 1111
```



[Configuring an External USB Bluetooth Dongle](#)

Bluetooth Dongle for wireless management

Command	Purpose
show ip interface bluetooth 0/4	Displays the usability status of Bluetooth interface.
show platform hardware bluetooth	Displays information about the Bluetooth interface.
show running include pin	Displays the current Bluetooth pin.

```
Device# show platform hardware bluetooth  
Controller:0:1a:7d:da:71:13  
Type:Primary  
Bus:USB  
State:DOWN  
Name:HCI Version:
```

[Configuring an External USB Bluetooth Dongle](#)

Summary

What?

Features that solve **your** problem

Which?

More than one way, which is the **best**?

How?

Configure, observe, **evaluate**, tweak



Appendix - Reference Scripts

- Find IPs in ARP table and ping them
- Upgrade IOS-XE version on Catalyst 9000
- Stack Event Checker
- Monitoring Cat9K CPU Punt cause & drops
- Cat9K QoS drops checker
- And more cool examples at <https://github.com/shashasi/BRKRST-2600>

Find IP addresses in ARP cache and ping them



Guestshell example

```
import cli

#Get a copy of the ARP table in Mgmt-vrf
arp_table = cli.execute('show ip arp vrf Mgmt-vrf')

#Find all IP addresses in the ARP table
hosts = re.findall(r"\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}", arp_table)
if hosts:
    for host in hosts:
        #ping each IP address
        ping_result = cli.execute("ping vrf Mgmt-vrf %s timeout 1" % host)
        #See if ping was successful
        success = re.findall(r"Success rate is 100 percent", ping_result)
        if success:
            cli.execute("send log %s is reachable" % host)
        else:
            cli.execute("send log %s is NOT reachable" % host)
```

Upgrade IOS-XE version on Catalyst 9000



EEM example

```
no event manager applet UPGRADE authorization bypass
event manager applet UPGRADE authorization bypass
event none maxrun 300
action 1.0 cli command "enable"
action 1.1 cli command "install add file flash:cat9k_iosxe.16.06.05.SPA.bin activate
commit" pattern "y\n"
action 1.2 cli command "y" pattern "y\n"
action 1.3 puts "Fixing to reload"
action 1.5 cli command "y"
```

Stack event checker

Guestshell example – Part 1/2



```
import re, time, cli

# check stack-ports summary
show_cmd = 'show switch stack-ports summary'
re.search('\d+\/\d+\/\d+\/s+?w+.*(\d{2,})', show_cmd)
summary_op = cli.execute(show_cmd)
pattern = '(\d+\/\d+\/\d+\/s+?w+.*(\d{2,}))'
if summary_op:
    for line in summary_op.splitlines():
        link_ok = re.search(pattern, line)
        if link_ok:
            cli.execute("send log" + " Stack port '%s' has %s 'Changes to LinkOK'"%(link_ok.group(1), link_ok.group(2)))

# collect switch and its asic count
show_switch_op = cli.execute('show switch')
stack_numbers = re.findall('[*\|s] (\d+)\s+?(?:Active|Standby|Member)', show_switch_op)
switch_asic_dict = {}
for stack in stack_numbers:
    show_cmd = 'show platform hardware fed switch %s fwd-asic drops exceptions | count ASIC'%(stack)
    asic_op = cli.execute(show_cmd)
    if asic_op:
        count = re.search('Number of lines which match regexp \s= (\d+)', asic_op)
        if count:
            switch_asic_dict.update({stack:count.group(1)})

# SDP counter check for non zero count against Tx Fail or Rx Fail
pattern = re.compile(r'(\w+.*)\s+?\d+\s+?(((\d{2,})|[1-9])\s+\d+\s+(0))|((0)\s+?\d+\s+(\d{2,})|[1-9]))')
for stack in stack_numbers:
    show_cmd = 'show platform software stack-mgr switch %s r0 sdp-counters'%(stack)
    sdp_counter_op = cli.execute(show_cmd)
    if sdp_counter_op:
        for line in sdp_counter_op.splitlines():
            match = pattern.search(line)
            if match:
                message = match.group(1).strip()
                tx_fail = match.group(4)
                rx_fail = match.group(8)
                if not tx_fail:
                    tx_fail = '0'
                if not rx_fail:
                    rx_fail = '0'
                cli.execute("send log" + " '%s' has %s Tx_Fail and %s Rx_Fail counters"%(message, tx_fail, rx_fail))
```

Stack event checker



Guestshell example – Part 2/2

```
# Register's snapshot
cmd_list = ['show platform hardware fed switch %s fwd-asic register read register-name SifRacDataCrcErrorCnt asic %s', 'show
platform hardware fed switch %s fwd-asic register read register-name SifRacRwCrcErrorCnt asic %s', 'show platform hardware fed
switch %s fwd-asic register read register-name SifRacPcsCodeWordErrorCnt asic %s', 'show platform hardware fed switch %s fwd-asic
register read register-name SifRacInvalidRingWordCnt asic %s']

snapshot = {}
for stack in stack_numbers:
    for asic in range(int(switch_asic_dict[stack])):
        # Loop for 2 snapshot
        for x in range(2):
            snapshot.update({x: []})
            # collect value for all 4 commands
            for cmd in cmd_list:
                sh_cmd = cmd%(stack, asic)
                output = cli.execute(sh_cmd)
                value = re.search('count\s+?:\s(\w+)', output)
                if value:
                    # check for hex 3+ digits
                    if int(value.group(1), 16) > int('0x99', 16):
                        cli.execute("send log" + " '%s' has '%s' count"%(sh_cmd, value.group(1)))
                        snapshot[x].append((sh_cmd, value.group(1)))
                else:
                    snapshot[x].append([sh_cmd, 'None'])
            # wait for 5 sec and collect again for above 4 commands
            time.sleep(5)
        # compare two snapshot values
        for index, cmd in enumerate(cmd_list):
            if snapshot[1][index][1] and snapshot[0][index][1]:
                if snapshot[1][index][1] > snapshot[0][index][1]:
                    cli.execute("send log" + " '%s' has increased from '%s' to '%s' within 5 seconds"%(snapshot[0][index][0],
                    snapshot[0][index][1], snapshot[1][index][1])
```

Monitoring Cat9K CPU Punt cause & drops



Guestshell example

```
import re,time,cli
import sys
```

```
#Get output of following commands
```

```
#(a) sh platform software fed sw active punt cause summary
```

```
#(b) sh platform hardware fed switch active qos queue stats internal cpu policer
```

```
#(c) sh platform software fed sw active punt cpuq all
```

Raw data

```
sh_sw_punt_cause = cli.execute('show platform software fed switch active punt cause summary')
sh_sw_cpu_int = cli.execute('show platform hardware fed switch active qos queue stats internal cpu policer')
sh_sw_punt_cpuq = cli.execute('show platform software fed switch active punt cpuq all')
```

Regex to match desired pattern

```
#Look for any non zero drop count. If found generate log along with the cause and non zero field.
```

```
non_zero_values_punt_cause = re.findall(r"\n\d+?\s+(\S+.*?)\s+\d+?\s+?([1-9][0-9]*?)\s+\n", sh_sw_punt_cause)
if non_zero_values_punt_cause:
    for cause, non_zero in non_zero_values_punt_cause:
        cli.execute("send log" + " Drop found %s, cause info %s. Check 'show platform software fed switch active punt cause summary'" % (non_zero, cause))
```

Monitoring Cat9K CPU Punt cause & drops



Guestshell example

```
##looking for any cpu policer queue dropping packets. If found generate alert along with the queue and corresponding non zero field.
```

```
cpu_int_queue = re.findall(r"\d?\s?\d?\s+(.?)\s+Yes.+?(\d?)\s?\n", sh_sw_cpu_policer)
if cpu_int_queue:
    for queue, dropped in cpu_int_queue:
        if int(dropped)>0:
            cli.execute("send log" + " Non zero value %s found for %s. Check 'show platform hardware led
switch active qos queue stats internal cpu policer'" % (dropped, queue))
else:
    cli.execute("send log" + " No queue found dropping any packets")
```

Regex to match desired pattern

Sends log

Monitoring Cat9K CPU Punt cause & drops



Guestshell example

```
#Look for any non zero count following fields -
#Send to IOSd failed count
#RX suspend count
#RX unsuspend send failed count
#RX dropped count
#RX non-active dropped count
#RX conversion failure dropped
#RX spurious interrupt

punt_cpuq = 0
non_zero_values_punt_cpuq = re.findall(r"(CPU Q Id\s+?\: (\d{1,}) (?:(?!?:CPU Q Id)[\s\S])*)", sh_sw_punt_cpuq)
if non_zero_values_punt_cpuq:
    for cpq_entry, cpu_q_id in non_zero_values_punt_cpuq:
        match = re.findall(r"(Send to IOSd failed count|RX suspend count|RX unsuspend send failed count|RX dropped count|RX non-active dropped count|RX conversion failure dropped|RX spurious interrupt)\s+?\: ([1-9][0-9]{0,})", cpq_entry)
        if match:
            punt_cpuq = 1
            for field, non_zero_value in match:
                cli.execute("send log" + " Non zero value %s found for %s - CPU Q ID %s. Check 'show platform software fed switch active punt cpuq all'" % (non_zero_value, field, cpu_q_id))

#If no drops are found, generating no problem found alert.
if not non_zero_values_punt_cause and not punt_cpuq:
    cli.execute("send log" + " No problem found")
```

Regex to match desired pattern

Cat9K QoS drops checker



Use case

```
Cat9K#show platform hardware fed sw active qos queue stats int Gi2/0/1
```

```
-----  
Queue Buffers Enqueue-TH0 Enqueue-TH1 Enqueue-TH2  
-----  
0      0          0          0          756751  
1      0          0          0          452  
2      0          0          0          37645  
3      0          0          0          0  
4      0          0          0          0  
5      0          0          0          0  
6      0          0          0          0  
7      0          0          0          0  
-----
```

```
-----  
Queue Drop-TH0 Drop-TH1 Drop-TH2 SBufDrop QebDrop  
-----  
0      0          0          0          9393      0          0  
1      0          0          0          0          0          0  
2      0          0          0          0          0          0  
-----
```

```
Cat9K#show platform hardware fed sw active qos queue stats inter Gi2/0/1
```

```
-----  
Queue Buffers Enqueue-TH0 Enqueue-TH1 Enqueue-TH2  
-----  
0      0          0          0          978374  
1      0          0          0          934  
2      0          0          0          37989  
3      0          0          0          0  
4      0          0          0          0  
5      0          0          0          0  
6      0          0          0          0  
7      0          0          0          0  
-----
```

```
-----  
Queue Drop-TH0 Drop-TH1 Drop-TH2 SBufDrop QebDrop  
-----  
0      0          0          0          10393      0          0  
1      0          0          0          0          0          0  
2      0          0          0          0          0          0  
-----
```

Cat9K QoS drops checker - continued



Guestshell example

```
import re,time,cli
import sys
```

```
#Get show interface status output
```

```
show_int_status_op = cli.execute('show interface status')
show_running_op = cli.execute('show running-config')
```

Raw data

```
#Find all interfaces that are in up/up state. Look for connected in show interface status output.
```

```
up_interfaces_list = re.findall(r"(Gi\d\\d\\d\\d{1,2}|Te\d\\d\\d{1,2}).+?connected", show_int_status_op)
```

```
if not up_interfaces_list:
```

```
    cli.execute("send log" + " Need interface is found to be up!")
```

```
    exit()
```

Regex to match desired pattern

```
#Check if drops are happening in any queue-threshold by comparing 2 snapshots with 2 secs time interval for the up interfaces
```

```
intf_drop_traffic = []
```

```
for up_interface in up_interfaces_list:
```

```
    snapshot_1 = cli.execute('show platform hard fed sw active qos queue stats interface ' + up_interface)
```

```
    time.sleep(2)
```

Introduce delay

```
    snapshot_2 = cli.execute('show platform hard fed sw active qos queue stats interface ' + up_interface)
```

Cat9K QoS drops checker - continued



Guestshell example

```
for queue in range (0, 8):
    match_2 = re.search(r"Drop Counters[^\!]+?(%d)\s+(\d+)\s+(\d+)\s+(\d+)\s+(\d+)\s+(\d+)\s+(\d+)\n"
% (queue), snapshot_2)
    match_1 = re.search(r"Drop Counters[^\!]+?(%d)\s+(\d+)\s+(\d+)\s+(\d+)\s+(\d+)\s+(\d+)\s+(\d+)\n"
% (queue), snapshot_1)
    for threshold in range (2, 7):
        if match_2:
            drop_count_2 = int(match_2.group(threshold))
        if match_1:
            drop_count_1 = int(match_1.group(threshold))
        if match_2 and match_1:
            if drop_count_2 > drop_count_1:
                intf_drop_traffic.append(up_interface)
            if threshold < 5:
                cli.execute("send log" + " Drop-Th%d is dropping traffic in queue %d. Drop count is
%d. Check 'show platform hard fed sw active qos queue stats interface %s'" % (threshold-2, queue,
drop_count_2, up_interface))
            elif threshold == 5:
                cli.execute("send log" + " SBufDrop is incrementing queue %d. Drop count is %d. Check
'show platform hard fed sw active qos queue stats interface %s'." % (queue, drop_count_2, up_interface))
            elif threshold == 6:
                cli.execute("send log" + " QebDrop is incrementing in queue %d. Drop count is %d.
Check 'show platform hard fed sw active qos queue stats interface %s'." % (queue, drop_count_2,
up_interface))
```

Regex to match
desired pattern

Check if drop counter
incremented between snapshots

Complete your online session survey



- Please complete your session survey after each session. Your feedback is very important.
- Complete a minimum of 4 session surveys and the Overall Conference survey (starting on Thursday) to receive your Cisco Live t-shirt.
- All surveys can be taken in the Cisco Events Mobile App or by logging in to the Content Catalog on ciscolive.com/emea.

Cisco Live sessions will be available for viewing on demand after the event at ciscolive.com.

Continue your education



Demos in the
Cisco Showcase



Walk-In Labs



Meet the Engineer
1:1 meetings



Related sessions



Thank you





You make **possible**